Interactive Drama, Art and Artificial Intelligence

Michael Mateas
December 2002
CMU-CS-02-206

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Thesis Committee:

Joseph Bates (co-chair)
Jaime Carbonell (co-chair)
Ian Horswill, Northwestern University
Janet Murray, Georgia Tech
Simon Penny, University of California, Irvine

© 2002 Michael Joseph Mateas
All rights reserved

This research was funded in part through fellowships from the Litton and Intel Corporations. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect those of the sponsors.

Keywords: artificial intelligence, art, entertainment, believable agents, interactive drama, interactive characters, interactive story

Abstract

Artificial intelligence methods open up new possibilities in art and entertainment, enabling rich and deeply interactive experiences. At the same time as AI opens up new fields of artistic expression, AI-based art itself becomes a fundamental research agenda, posing and answering novel research questions that would not be raised unless doing AI research in the context of art and entertainment. I call this agenda, in which AI research and art mutually inform each other, *Expressive AI*. Expressive AI takes seriously the problem of building intelligences that robustly function outside of the lab, engaging human participants in intellectually and aesthetically satisfying interactions, which, hopefully, teach us something about ourselves.

This thesis describes a specific AI-based art piece, an interactive drama called *Façade*, and describes the practice of Expressive AI, using *Façade*, as well as additional AI-based artwork described in the appendices, as case studies.

An interactive drama is a dramatically interesting virtual world inhabited by computer-controlled characters, within which the player experiences a story from a first person perspective. Over the past decade, there has been a fair amount of research into believable agents, that is, autonomous characters exhibiting rich personalities, emotions, and social interactions. There has been comparatively little work, however, exploring how the local, reactive behavior of believable agents can be integrated with the more global, deliberative nature of a story plot, so as to build interactive, dramatic worlds. This thesis presents *Façade*, the first published interactive drama system that integrates character (believable agents), story (drama management) and shallow natural language processing into a complete system. *Façade* will be publicly released as a free download in 2003.

In the *Façade* architecture, the unit of plot/character integration is the dramatic beat. In the theory of dramatic writing, beats are the smallest unit of dramatic action, consisting of a short dialog exchange or small amount of physical action. As architectural entities, beats organize both the procedural knowledge to accomplish the beat's dramatic action, and the declarative knowledge to sequence the beat in an evolving plot. Instead of conceiving of the characters as strongly autonomous entities that coordinate to accomplish dramatic action through purely local decision-making, characters are instead weakly autonomous – the character's behavioral repertoire dynamically changes as beats are sequenced. The *Façade* architecture includes ABL (A Behavior Language), a new reactive planning language for authoring characters that provides language support for joint action, and a drama manager consisting of both a language for authoring the declarative knowledge associated with beats and a runtime system that dynamically sequences beats.

Façade is a collaboration with independent artist and researcher Andrew Stern.

Expressive AI is not the "mere" application of off-the-shelf AI techniques to art and entertainment applications. Rather, Expressive AI is a critical technical practice, a way of doing AI research that reflects on the foundations of AI and changes the way AI is done. AI has always been in the business of knowing-by-making, exploring what it means to be human by building systems. Expressive AI just makes this explicit, combining the thought experiments of the AI researcher with the conceptual and aesthetic experiments of the artist. As demonstrated through *Façade* and the other systems/artworks described in the appendices, combining art and AI, both ways of knowing-by-making, opens up new research questions, provides a novel perspective on old questions, and enables new modes of artistic expression. The firm boundary normally separating "art" and "science" is blurred, becoming two components of a single, integrated practice.

Acknowledgements

I would like to thank my committee, Joseph Bates, Jaime Carbonell, Ian Horswill, Janet Murray, and Simon Penny, for their comments, suggestions, and guidance through the process of finishing this dissertation.

My various office mates during my years at CMU, namely Adam, Karen, Kamal, Bryan, Andrew, Eric, Rune, and Alex, provided stimulating and entertaining discussions.

Thanks to my fellow interdisciplinarians Phoebe Sengers and Belinda Thom for their intellectual and emotional companionship. Working at the boundaries between fields is a lonely and difficult endeavor – Phoebe and Belinda provided much needed intellectual stimulation, as well as moral support when The Man would get me down.

Though my discussions with previous Oz Project members Bryan Loyall, Scott Neal Reilly and Peter Weyhrauch, have unfortunately been infrequent, I have always found these discussions deeply stimulating.

I have learned an incredible amount from my collaborators: Marc Böhlen (*Office Plant #1*), Steffi Domike and Paul Vanouse (*Terminal Time*), and Andrew Stern (*Façade*). These collaborations have provided some of my most stimulating and rewarding experiences at CMU.

I spent many long hours working in the congenial atmosphere of the Squirrel Hill coffee shops 61C and Coffee Tree. Thanks to both establishments for providing good coffee, a nice work environment, and needed distraction.

Many other people have made my time both at CMU and in Pittsburgh entertaining and rewarding. Thanks to all of our friends in Pittsburgh for making it a fun place to live.

My parents, Bob and Jackie, and my sister, Kathleen, have always provided me with love and support.

We adopted our first cat, Monongahela, while in Pittsburgh. Thanks to her for being so cute and fuzzy.

Many more people in my life have ultimately contributed to this work than I can name here. If you feel left out of these acknowledgements, consider yourself acknowledged.

And finally, I thank Anne for her constant love and companionship, and for providing much needed balance in my life.

Contents

CHAPTER 1 INTRODUCTION 1

STRUCTURE AND CONTRIBUTIONS 2
Interactive Drama 3
Expressive AI 5
INTERACTIVE DRAMA AND THE OZ PROJECT 6
Drama = Character + Story 7
Believable Agents 8
Drama Management 9
Presentation 10
Oz Research Philosophy 10
ADDITIONAL SYSTEMS 12
Subjective Avatars 12
Office Plant #1 12
Terminal Time 13
FAÇADE COLLABORATION 14
DISSERTATION OUTLINE 15

CHAPTER 2 INTERACTION AND NARRATIVE 19

APPROACHES 19
Commercial Computer Games 19
Emergent and Player Constructed Narrative 20
Narrative and New Media Art 20
Electronic Literature 21
Interactive Drama 22
A NEO-ARISTOTELIAN THEORY OF INTERACTIVE DRAMA 22
Defining Interactive Drama 22
Murray's Aesthetic Categories 23
Integrating Agency into Aristotle 24
Clarification of the Conceptual Experiment 29
Technical Agenda 31
CRITIQUES OF INTERACTIVE DRAMA 32
A Specific Ludological Critique 32
Narrativist Critiques of Interactive Drama 34
Middle Ground Positions 35
FAÇADE DESIGN GOALS 36
Project Goals 36
Story Requirements 37
The Story 38

CHAPTER 3 THE *FAÇADE* ARCHITECTURE 40

AUTONOMY AND STORY-BASED BELIEVABLE AGENTS 40
INTEGRATING PLOT AND CHARACTER WITH THE DRAMATIC BEAT 43

Beats Become Architectural Entities *43*
The Function of Beats *44*
A Response to the Problem of Autonomy *44*
JOINT GOALS AND BEHAVIORS 45
NATURAL LANGUAGE PROCESSING 46
Phase I: Surface Text to Discourse Acts *47*
Phase II: Discourse Acts to Reactions *48*
OVERVIEW OF THE ARCHITECTURE 48
Story World *49*
Drama Manager *50*
Natural Language Processing *50*

CHAPTER 4 AI AND ART 52

THE CLASSICAL/INTERACTIONIST AI DEBATE 52
Characterizing Classical and Interactionist AI *52*
Interactionist AI's Affinity with Cultural Theory *54*
AI & CULTURAL PRODUCTION 55
The Limitations of Agent as Metaphor *55*
Cultural Production vs. AI. *56*
A Random Walk Around the AI Landscape *58*
Artistic Practice Transforms AI. *62*
AUTHORSHIP 63
Metaphors Structuring AI-based Art Practice *63*
TOWARDS AN INTEGRATED PRACTICE 65

CHAPTER 5 *A BEHAVIOR LANGUAGE* 66

HAP 66
Example Behaviors *67*
Step and Behavior Annotations *70*
Conflicts *72*
Decision Cycle *72*
Em *73*
SUPPORT FOR JOINT ACTION 73
Introduction to Joint Goals and Behaviors *74*
Example of Basic Joint Goal and Behavior Support *76*
Additional Annotations for Joint Goals and Behaviors *79*
Full Complexity of the Joint Negotiation Protocol *80*
Coupled ABL Agents Form A Multi-Mind *89*
ADDITIONAL ABL EXTENSIONS 90
Support for Asynchronous Sensory-Motor Systems *90*
Meta-ABL *93*
Working Memory Extensions *98*
Step posting *99*
Atomic *99*
RELATED WORK 100
FUTURE WORK 101
Transition Behaviors *101*

Synchronization *102*

CHAPTER 6 ABL IDIOMS IN *FAÇADE* **105**

INTRODUCTION 105
BODY RESOURCES 105
BEAT GOALS AND HANDLERS 109
Beat Goals *109*
Handlers *112*
Interaction = Beat Goals + Handlers *117*
PERFORMANCE BEHAVIORS 117
STAGING AND LONG-TERM BEHAVIORS 121
BEAT LOGIC AND JOINT GOALS 122

CHAPTER 7 EXPRESSIVE AI: AFFORDANCES AND SEMIOTICS **124**

AFFORDANCES 124
Interpretive Affordance *124*
Authorial Affordance *125*
Combining Interpretive and Architectural Concerns *127*
AN INTRODUCTION TO SEMIOLOGY 127
THE CODE MACHINE AND THE RHETORICAL MACHINE 131
The Code System *132*
The Rhetorical System *137*
Idioms *142*
Generality of the Double Machine *142*
CONCLUSION 143

CHAPTER 8 THE *FAÇADE* DRAMA MANAGER **145**

DRAMA MANAGEMENT 145
Global vs. Local Agency *145*
The Combinatorics of Global Agency *146*
Story Policies *147*
THE BEAT MANAGER 148
The Beat Language *148*
The Beat Sequencer *150*
THE *FAÇADE* STORY DESIGN 156
Story Topics *156*
Story Values: Tension and Affinity *158*
Additional Beats *160*
Global Mix-Ins *160*
Beat Collection for Part I *161*
BEAT MANAGEMENT IDIOMS 163
Beat Clusters *163*
Support Beat Weight Boosts *165*
Story Topic Affinity Sets *166*
Story Topic Singlets *169*
EXAMPLE STORY TRACES 170
Abstract Traces *170*

Concrete Traces *172*
RELATED WORK 181
Interactive Story *181*
Story Generation *183*
FUTURE WORK 184

CHAPTER 9 EXPRESSIVE AI: DISCIPLINARY ISSUES 186

INTRODUCTION 186
WHY USE AI IN CULTURAL PRODUCTION? 186
Deep Interaction on Human Terms *186*
Tap into Rich Practice of Doubled Machines *187*
Exploring the Human by Making *188*
Build Microworlds with Human Significance *188*
Videogames *189*
FORMS OF AI-BASED ARTWORK 189
Procedural Portraits of Human Meaning-Making *189*
Characters *190*
Alien Presence *190*
Narrative *191*
Robotic Art *191*
Meta-Art *192*
EXPRESSIVE AI AS DISCIPLINE 192
Critiques of AI *193*
Critical Technical Practice *196*
CONCLUSION 199

CHAPTER 10 NATURAL LANGUAGE PROCESSING IN *FAÇADE* 200

PHASE I: SURFACE TEXT TO DISCOURSE ACTS 200
Discourse Acts *201*
The Template Language *203*
Compilation Strategy *210*
Runtime Processing *211*
Idiom for General Template Rules *212*
Templates And Ungrammatical Inputs *213*
PHASE II: DISCOURSE ACTS TO REACTIONS 214
Reaction Selection Architecture *214*
Global Mix-Ins *218*
Selection Examples *222*
RELATED WORK 223
FUTURE WORK 224

CHAPTER 11 CONCLUSION 225

INTERACTIVE DRAMA 225
The Dramatic Beat as an Architectural Entity *225*
Joint Goals and Behaviors *225*
Story Design *226*
Authorial Burden *227*

EXPRESSIVE AI 227
Critiques of AI *227*
Interpretive and Authorial Affordance *228*
Expressive AI is AI Research *229*

APPENDIX A SUBJECTIVE AVATARS 231

INTRODUCTION 231
WHY SUBJECTIVE AVATARS? 231
FASTFOOD WORLD 232
The Framework *232*
The Characters *232*
The Story *233*
SUBJECTIVE STATE 234
Emotional State *234*
Story Context *235*
NARRATIVE EFFECTS 236
Sensory Descriptions *236*
Stream of Thought *239*
RELATED WORK 239
CONCLUSION 240

APPENDIX B OFFICE PLANT #1 241

INTRODUCTION 241
CONCEPTS 242
Email space *242*
Text classification *243*
Plant behavior architecture *243*
Sculptural Presence *244*
PHYSICAL DESIGN 244
Implementing plant movement *245*
INTIMATE TECHNOLOGY 245
Acknowledgements *246*

APPENDIX C TERMINAL TIME 247

ARTISTIC GOALS 249
Documentary Form *249*
UTOPIAN NAVIGATION 249
AUDIENCE EXPERIENCE 250
KNOWLEDGE BASE 250
Upper Cyc Ontology *250*
Example Historical Event *250*
Inference Engine *251*
IDEOLOGICAL GOAL TREES 251
Tests for Event Applicability *252*
Plans for Event-level Story Generation *253*
RHETORICAL DEVICES 253
Prescope and Postscope Tests *253*

Rhetorical Device NLG Rule *254*
Example Rhetorical Device *254*
Story Generation *254*
NLG RULES 255
NLG Rule Syntax *255*
VIDEO SEQUENCING 256
CURRENT STATUS 257
PERFORMANCE EXPERIENCES 257
RELATED WORK 257
BIBLIOGRAPHY 259

CHAPTER 1
INTRODUCTION

Animating the inanimate, representing what it means to be human in dead matter, even creating life, has been a dream, a desire, running through human history. This dream can be seen operating in the myth of Pygmalion and Galatea, in which the sculptor Pygmalion creates the sculpture Galatea whom the gods bring to life, in the Golem of Prague, created from clay to protect the Jewish ghetto, and in 18th century mechanical automata such as the writing boy and harpsichord-playing girl of Jacquet-Droz, and the quacking, eating, and eliminating duck of Jacques de Vaucanson. The field of Artificial Intelligence (AI) is a modern incarnation of this dream, with computational systems, rather than stone, clay or clockworks, becoming the medium in which life is inscribed.

It is this dream, fueled by science fiction representations of AI such as Hal 9000 or Commander Data, that is the initial inspiration for many researchers entering the field. This dream is not just about modeling rational problem solvers, but about building machines that in some sense engage us socially, have emotions and desires, and, through our interactions with them, tell us something about ourselves. AI is a way of exploring what it means to be human by *building systems*. An AI architecture is a machine to think with, a concrete theory and representation of some aspect of the human world. Art also explores what it means to be human by *building concrete representations* of some aspect of the human world. Artists often explore aspects of humanity that have been under-explored or ignored in AI research. What Joseph Bates wrote about character animators in particular, applies to the arts in general:

> It can be argued that while scientists may have more effectively created [machines that act like] scientists, it is the artists who have come closest to understanding and perhaps capturing the essence of humanity that … AI researchers ultimately seek. [Bates 1994]

Artists, of course, have been in the business of representing life and the human world for all of human history, so it was natural for artists to quickly begin experimenting with cybernetic and AI techniques. Ed Inahtowicz's *Senster*, built in 1969 and 1970, is an early example of this fusion [Ihnatowicz 1986]. The *Senster* responded to audience noises, and would crane it's head, pull back in apparent alarm, and move it's head from bystander to bystander with amazingly life-like motions. Around the same time, Jack Burnham wrote the prophetic book *Beyond Modern Sculpture* [Burnham 1968] in which he related cybernetic art to the history of sculpture and predicted the rise of an aesthetics of artificial intelligence.

AI is also playing an ever bigger role in popular art, as manifest by the increasing use of AI techniques in contemporary video games. For example, in *Black and White*, the player's creature learns to autonomously accomplish tasks on the player's behalf using decision tree learning. In the *Sims*, the simulated people decide what to do next by hill climbing on a desire satisfaction landscape defined by objects in the world.

So this same dream, to represent the living, human world, has been a driving force in both the long history of art and the short history of AI. Combining these two ways of knowing-by-making opens a new path towards the AI dream, a path that takes

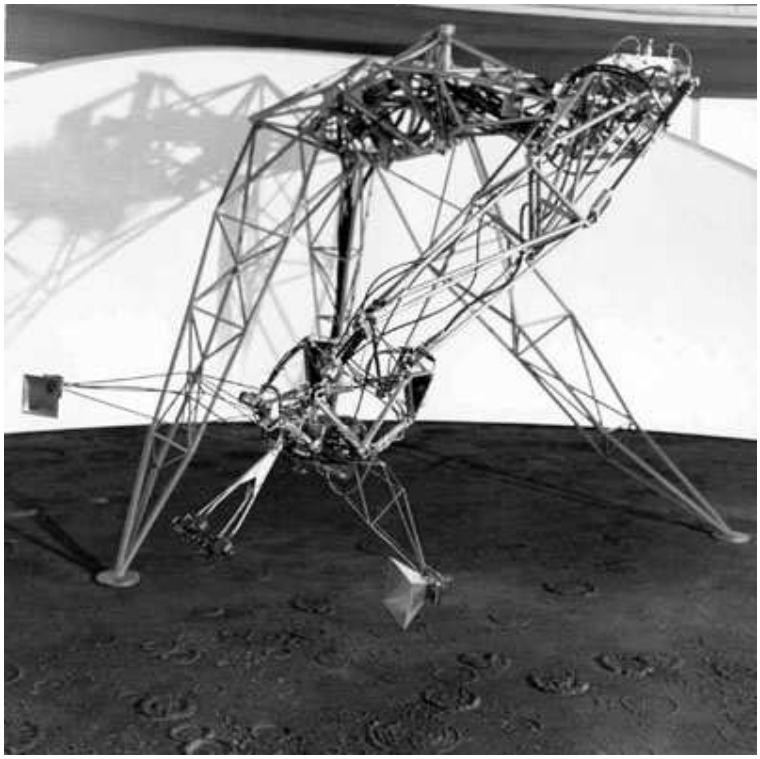

Figure 1-1. The *Senster* at Philip's Evoluon

seriously the problem of building intelligences that robustly function outside of the lab, engaging human participants in intellectually and aesthetically satisfying interactions, which, hopefully, teach us something about ourselves. I call this intertwined combination of AI research and art practice *Expressive AI*. Expressive AI has two major, interrelated thrusts:

1) *exploring the expressive possibilities of AI architectures* - posing and answering AI research questions that wouldn't be raised unless doing AI research in the context of an art practice, and
2) *pushing the boundaries of the conceivable and possible in art* - creating artwork that would be impossible to conceive of or build unless making art in the context of an AI research practice.

This thesis describes a specific AI-based art piece, an interactive drama called *Façade*, and describes the practice of Expressive AI, using *Façade*, as well as additional AI-based artwork described in the appendices, as case studies.

Structure and Contributions

In Expressive AI, technical research and artistic exploration are intertwined. Building an AI-based artwork such as *Façade* requires deep changes in both AI research and art practice; neither the researcher nor the artist can continue in a "business as usual" way. This dissertation attempts to mirror this intertwining in its structure, interleaving chapters that primarily describe technical and conceptual issues and contributions in *Façade*, with chapters that primarily explore theoretical, conceptual and disciplinary issues in Expressive AI. While this dissertation tries to tell an integrated story, I recognize that

readers will come with disparate backgrounds and interests. The dissertation outline at the end of this chapter serves as a guide to the reader.

Interactive Drama

Interactive drama concerns itself with building dramatically interesting virtual worlds inhabited by computer-controlled characters, within which the user (hereafter referred to as the player) experiences a story from a first person perspective [Bates 1992] (for more description of the aims of interactive drama, see page 6). Over the past decade there has been a fair amount of research into believable agents, that is, autonomous characters exhibiting rich personalities, emotions, and social interactions [e.g. Mateas 1999b; Bates, Loyall & Reilly 1992a; Blumberg 1996; Hayes-Roth, van Gent & Huber 1997; Lester & Stone 1997; Stern, Frank & Resner 1998]. There has been comparatively little work, however, exploring how the local, reactive behavior of believable agents can be integrated with the more global, deliberative nature of a story plot, so as to build interactive, dramatic worlds [Weyhrauch 1997; Blumberg & Galyean 1995]. The *Façade* dramatic world, and the AI architecture within which this world is implemented, explores one technique for integrating plot and character in the context of a first-person dramatic world. Building a *complete* dramatic world necessarily touches on many areas. The main areas of contribution are described below.

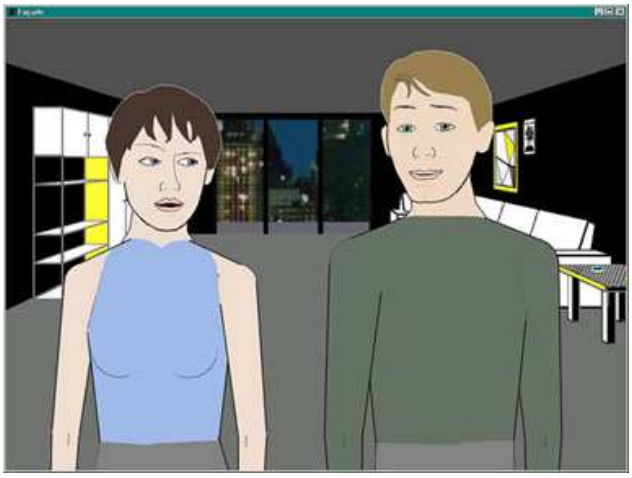

Figure 1-2. Screen-capture from *Façade*

Integrated Interactive Drama Architecture

The *Façade* architecture integrates story level interaction (drama management), believable agents, and shallow natural language processing, in the context of a first person, graphical, real-time interactive drama. To my knowledge, this is the first published architecture to integrate all these pieces[1]. The primary architectural contribution, besides achieving the integration itself, is architectural support for authoring dramatic beats, an architectural level that combines aspects of character and story.

[1] Zoesis, the company founded by former Oz Project leader Joe Bates and Oz Project members Bryan Loyall, Scott Neal Reilly, and Peter Weyhrauch, demonstrated an unpublished interactive drama architecture integrating believable agents, gestural language, adversary-search-based drama management, and a reactive musical score, at the 1999 AAAI Fall Symposium on Narrative Intelligence.

Reactive Planning Language for Believable Agents

ABL (**A B**ehavior **L**anguage, pronounced "able") is based on the Oz Project believable agent language Hap developed by A. B. Loyall [Loyall 1997, Bates, Loyall & Reilly 1992a]. The ABL compiler is written in Java and targets Java; the generated Java code is supported by the ABL runtime system. ABL modifies Hap in a number of ways, changing the syntax (making it more Java-like), generalizing the mechanisms by which an ABL agent connects to a sensory-motor system, and, most significantly, adding new constructs to the language, including language support for multi-agent coordination in the carrying out of dramatic action and rich support for reflection.

Broad and Shallow Natural Language Processing

In *Façade*, dialog is one of the primary mechanisms by which a player interacts with characters and influences how the story unfolds (the player enters their responses as typed text). Deep natural language processing (NLP) techniques are of necessity tightly focused on narrow, typically task-based domains (e.g. [Allen et. al. 1995; Ferguson et. al. 1996]). It is currently not possible to apply such techniques to broad, complex story domains such as *Façade*'s. Instead, the *Façade* NLP architecture provides, broad, shallow, story-specific support for language processing. Processing of player utterances is done in two phases.

1. Phase I maps surface text to discourse act(s), simple semantic representations of the social action performed by the surface text utterance. A custom rule language (implemented as a compiler targeting the forward chaining rule language Jess [Friedman-Hill 1995-2002]) supports authoring rules that match on textual patterns to recognize primitive features, and chain features to produce discourse acts.
2. Phase II takes the one or more discourse acts resulting from phase I (a single surface text utterance can result in multiple, potentially conflicting discourse acts) and selects character reaction(s) to the discourse acts.

Drama Management

A drama manager actively provides story structure for an ongoing interactive experience by intervening in the story world in such a way as to help a story to happen. The *Façade* drama manager is organized around the idea of the dramatic beat, the smallest unit of story change [McKee 1997]. As the story progresses, beats are sequenced in such a way as to be responsive to recent player interaction while providing story structure. The drama manager consists of a custom beat description language (implemented as a compiler targeting Java), and a runtime framework that, given a collection of compiled beats, maintains a changing probability distribution (probability of a beat being the next one selected) over the beats.

Story Design

The issues of interactive story design are at least as important as the architecture. In fact, the story design is tightly intertwined with the architecture; the story and architecture co-evolve as they provide mutual constraints and opportunities. Interactive story design questions include:

1. What are the re-sequenceable story pieces? What principles govern the design of the pool of pieces?

2. What is the range of player interaction supported within the story?
3. How is player interaction incorporated into both the local and global story structure?

This dissertation describes the *Façade* story design and the relationship between the story design and architecture.

Expressive AI

The field of Artificial Intelligence (AI) has produced a rich set of technical practices and interpretive conventions for building rich procedural representations of intelligent activity. Artists have begun to incorporate AI practices into their work, building an impressive range of work including robotic sculptures [Penny 2000; Ihnatowicz 1986], machines providing poetic commentary on visual stimuli [Huhtamo 1998], music generators [Cope 1996], painting generators [McCorduck 1991], interactive characters [Stern 1999; Johnson et. al. 1999], and narrative and poetry generators [Mateas, Vanouse & Domike 2000; Manurung, Ritchie & Thompson 2000]. One way of conceiving the relationship between AI and art is to view these artworks as "applications" of AI, that is, as the unproblematic use of "off the shelf" AI techniques in the service of art. This impoverished view assumes that an artist develops a conceptual and aesthetic plan for her work, then chooses AI solutions off the menu provided by AI research scientists. On the contrary, AI-based artwork raises research issues for which AI scientists not only don't have answers, but have not yet begun asking the questions. Further, the artist's conceptual and aesthetic exploration is not an independent "driver" of the research, providing the specifications for the technical work, but rather is deeply intertwined with the technical work; conceptual, aesthetic, and technical issues mutually inform each other. Expressive AI, my own practice of AI-based cultural production, takes both AI research and art making as first-class, interdependent goals.

The Goals of AI Research and Art

Arguments about the top-level research goals AI are sometimes structured as a debate between classical, or Good Old Fashioned AI (GOFAI), and interactionist, or embodied AI [Brooks 1990; Agre 1997a; CogSci 1993]. This debate has shaped much contemporary practice combining AI and cultural production, with practitioners commonly aligning themselves with the interactionist camp. However, combining AI and art is not as simple as identifying and employing a privileged technical tradition that is somehow peculiarly suited for art production. Rather, there are strong tensions between the fundamental goals of AI research and art, tensions that Expressive AI must navigate. This dissertation positions Expressive AI with respect to various AI research traditions, identifies the tensions between AI research and art practice, and describes how, despite these tensions, Expressive AI provides an alternative agenda for pursuing the AI quest.

Interpretive and Authorial Affordances

Art practice is concerned with communication and interpretation, with crafting art interventions, whether as tangible as painting or sculpture, as ephemeral as a performance, or as conceptual as a set of audience injunctions, that create for an audience a meaningful experience. AI has traditionally been engaged in the study of the possibilities and limitations inherent in the physical realization of intelligence [Agre 1997a]. The notions of interpretive and authorial affordance create a unified conceptual framework for these two apparently disparate concerns. Interpretive affordances are the

"hooks" supporting the audience interpretation of the operation of an AI system. Authorial affordances are the "hooks" an architecture provides for inscribing authorial intention within an AI system. This dissertation introduces the ideas of interpretive and authorial affordance, provides a semiotic analysis to further develop the notion of affordance, and uses these idea to present a unified agenda for AI and art.

Critical Technical Practice

Expressive AI is a form of critical technical practice, that is, a technical practice that actively reflects on its own philosophical underpinnings and, by bringing humanistic and artistic knowledge, approaches, and techniques to bear, opens up new technical directions and approaches. Agre coined the phrase "critical technical practice" to refer to research approaches that incorporate reflection on foundations into day-to-day technical practice itself:

> A critical technical practice would not model itself on what Kuhn called "normal science", much less on conventional engineering. Instead of seeking foundations it would embrace the impossibility of foundations, guiding itself by a continually unfolding awareness of its own workings as a historically specific practice. It would make further inquiry into the practice of AI an integral part of the practice itself. It would accept that this reflexive inquiry places all of its concepts and methods at risk. [Agre 1997a: 23].

This dissertation relates Expressive AI to other critical technical practices, positions Expressive AI relative to a number of critiques of AI, and describes disciplinary issues raised in negotiating the boundaries between AI and art.

The Range of AI-based Cultural Production

Expressive AI certainly didn't arise within a cultural and historical vacuum. There is a rich history of electronic media art, some of which explores AI techniques, but much of which has nothing to do with AI. This dissertation explores a number of questions that arise in relating AI-based art to the broader category of electronic media art.

1. The combination of AI and art may seem like an arbitrary conjunction of a specific technical subfield with art practice. Why not database art, network art, or scientific visualization art, all of which are active genres in electronic media art? What distinguishes AI-based art from other electronic media genres?
2. What is the history of AI-based art?
3. What is the range of AI-based art and entertainment? What categories of work and general desiderata can we currently envision?

Interactive Drama and the Oz Project

The Oz Project at CMU, lead by Bates [Bates 1992], has been studying interactive drama, both building architectures and completed worlds, since 1987. The Oz Project had a distinct vision for interactive drama and a distinct technical approach for achieving this vision. Though I joined the project near the end, having significant overlap only with Phoebe Sengers, *Façade's* approach to interactive drama certainly continues in the Oz tradition, being informed by and directly building upon the Oz Project approach. This section provides a brief Oz-centric overview of interactive drama [Mateas 1999b].

Drama = Character + Story

In an interactive drama the player finds herself in an immersive world, inhabited by personality-rich, robustly interactive characters. The player is free to move around the world, manipulate objects, and, most importantly, interact with the other characters. But all this activity isn't meandering, repetitious, or disjoint, but rather builds in compelling ways towards a climax and resolution. Thus, in an interactive drama, the player should experience a sense of *both* interactive freedom and a compelling, dramatic structure (what Weyhrauch terms "dramatic destiny" [Weyhrauch 1997]). Laurel was the first to define this notion of interactive drama in her 1986 dissertation *Toward the Design of a Computer-Based Interactive Fantasy System* [Laurel 1986:10-11]:

> An "interactive drama", then, is a first-person experience within a fantasy world, in which the user may create, enact, and observe a character whose choices and actions affect the course of events just as they might in a play. The structure of the system proposed in the study utilizes a playwriting expert system that enables first-person participation of the user in the development of the story or plot, and orchestrates system-controlled events and characters so as to move the action forward in a dramatically interesting way.

This definition, except for the proposed technology of expert systems for drama management, captures the Oz vision of interactive drama.

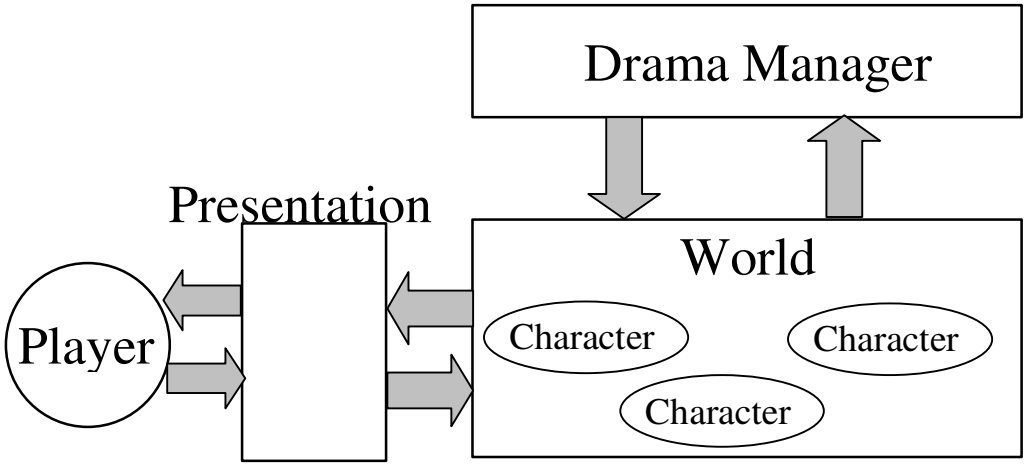

Figure 1-3. The Oz interactive drama architecture

The Oz Project organized its work around the architecture depicted in Figure 1-3, an architecture that treats both character *and* story as necessary ingredients of powerful dramatic experiences.

The simulated world contains believable agents, autonomous characters exhibiting rich personalities, emotion, social behavior, motivations and goals.

The user interacts with this world through the presentation. This presentation may perform active dramatic filtering – effecting camera angles and point of view in graphical worlds, or changing the style of language used in textual worlds.

The drama manager can see everything happening in the world. It tries to guide the experience of the user in order to make a story happen. This may involve such actions as changing the physical world model, inducing characters to pursue a course of action, adding or deleting characters, etc.

Believable Agents

Believable agents are the union of AI-based autonomous agents and the personality-rich, emotive characters that appear in the dramatic arts (e.g. theater, cinema). "Believability" is a term used by character artists to describe a property of compelling characters, namely, that of engaging in internally consistent, lifelike and readable behavior in such a manner as to support an audience in suspending disbelief and entering the internal world of the character. This is *not* the same as realism. Characters are not simulations of people, exhibiting behavior indistinguishable from humans under some controlled laboratory condition. Rather, characters are *artistic abstractions* of people, whose behavior, motivations, and internal life have been simplified and exaggerated in just such a way as to engage the audience in the artist's vision. One of the important contributions of the Oz Project has been to identify believability as a first-class AI research goal. After examining the writings of several character artists including [Thomas & Johnston 1981; Jones 1989; Egri 1946], the Oz group defined a set of requirements for believability including the following:

- Personality – Rich personality should infuse everything that a character does, from they way they talk and move to the way they think. What makes characters interesting are their unique ways of doing things. Personality is about the *unique* and *specific*, not the *general*.
- Emotion – Characters exhibit their own emotions and respond to the emotions of others in personality-specific ways.
- Self-motivation – Characters don't just react to the activity of others. They have their own internal drives and desires, which they pursue whether or not others are interacting with them.
- Change – Characters grow and change with time, in a manner consistent with their personality.
- Social relationships – Characters engage in detailed interactions with others in a manner consistent with their relationship. In turn, these relationships change as a result of the interaction.
- Illusion of life – This is a collection of requirements such as: pursuing multiple, simultaneous goals and actions, having broad capabilities (e.g. movement, perception, memory, language), and reacting quickly to stimuli in the environment. Traditional character artists do not mention these requirements explicitly, because they often get them for free (from a human actor, or as a deep assumption in animation). But builders of interactive characters must concern themselves explicitly with building agent architectures that support these requirements.

[Loyall 1997:15-27] provides a more detailed analysis of the requirements for believability.

Bryan Loyall [Loyall 1997; Loyall & Bates 1991] developed both the text world interpreter and graphics world compiler for Hap, a reactive planning language designed for authoring believable agents. ABL, the believable agent language introduced in this dissertation, is based on Hap. Hap was used to build all of the Oz Project characters, including the animated characters the *Woggles* [Loyall & Bates 1993]. He also developed a feature-based natural language generation framework within Hap, which was used in an internal experimental version of the *Woggle* world [Loyall & Bates 1997]. In this framework, natural language processing is integrated with goal and emotion processing throughout the character, supporting rich interactions between language processing and other simultaneously pursued goals.

Scott Neal Reilly [Neal Reilly 1996; Bates, Loyall & Reilly 1992a; Bates, Loyall & Reilly 1992b] developed Em, an emotion system integrated with Hap's goal processing and based on the OCC cognitive appraisal model of emotion [Ortony, Clore & Collins 1988]. He also developed a methodology for authoring personality rich social behaviors within Hap. His text-worlds, *Robbery World* (in which the player confronts an armed gunman during a convenience store robbery), *Office Politics* (in which the player is involved in backstabbing office politics), and *The Playground* (in which the player is a kid trading baseball cards with Melvin the nerd and Sluggo the bully) demonstrated characters with rich emotional responses and social relationships.

Phoebe Sengers [Sengers 1998a; Sengers 1999b] performed interdisciplinary research in both cultural theory and AI. Her cultural-theoretic critique of agent architectures revealed the ways in which standard AI approaches to autonomous agents result in agents that engage in fragmented behavior that is difficult for human observers to understand. This analysis motivated her design of the *Expressivator*, an agent architecture based on Hap, which employs the idea of transition behaviors to help an audience achieve a coherent interpretation of agent behavior. The *Expressivator* was used to build *The Junkyard*, a graphical story world in which a desk lamp (ala Pixar) has been thrown away in a psychiatric institution/junkyard. The work presented in this dissertation shares Phoebe's deep concern with critical technical practice, interdisciplinary work that tightly intertwines humanistic and technical research.

Drama Management

Stories, particularly dramatic stories, have a strong temporal structure, often depicted as in Figure 1-4.

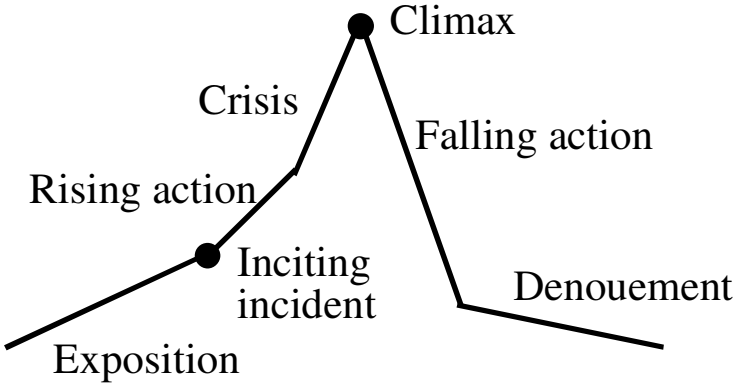

Figure 1-4. Dramatic arc

The vertical axis represents tension, unresolved issues and questions. The horizontal axis represents time. At the beginning, during the exposition, the tension rises slowly as the audience learns the background of the story. An inciting incident then sparks the story, leading to a rapid rise in tension, eventually building to a climax. During the climax, questions are answered and tensions resolved. After the climax, the tension falls rapidly as the world depicted in the story returns to a new status quo. The experience of a story thus has a global shape; events don't happen in a willy-nilly fashion.

Interaction, on the other hand, is generally construed as the freedom to do anything at anytime. Where story is predestination, interaction is freedom; story and interaction seem

fundamentally opposed. Drama management is concerned with resolving this apparent opposition by building systems that actively structure an ongoing interactive experience as a story.

Peter Weyhrauch [Weyhrauch 1997] developed *Moe*, a drama manager that uses adversary-search to select interventions in the interactive drama. *Moe* chooses interventions by projecting all possible abstract story futures, rating the "goodness" of each total story history (a total history is the actual story-so-far plus a projected story future), and selecting the intervention that maximizes the expected total story "goodness". An evaluation function and story moves were defined for a specific text-based story world called *Tea for Three* (a simplified version of the Infocom interactive fiction *Deadline*), though the drama manager was never connected to the concrete world (since *Moe* performs its search in an abstract story space, it was possible to explore *Moe's* performance without hooking it up to a concrete world). While *Façade* does not use adversary-search for drama management, it shares the motivations and many of the concerns explored in Peter's work. More detail on *Moe*, and the relationship between Moe and the *Façade* drama manager, can be found in Chapter 8.

Presentation

The presentation system is the interface between the dramatic world and the player. In some cases the presentation system may be almost entirely absent, providing the user interface for interacting in the world, but performing no active role itself in transforming or filtering the presented activity in the world. The Oz project text worlds employed an active presentation system. In the text world architecture, sensory information is explicitly represented as sense packets (containing formal ontological descriptions of events) propagating throughout the world. When these sensory representations impinge on the player, Mark Kantrowitz's natural language generation system Glinda [Kantrowitz & Bates 1992] generates descriptive text. As an integrated, deep generator (meaning that Glinda, within a single framework, performs both text planning and realization down to the lexical choice level), Glinda is capable of interesting pragmatic variations in the service of dramatic effects. Within this framework, Kantrowitz explored pragmatic variation for natural referring expressions. My own work on subjective avatars in the text world architecture [Mateas 1997; Mateas 1998] (see also Appendix A) explored further presentation effects in text production. [Smith & Bates 1989] is a nice thought piece exploring a range of presentation-level narrative effects.

The idea of an active presentation layer is of course not limited to text worlds. One can imagine automating a variety of cinematic and rendering effects, employing them as a function of the history and current activity within the dramatic world. Indeed, the automatic camera control found in some video games can be considered a version of this. *Façade*, however, employs no active presentation system.

Oz Research Philosophy

The Oz Project's interest in story and character, and particularly the adoption of believability as a first class research goal, lead them to develop a unique AI research philosophy. This philosophy includes a concern for the specific vs. the general, an interest in architectures and approaches that directly support artistic expression, and a neutrality towards the ideological battles structuring traditional AI research.

The desire to pursue the specific rather than the general is strongly connected with the desire to support the direct artistic creation of characters. In traditional media, such as

writing, painting, or animation, artists exhibit fine control over their creations. Starting with a rough idea or vision in her head, the artist uses this fine control to further explore and represent her vision in her chosen medium. Similarly, Oz wanted to support the same level of artistic control in the creation of believable agents. This focus on specificity and artistic control provides an interesting contrast with how both top-down and bottom-up AI would approach the problem of building a character.

Top-down architectures tend to explicitly encode, often in a human-readable form, high level features of a system. Suppose we want to build James Bond using this mindset. First, we would think about characters in the abstract. What general theory captures the notion of character? How might this general theory by parameterized (perhaps through infusions of "knowledge") to select specific characters? To inform the work, we might look at the dimensions of personality as described by various personality models in psychology (e.g. introvert-extrovert, thinking-feeling, intuitive-sensing, judging-perceiving). Once a generic architecture has been built, we could then define different characters by setting the right personality knobs. The problem is, how many of these personality knobs are needed to "tune in" a large number of characters? How many personality knobs need to be turned, and how many degrees of freedom does each knob need, in order to allow the creation of Bugs Bunny, Hamlet, The Terminator, Bambi? The differences between these characters seem to far outweigh any similarities. Or to put it another way, is Bugs Bunnyness captured in a few symbols, which can be read off inside the mind, or is his way-of-being spread throughout his mind and body?

Bottom-up architectures, in contrast, include no such concern with high-level features. A major methodological assumption of such architectures is that high-level features (such as introvertedness) must emerge from simple, low-level mechanisms. This sounds like a promising approach for avoiding the "knobby" problem. However, such bottom-up approaches tend to replace a small number of high-level, descriptive knobs, with a huge number of low-level, real-valued knobs. For example, imagine building a specific character, say James Bond, using a sub-symbolic architecture such as a neural network. A large number of numeric parameters, corresponding to the connection weights between the neurodes of the network, must be set such that the interactive character responds in a Bond-y way in a variety of situations. Given that there is a huge number of such parameters, setting them by hand is hopeless – one alternative is to employ a learning algorithm to automatically set the parameters. But in order to do this, we need a detailed behavioral model of James Bond in order to generate examples for our training set. Yet creating a detailed behavioral model is exactly what we're trying to do in the first place! The difficulty here is that the semantic distance between individual connection weights and the resulting behavior ("emergence"), prevents us from understanding the relationship between high-level behavior and low-level mechanism. Yet that understanding is necessary in order for us to behaviorally specify our character.

The Oz approach to character (and to drama management as well) focused on computational frameworks and authoring methodologies that provide artists a high level of control in the authoring of *specific* characters and stories.

The architectural commitments of bottom-up and top-down AI are often accompanied by ideological commitments about the ultimate nature of intelligence. The Oz approach remained neutral in many of these ideological battles, structuring research around concerns with believability, specificity, and authorship, rather than arguments about the essence of intelligence.

Expressive AI is influenced by and further develops the Oz research philosophy. It can be seen as a generalization of these ideas beyond interactive drama, a further

development of these ideas in the contexts of new media art practice and cultural critiques of science (particularly AI).

Additional Systems

Three additional AI-based artworks, created during my graduate work, are described in Appendices A – C. In discussing Expressive AI, these systems are used as additional examples throughout the dissertation. Here I give brief descriptions of the three systems. While reading the rest of the dissertation, the reader is advised to refer to the more detailed descriptions in the appendices as interest and the need for clarification demand.

Subjective Avatars

Subjective avatars, employed in story worlds, have autonomous emotional reactions to events in the world and keep track of story context. The emotional state and story context are used to provide subjective descriptions of sensory information. The purpose of such description is to help a user gain a deeper understanding of the role they are playing in the story world.

A specific subjective avatar was built for the text world *Fastfood World*, a negative job scenario set in a fastfood restaurant. In this world, the player is a young worker in his early 20's stuck in a dead-end fastfood job. The worker's nemesis is the manager, Barry, who uses every opportunity to dominate the worker.

The avatar is implemented as a Hap agent, autonomously processing goals, generating emotions, and tracking story context. But the avatar never takes direct action in the world; rather events in the world are described subjectively as a function of the current story context and emotional state of the avatar. The avatar becomes an *inverse* user model; the avatar attempts to make the player experience the world in a manner consistent with the avatar's subjective interpretation of the world.

Office Plant #1

Walking into a typical, high tech office environment one is likely to see, among the snaking network wires, glowing monitors, and clicking keyboards, a plant. What a sad creature it is. Domesticated yet ill adapted to its artificial niche of human design, this generic plant sits on a desk corner under artificial illumination, serving as a placeholder for that which electronic machinery can not offer: personal attachment. Office plants are an expression of a need for undemanding presence in an efficiently impersonal environment.

Office Plant #1 (*OP#1*) is an exploration of a technological artifact, adapted to the office ecology, that fills the same social and emotional niche as an office plant. *OP#1*, employing text classification techniques, monitors its owner's email activity. Its robotic body, reminiscent of a plant in form, responds in slow, rhythmic movements to comment on the monitored activity. In addition, it makes its presence and present knowledge known through low, quiet, ambient sound. *OP#1*, observing it's user through the narrow window of email activity, acts as a companion and commentator on these activities.

OP#1 assigns a set of labels to every incoming email using a collection of naïve bayes and k-nearest-neighbor text classifiers. Example recognized email classes include intimate, chatty, fyi, request, and apology. Rules then map boolean combinations of recognized classes into activation energy added to nodes within a fuzzy cognitive map

(similar to a recurrent neural net). Each node of the map corresponds to a plant behavior (e.g. bloom, rest). The nodes influence each other via inhibitory and excitory links; at any given time the plant's behavior is controlled by the node with the highest activation energy.

Figure 1-5. Close-up of *Office Plant #1*

OP#1 is a collaboration with roboticist and artist Marc Boehlen.

Terminal Time

Terminal Time is a machine that constructs ideologically-biased documentary histories in response to audience feedback. It is a cinematic experience, designed for projection on a large screen in a movie theater setting. At the beginning of the show, and at several points during the show, the audience responds to multiple choice questions reminiscent of marketing polls. Their answers to these questions determine an ideological bias; *Terminal Time* generates historical narratives that attempt to mirror and often exaggerate this bias. The engine uses the past 1,000 years of world history as "fuel" for creating these custom-made historical documentaries. By creating histories that clearly and instantly respond to changes in audience make-up, the project is intended to raise fundamental questions about the relationship of points of view to constructions of history, particularly at the dawn of a new Millennium.

Terminal Time's architecture consists of the following major components: knowledge base, ideological goal trees, rule-based natural language generator, rhetorical devices, and a database of indexed audio/visual elements primarily consisting of short digital movies and sound files containing music. The knowledge base contains representations of historical events. This is the raw material out of which the ideologically-biased histories are constructed. Examples of historical events are the First Crusades, the invention of Bakelite, and the rise of enlightenment philosophy. Ideological goal trees represent the current ideological bias being pursued by the narrator. The goal trees consist of rhetorical goals ordered by subgoal and importance (to the ideologue) relationships. These goals are used both to select historical events to include in the story and to "spin" the event in an ideologically-consistent manner. The rule-based natural language generator (NLG) generates the narrative text once specific facts have been selected and connected to make a story. The storyboard serves as a working memory for processes that impose a narrative order on event spins created by the goal tree. Rhetorical devices are connecting pieces of

text with accompanying constraints on story structure. These devices are used to create narrative connections between historical events. Finally, the multimedia database contains the audio/visual elements for the assembled documentary. Once a narrative track has been constructed, information retrieval techniques are used to match the "best" indexed multimedia elements to the appropriate pieces of text. Once the multimedia elements have been selected, the resulting documentary is displayed, layering text-to-speech synthesis of the narrative track, with the video and audio elements.

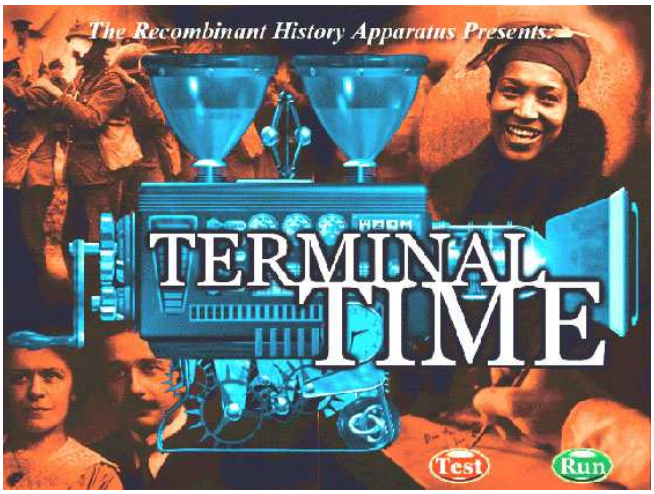

Figure 1-6. Title screen of *Terminal Time*

Terminal Time is a collaboration with documentary filmmaker Steffi Domike and new media artist Paul Vanouse.

Façade Collaboration

The construction of *Façade* is a collaboration with Andrew Stern. Andrew has 8 years of experience in the game industry, most notably for his work on the conceptual and architectural design and implementation of the virtual pets products *Dogz*, *Catz*, and *Babyz* [Stern, Frank & Resner 1998; Stern 2002]. Andrew and I are both intimately involved in the development of the *Façade* concept, including the story and player experience. We are both involved in the high-level design of the architecture. We both authored (and continue to author) the story within the architecture, including the authoring of beats, behaviors, and dialog, and the development of authorial idioms. In addition to these shared efforts, we each have particular areas of focus. I am primarily responsible for the detailed (i.e. code-level) design and implementation of the AI architecture. In addition, I bring general knowledge of a range of AI techniques and architectures, and experience building AI-based interactive art. Andrew is primarily responsible for the detailed (i.e. code-level) design and implementation of the non-photorealistic real-time rendering engine and the user interface. In addition, Andrew brings a wealth of knowledge and experience in the authoring of autonomous characters and the design of successful interactive experiences.

Building something as ambitious as *Façade* requires multiple people, ideally several more than the two of us. During the project, Andrew and I egged each other on to take on ever more complex conceptual and technical issues. Any contributions that *Façade* makes to the field of interactive drama are the fruits of this collaboration.

Additional contributors include Mehmet Fidanboylu, who, as a senior thesis student, built the first implementation of the template compiler, J.P. Lavin, who consulted on the story, and John Rines, who made some of the character animation.

Dissertation Outline

This dissertation speaks in several voices. Chapters 2, 3, and parts of chapter 8 are written in the voice of a designer interested in interactive drama – they discuss the high-level design goals of interactive drama, and how these goals play out in system design. Chapters 5, 6, parts of chapter 8, and chapter 10 are written in the voice of a computer scientist – they discuss the detailed design and implementation issues of various components of the *Façade* architecture. Chapters 4 and 9 are intended to be accessible to both artists and computer scientists who are interested in AI-based art, though the style of these chapters will be somewhat more familiar to artists than computer scientists. And finally, Chapter 7 is written in the voice of the critical theorist – the structuralist semiotic analysis of interpretive and authorial affordance will most likely be enjoyed by new media theorists and theoretically inclined artists. Perhaps, in some ideal world, this entire document would have been written in a single voice, with the various perspectives translated into this one voice. However, in my own work, I tend to move between these several ways of thinking and speaking, allowing them to intertwine and inform each other. Part of my argument for Expresive AI as a distinct art and research agenda is precisely that it involves this deep mixing of modes and approaches, demanding a scientific and technical, artistic, and (critical) theoretical facility from practitioners. The reader is encouraged to use the chapter descriptions here to find her own path through this document.

Chapter 2, Interaction and Narrative, situates first-person interactive drama with respect to other approaches to interactive narrative. The chapter opens with a brief tour of the landscape of interactive narrative, including commercial games, emergent narratives, narrative-based new media art, electronic literature, and interactive drama. Then a neo-Aristotelian theory of interactive drama is presented, further clarifying *Façade's* approach to interactive story vis-à-vis the broader landscape of interactive narrative. Two critiques of interactive drama, specifically the ludological claim that narrative structures are fundamentally at odds with interactive experiences, and the narrativist claim that game-like interaction is fundamentally at odds with interactive narrative, are addressed. The chapter concludes with a description of the design goals for *Façade*.

Chapter 3, The *Façade* Architecture, provides an overview of the *Façade* interactive drama architecture. This architecture is one concrete instantiation of the general Oz project architecture. A primary contribution of the architecture is the specification of the mechanism for communication between the drama manager and the characters; rather than strongly autonomous characters receiving guidance via high-level, low-bandwidth drama manager requests, the drama manager frequently changes the high-level behavioral repertoires of weakly autonomous characters. Each of these changing collections of behaviors specify the joint activity necessary for the characters to carry out a dramatic beat. The architecture has three major components: the drama manager, which consists of both the language for specifying beats and the online system that sequences beats over time, an agent language (ABL) for authoring autonomous believable agents, and a discourse management system, which recognizes the pragmatic intent of natural language input, and selects one or more character reactions to the recognized discourse act as a

function of the current discourse context(s). The individual pieces of the architecture are described in more detail in subsequent chapters.

Chapter 4, AI and Art, examines the differences in the high-level goals of AI research and art practice. Understanding these differences is important for developing a hybrid practice that combines AI research and art making. Additionally, this chapter situates Expressive AI relative to the debate between classical and interactionist AI, arguing that art practice should not align itself with any particular technical agenda. In fact, the classical/interactionist split is just one of the many distinctions, disputes and historical currents structuring the AI landscape. The AI-based artist should potentially use *all* of AI as a conceptual resource, not limiting themselves to one specific subfield or line of argument. The chapter concludes with an examination of the differences in the metaphors structuring AI research and art making – these metaphors will serve as a starting point for an understanding of how Expressive AI combines these two metaphors.

Chapter 5, *A B*ehavior *L*anguage, describes the details of ABL, a new behavior language based on the Oz Project language Hap. The chapter begins with a brief review of Hap, whose semantics ABL reimplements. The bulk of the chapter describes ABL's primary contribution, support for joint goals and behaviors, a mechanism supporting the authoring of tightly coordinated action among multiple believable agents. The joint goal mechanism allows teams of ABL agents to variably couple their individual goal and behavior structures so as to propagate the effects of goal processing (e.g. goal success, failure and suspension) both within an agent (normal goal semantics inherited from Hap) and *across* the members of the team. Additional ABL contributions are also described, including support for asynchronous sensory-motor systems, meta-ABL (support for authoring meta-behaviors that reflect on and manipulate the runtime state of behaviors), working memory extensions, step posting and atomic behaviors. The chapter concludes with a discussion of related and future work.

Chapter 6, ABL Idioms in *Façade*, describes how ABL is used to write the behaviors for *Façade*. These idioms are important for understanding how ABL's features can be used to structure a real-time interactive drama. The idioms discussed include:

- behaviors for body resource management, and a discussion of why the body resource mechanisms ABL borrows from Hap (action conflicts) don't provide enough structure on their own,
- beat goals and handlers, which provide the structure for beat-specific behavior,
- cross-beat behaviors, which continue across beat goals and beat boundaries and mix with the beat-specific behaviors,
- performance utility behaviors, which provide structured mechanisms for the detailed performance of actions and lines of dialog.

Chapter 7, Expressive AI: Affordances and Semiotics, discusses the ideas of interpretive and authorial affordances, and how they can be used to structure an AI-based art and research practice. Interpretive affordances support the interpretations an audience makes about the operations of an AI system. They provide resources both for narrating the operation of the system, and additionally, in the case of an *interactive* system, for supporting intentions for action. The authorial affordances of an AI architecture are the "hooks" that an architecture provides for an artist to inscribe her authorial intention in the machine. Different AI architectures provide different relationships between authorial control and the combinatorial possibilities offered by computation. Expressive AI practice combines these two concerns into a dialectically related whole; the concerns mutually inform each other. The architecture is crafted in such a way as to enable just those authorial affordances that allow the artist to manipulate the interpretive affordances

dictated by the concept of the piece. At the same time, the architectural explorations suggest new ways to manipulate the interpretive affordances, thus suggesting new conceptual opportunities. A semiotic[2] analysis brings further insight into the relationship between interpretive and authorial affordance, providing architectural criteria for good affordances.

Chapter 8, The *Façade* Drama Manager, begins with a general discussion of drama management, characterizing a drama manager as a policy for sequencing story "pieces" in response to the interaction state (including the interaction history). The goal is to create an experience that has both local agency, meaning that the player's local actions immediately generate a response consonant with those actions, and global agency, meaning that the player's actions "add up" over time in such a way as to have a real effect on the global (long term, temporal) structure of the story. The details of *Façade's* drama manager, the beat manager, are then discussed. The beat manager consists of both a beat description language for locally describing the sequencing conditions on individual beats, and a runtime system for using these beat descriptions to dynamically sequence beats. A brief description of *Façade's* story design then sets the stage for a discussion of beat description idioms, specific ways of using the beat description language to specify a beat sequencing policy. The chapter concludes with a description of related and future work.

Chapter 9, Expressive AI: Disciplinary Issues, discusses the art and research goals of Expressive AI relative to both New Media practice and traditional AI research. The chapter opens with a discussion of why AI is useful in cultural production (art making), starting with the premise that for Expressive AI, the fundamental property of the computer as a representational medium is not image production, nor interaction, nor control of electro-mechanical systems, nor networked communications, but rather computation, that is, processes of mechanical manipulation to which observers can ascribe meaning. The chapter continues with a discussion of the major modes or genres of AI-based art – this list is not intended to be exhaustive, but rather to suggest the possibilities of AI-based art production. The chapter concludes with a discussion fo Expressive AI as a discipline, positioning Expressive AI relative to a number of critiques of AI, and describing it as a critical technical practice.

Chapter 10,
Natural Language Processing in *Façade* , describes *Façade's* natural language processing (NLP) system. This system processes natural language input in two phases: phase I, which maps surface text to discourse act(s), and phase II, which maps discourse act(s) to one or more character reactions. The recognition of discourse acts from surface text is accomplished by rules written in a custom template language. The template language provides a set of constructs for detecting patterns in surface text. Rules map "islands" of patterns in the surface text into intermediate meanings, which are then chained together to produce the final discourse act(s). The discourse acts capture the pragmatic meaning of the surface text. The template language provides support for automatically matching word synonyms (using WordNet [Fellbaum 1998] to perform term expansion), for matching stemmed forms of words, and for matching hierarchical (and recursive) template structures. Phase II provides a framework for managing discourse contexts and selecting a reaction to a discourse act given the active contexts. At any point in time there may be several active discourse contexts, with each context proposing multiple potential

[2] Semoitics is a field in the humanities that studies how signs have meaning, that is, how it is that something (e.g. a mark on a page, an article of clothing, a course in a meal), can have meaning for somebody.

reactions to a given discourse act. Customizable selection logic then picks one or more reactions from among the multiple proposed reactions in the multiple active contexts. The chapter concludes with a description of related and future work.

Chapter 11, Conclusion, concludes with a brief description of the major contributions and findings of *Façade* and the *Façade* architecture, and a description of the major theoretical and practical positions of Expressive AI, my art and research practice.

CHAPTER 2
INTERACTION AND NARRATIVE

Many game designers, writers and theorists have wrestled with the vexing question: "what is an interactive narrative?". This chapter situates *Façade's* approach to this question relative to the current interactive narrative landscape. Portions of this chapter first appeared in [Mateas 2000b; Mateas 2001b; Mateas 2003a].

Approaches

A number of approaches are currently being pursued in the theorizing and building of interactive narratives. Each of these approaches foregrounds a different aspect of the problem, focusing on a different point within the design space of interactive narrative.

Before continuing, a note about terminology. When speaking generally about interactive story, I will sometimes use the word *story* and sometimes the word *narrative*. I use *story* when talking about experiences that have a tightly organized plot arc, progression towards a climax, beginning, middle and end, etc., that is, experiences such as "mainstream" novels and movies, which are understood as "stories" by the general population. I use *narrative* when talking about the abstract properties or qualities of stories, and more loosely structured, "experimental", story-like experiences.

Commercial Computer Games

The relationship between narrative and game is a hot topic within the computer game design community. The contemporary gaming scene, perhaps driven by the ever increasing capabilities of computer graphics, and the resulting inexorable drive towards real-time photo-realism, is dominated by mimetic representations of physical scenes, objects and characters. With mimetic representation approaching the richness of animated movies, and with the increasing use of cinematic techniques, such as virtual cameras implementing automated shot vocabularies, comes the desire to provide a narrative explaining who these characters are and why they are in the situation they're in. Contrast this with classic arcade games such as *Pac Man* or *Tempest*, in which the more iconic mode of representation led to games where the proto-narrative was completely dominated by gameplay, and in fact could be safely ignored.

But with this increased interest in narrative, game designers also experience a deep ambivalence. The ephemeral quality of gameplay, the experience of manipulating elements within a responsive, rule-driven world, is still the raison d'être of games, perhaps the primary phenomenological feature that uniquely identifies the computer game as a medium. Where gameplay is all about interactivity, narrative is all about predestination. There is a pervasive feeling in the game design community that narrative and interactivity are antithetical:

> I won't go so far as to say that interactivity and storytelling are mutually exclusive, but I do believe that they exist in an inverse relationship to one another… Interactivity is almost the opposite of narrative; narrative flows under the direction

> of the author, while interactivity depends on the player for motive power… [Adams 1999a]

This tension is reflected in the decline of the most story-based game genre, the commercial adventure game. Text adventures were a highly successful form in the 1980's, giving way to the graphic adventures of the early and mid 1990's. And through the mid 1990's, with the release of critically acclaimed titles such as *Myst* and *Grim Fandango*, the adventure game remained a vibrant form. But by the late 1990's the form was in trouble, with reviewers and critics pronouncing the death of the adventure game [Adams 1999b; OMM 2001]. But while early declarations of the death of the adventure game sometimes ended with hope (e.g. "Adventure games appeal to a market which is unimpressed by the size of the explosions or the speed of the engine, a market that for the most part, we're ignoring. But those people want to play games too. It's time to bring adventure games back." [Adams 1999b]), the decline continues to this day, with a recent review in the New York Times declaring "So far, 2002 has been the worst year for adventure games since the invention of the computer."[Herold 2002]. While adventure elements continue to live on in action adventures such as *Luigi's Mansion*, the *Resident Evil* franchise, and the *Tomb Raider* franchise, action adventures emphasize physical dexterity (e.g. shooting, running, jumping) over puzzle solving and plot progression.

In contemporary game design, narrative elements are primarily employed to provide a an explanatory background against which the high-resolution mimetic action of the game takes place. Thus characters and situations may make reference to well known linear narratives (e.g. *Star Wars*), or nuggets of backstory may be revealed as the game progresses, or the game action may occur within an inexorably progressing narrative. But strongly authored stories whose path and outcome depend on player interaction are not currently an active line of exploration in commercial game design.

Emergent and Player Constructed Narrative

Rather than viewing narratives as highly structured experiences created by an author for consumption by an audience, emergent narrative is concerned with providing a rich framework within which individual players can construct their own narratives, or groups of players can engage in the shared social construction of narratives. Autonomous characters may be designed in such a way that interactions among autonomous characters and between characters and the player may give rise to loose narratives or narrative snippets [Stern 2002; Stern 1999; Aylett 1999]. Multi-user online worlds, including text-based Multi-User Dungeons (MUDs), avatar spaces, and massively multiplayer games such as *Everquest* and *Ultima Online*, create social spaces in which groups co-construct ongoing narratives. And simulation environments such as *The Sims* may be used by players to construct their own stories. Using the ability to capture screen shots and organize them into photo albums, plus the ability to construct new graphical objects and add them to the game, players of *The Sims* are constructing and posting online thousands of photo album stories.

Narrative and New Media Art

In fine art practice, narrative is understood as one, rather powerful, form of representation. Much of contemporary art practice involves self-consciously questioning representational modes, exploring the boundaries, breaking the representation, questioning whose power is being preserved by a representational mode, and hybridizing modes in order to create new ones. Thus, when engaging in narratively-based work,

artists rarely tell straightforward narratives employing the standard narrative tropes available within their culture, but rather ironize, layer, and otherwise subvert the standard tropes from a position of extreme cultural self-consciousness. For example, *Terminal Time* constructs ideologically-biased documentary histories based on audience responses to psychographic profiles. The narrative structure of the traditional documentary form is made visible through endless replication [Domike, Mateas & Vanouse 2002, Mateas, Vanouse & Domike 2000] (see Appendix C). *The Dr. K— Project* creates a narrative landscape that, rather than having a mimetic, independent existence, is created in response to audience interaction [Rickman 2002]. In these and similar works, interaction is used to open the narrative, to make its internal structure visible.

A highly active area in new media interactive narrative is net art. Such work, while employing multi-media elements such as sound, still and moving imagery as in Mark Amerika's *Grammatron*, or making use of interaction tropes from classic video games as in Natalie Bookchin's *Intruder*, often makes heavy use of textual presentation and literary effects, and thus is also a form of electronic literature.

Electronic Literature

Electronic literature is concerned with various forms of interactive reading, that is, interactive literary textual narratives. While there is certainly much exploration in this area combining multi-media elements, kinetic text, and novel interfaces, the canonical forms of electronic literature are hypertext and interactive fiction.

A hypertext consists of a number of interlinked textual nodes, or lexia. The reader navigates these nodes, selecting her own path through the space of lexia, by following links. Links may be dynamic, appearing and disappearing as a function of the interaction history, the contents of nodes may dynamically change, and navigation may make use of spatial mechanisms and metaphors rather than relying purely on link following [Rosenberg 1998]. However, a static node and link structure is the skeleton upon which such effects are added; many hypertext works consist solely of static node and link structures. The production of hypertext literature is intimately connected with the production of hypertext theory. Early theorists saw hypertext as the literal embodiment of postmodernist theories of deferred and intertextual signification [Landow 1992]. Like new media artists, hypertext authors tends to engage in theoretical explorations of the limits of narrative. Interactivity is seen as enabling rhizomatic stories that avoid the authorial imposition of a preferred viewpoint. Every story event can be viewed from multiple points of view, with closure indefinitely deferred.

Interactive fiction is a generalized term for "text adventure", the form inaugurated with the 1976 creation of *Adventure*, a textual simulation of a magical underground world in which the player solves puzzles and searches for treasure. *Adventure*, and all later interactive fictions, makes use of a conversational interface in which the player and the computer exchange text; the player types commands she wishes to perform in the world and the computer responds with descriptions of the world and the results of commands. While text adventures have not been commercially viable since the early 90's, there remains a very active non-commercial interactive fiction scene producing many literary interactive fictions, holding a number of yearly competitions, and actively theorizing the interpretation and production of interactive fiction [Montfort 2003].

Interactive Drama

Interactive drama per se was first conceived in Laurel's 1986 dissertation [Laurel 1986], an extended thought experiment involving dramatic stories in which the player enters as a first-person protagonist. While based most closely on the genres of the text and graphic adventure, interactive drama distinguishes itself from these and other conceptions of interactive narrative in a number of ways.

- Interactive drama takes *drama*, rather than literature, fine art, or game interaction tropes, as the guiding narrative conception. With this focus on drama comes a concern with intensity, enactment and unity.
- Interactive drama wants player interaction to deeply shape the path and outcome of the story, while maintaining a tight, author given story structure. Thus interactive drama confronts head-on the tension between interactive freedom and story structure.
- Interactive drama seeks first-person immersion as a character *within* the story.

Façade continues in the tradition of interactive drama.

A Neo-Aristotelian Theory of Interactive Drama

This section describes a neo-Aristotelian theory of interactive drama, continuing a specific thread of discussion first begun by Laurel's adoption of an Aristotelian framework for interactive drama [Laurel 1986], and then more generally for interactive experiences [Laurel 1991], and continued by Murray's description of the experiential pleasures and properties of interactive narratives [Murray 1998]. As an interactive narrative approach, interactive drama foregrounds the tension between interaction and story: how can an interactive experience have the experiential properties of classical, Aristotelian drama (identification, economy, catharsis, closure) while giving the player the interactive freedom to have a real effect on the story? This section provides a theoretical grounding for thinking about this question by developing a theory of interactive drama based on Aristotle's dramatic theory [Aristotle 330BC] but modified to address the interactivity added by player agency. This theory provides both design guidance for maximizing player agency within interactive dramatic experiences (answering the question "What should I build?") and technical direction for the AI work necessary to build the system (answering the question "How should I build it?").

As described above, interactive drama is one approach among many in the space of interactive narrative. The neo-Aristotelian poetics developed here is not intended to be a superiority argument for interactive drama, isolating it as the preferred approach in interactive narrative; rather, this poetics informs a specific niche within the space of interactive narrative and provides a principled way of distinguishing this niche from other interactive narrative experiences.

Defining Interactive Drama

In interactive drama, the player assumes the role of a first person character in a dramatic story. The player does not sit above the story, watching it as in a simulation, but is immersed *in* the story.

Following Laurel, Table 2-1 lists distinctions between dramatic and literary narratives.

Dramatic narratives	*Literary narratives*
Enactment	**Description**
Intensification	**Extensification**
Unity of Action	**Episodic Structure**

Table 2-1. Distinctions between dramatic and literary narratives

Enactment refers to action. Dramas utilize action rather than description to tell a story. Intensification is achieved by arranging incidents so as to intensify emotion and condense time. In contrast, literary forms often "explode" incidents by offering many interpretations of the same incident, examining the incident from multiple perspectives, and expanding time. Unity of action refers to the arrangement of incidents such that they are all causally related to a central action. One central theme organizes all the incidents that occur in the story. Literary narratives tend to employ episodic structure, in which the story consists of a collection of causally unrelated incidents.

Though the model developed in this paper will provide design guidance on how to generate a sense of user agency in any interactive experience, it is primarily designed to illuminate interactive drama, that is, an interactive experience with the properties of dramatic stories.

Though interactive drama is strongly related to interactive fiction, it is interesting to note that a major trope of interactive fiction, the puzzle, is in conflict with the dramatic properties of enactment, intensification, and unity of action. Puzzles disrupt enactment, breaking immersion in the action and forcing reflection on the action as a problem to be solved. As the player thinks about the puzzle, action grinds to a halt. Solving puzzles invariably involves trial-and-error problem solving. All the dead ends involved in solving a puzzle introduce incidents that expand time and reduce emotion, thus disrupting intensification. Each puzzle can be thought of as having a "halo" consisting of all the failed attempts to solve the puzzle. These "halos" are extensive; they expand the experience rather than focus it. Puzzle-based experiences tend to be episodic; individual puzzles are loosely related by virtue of being in the same world, but are not strongly related to a central action. Puzzles have an internal logic that makes them self sufficient and internally consistent, but disrupts unity of action across the entire experience.

This is not to say that puzzles lack any aesthetic value or are a uniformly "bad" idea in interactive experiences. Montfort convincingly argues that puzzles in interactive fiction are related to the literary figure of the riddle, "…inviting the riddlee to awaken to a new vision of the world."[Montfort 2003]. It is only to say that the form of engagement demanded by the puzzle is disruptive of dramatic properties.

Murray's Aesthetic Categories

Murray [Murray 1998] proposes three aesthetic categories for the analysis of interactive story experiences: immersion, agency, and transformation.

Immersion is the feeling of being present in another place and engaged in the action therein. Immersion is related to Coleridge's "willing suspension of disbelief" - when a participant is immersed in an experience, they are willing to accept the internal logic of the experience, even though this logic deviates from the logic of the real world. A species

of immersion is telepresence, the feeling of being physically present (from a first person point of view) in a remote environment.

Agency is the feeling of empowerment that comes from being able to take actions in the world whose effects relate to the player's intention. This is not mere interface activity. If there are many buttons and knobs for the player to twiddle, but all this twiddling has little effect on the experience, there is no agency. Furthermore, the effect must relate to the player intention. If, in manipulating the interface elements, the player does have an effect on the world, but they are not the effects that the player intended (perhaps the player was randomly trying things because they didn't know what to do, or perhaps the player thought that an action would have one effect, but it instead had another), then there is no agency.

Transformation is the most problematic of Murray's three categories. Transformation has at least three distinct meanings.

- Transformation as masquerade. The game experience allows the player to transform themselves into someone else for the duration of the experience.
- Transformation as variety. The game experience offers a multitude of variations on a theme. The player is able to exhaustively explore these variations and thus gain an understanding of the theme.
- Personal transformation. The game experience takes the player on a journey of personal transformation.

Transformation as masquerade and variety can be seen as means to effect personal transformation.

Integrating Agency into Aristotle

Murray's categories are phenomenological categories of the interactive story experience, that is, categories describing what it *feels* like to participate in an interactive story. Aristotle's categories (described below) are structural categories for the analysis of drama, that is, categories describing what *parts* a dramatic story is made out of. The trick in developing a theoretical framework for interactive drama is integrating the phenomenological (that is, what it feels like) aspect of a first person experience with the structural aspect of carefully crafted stories. In attempting this integration, I will first discuss the primacy of the category of agency. Second, I will briefly present an interpretation of the Aristotelian categories in terms of material and formal cause. Finally, agency will be integrated into this model.

Primacy of Agency

From an interactive dramatic perspective, agency is the most fundamental of Murray's three categories. Immersion, in the form of engagement, is already implied in the Aristotelian model. Engagement and identification with the protagonist are necessary in order for an audience to experience catharsis. Transformation, in the form of change in the protagonist, also already exists in the Aristotelian model. Murray's discussion of transformation as variety, particularly in the form of the kaleidoscopic narrative that refuses closure, is contrary to the Aristotelian ideals of unity and intensification. To the extent that we want a model of interactive *drama*, as opposed to interactive narrative, much of Murray's discussion of transformation falls outside the scope of such a model. While immersion and transformation exist in some form in non-interactive drama, the audience's sense of having agency within the story is a genuinely new experience

enabled by interactivity. For these reasons, agency will be the category integrated with Aristotle.

Aristotelian Drama

Following Laurel [Laurel 1991], Aristotle's theory of drama is represented in Figure 2-1.

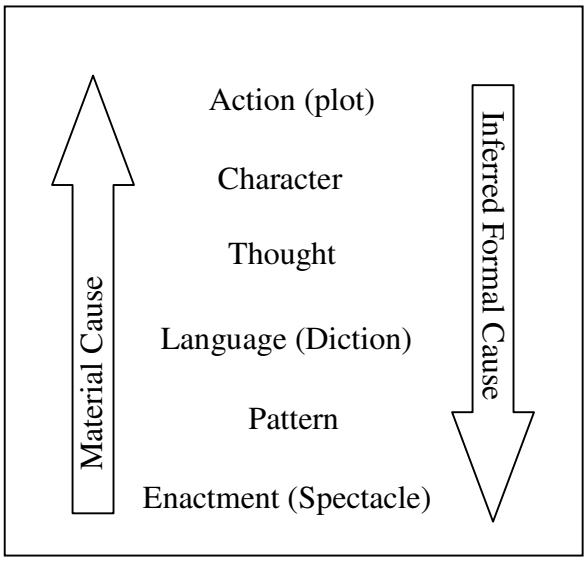

Figure 2-1. Aristotelian theory of drama

Aristotle analyzed plays in terms of six hierarchical categories, corresponding to different "parts" of a play. These categories are related via material cause and formal cause. The material cause of something is the material out of which the thing is created. For example, the material cause of a building is the building materials out of which it is constructed. The formal cause of something is the abstract plan, goal or ideal towards which something is heading. For example, the formal cause of a building is the architectural blueprints.

In drama, the formal cause is the authorial view of the play. The author has constructed a plot that attempts to explicate some theme. The characters required in the play are determined by the plot; the plot is the formal cause of the characters. The characters' thought processes are determined by the kinds of characters they are. The language spoken by the characters is determined by their thought. The patterns (song) present in the play are determined, to a large extent, by the characters' language (more generally, their actions). The spectacle, the sensory display presented to the audience, is determined by the patterns enacted by the characters.

In drama, the material cause is the audience view of the play. The audience experiences a spectacle, a sensory display. In this display, the audience detects patterns. These patterns are understood as character actions (including language). Based on the characters' actions and spoken utterances, the audience infers the characters' thought processes. Based on this understanding of the characters' thought processes, the audience develops an understanding of the characters, the characters' traits and propensities. Based on all this information, the audience understands the plot structure and the theme. In a successful play, the audience is then able to recapitulate the chain of formal causation. When the plot is understood, there should be an "ah-ha" experience in which the audience is now able to understand how the characters relate to the plot (and why they

must be the characters they are), why those type of characters think they way do, why they took the actions they did and said what they did, how their speech and actions created patterns of activity, and how those patterns of activity resulted in the spectacle that the audience saw. By a process of interpretation, the audience works up the chain of material cause in order to recapitulate the chain of formal cause.

Interactive Drama

Adding interaction to the Aristotelian model can be considered the addition of two new causal chains at the level of character as depicted in Figure 2-2. The gray arrows are the traditional chains of material and formal causation. The player has been added to the model as a character who can choose his or her own actions. This has the consequence of introducing two new causal chains. The player's intentions become a new source of formal causation. By taking action in the experience, the player's intentions become the formal cause of activity happening at the levels from language down to spectacle. But this ability to take action is not completely free; it is constrained from below by material resources and from above by authorial formal causation from the level of plot.

The elements present below the level of character provide the player with the material resources (material cause) for taking action. The only actions available are the actions supported by the material resources present in the game. The notion of affordance [Norman 1988] from interface design is useful here. In interface design, affordances are the opportunities for action made available by an object or interface. But affordance is even stronger than implied by the phrase "made available"; in order for an interface to be

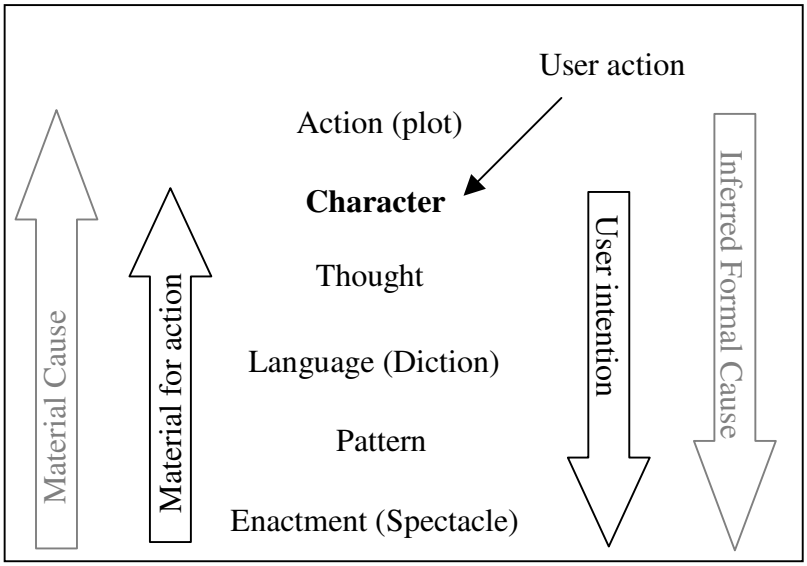

Figure 2-2. Neo-Aristotelian theory of interactive drama

said to afford a certain action, the interface must in some sense "cry out" for the action to be taken. There should be a naturalness to the afforded action that makes it the obvious thing to do. For example, the handle on a teapot affords picking up the teapot with your hand. The handle cries out to be grasped. In a similar manner, the material resources in an interactive drama afford action. Thus these resources not only limit what actions can be taken (the negative form of constraint) but cry out to make certain actions obvious (the positive form of constraint). Several examples of the material affordances in interactive drama are provided below.

The characters in an interactive drama should be rich enough that the player can infer a consistent model of the characters' thought. If the characters' thought can be understood (e.g. goals, motivations, desires), then this thought becomes a material resource for player action. By reasoning about the other characters' thoughts, the player can take actions to influence these characters, either to change their thoughts, or actively help or hinder them in their goals and plans.

The dialog (language) spoken by the characters and the opportunities for the player to engage in dialog are another material resource for action. Dialog is a powerful means for characters to express their thoughts, thus instrumental for helping the player to infer a model of the characters' thoughts. Conversely, dialog is a powerful means to influence character behavior. If the experience makes dialog available to the player (and most contemporary interactive experiences do not), this becomes a powerful resource for expressing player intention.

The objects available in the experience (I place the presence of interactive objects somewhere between spectacle and pattern) are yet another material resource for player action.

Finally, the mechanics of interaction (spectacle) provide the low-level resources for player actions. The mechanics provide the interface conventions for taking action.

In addition to the material affordances (constraints) from below, the player experiences formal constraints from above. Of course, these constraints are not directly perceived by the player, but, just as in non-interactive drama, are understood by recapitulating the author's chain of formal causation by making inferences along the chain of material causation. In non-interactive drama, understanding the formal chain of causation allows the audience to appreciate how all the action of the play stems from the dramatic necessity of the plot and theme. In interactive drama, the understanding of the formal causation from the level of plot to character additionally helps the player to have an understanding of what to do, that is, why they should take action within the story world *at all*. Just as the material constraints can be considered as affording action from the levels of spectacle through thought, the formal constraints afford *motivation* from the level of plot. This motivation is conveyed as dramatic probability. By understanding what actions are dramatically probable, the player understands what actions are worth considering.

Agency

We are now ready to propose a prescriptive, structural model for agency. *A player will experience agency when there is a balance between the material and formal constraints.* When the actions motivated by the formal constraints (affordances) via dramatic probability in the plot are commensurate with the material constraints (affordances) made available from the levels of spectacle, pattern, language and thought, then the player will experience agency. An imbalance results in a decrease in agency. This will be made clearer by considering several examples.

Many puzzle-based adventures suffer from the imbalance of providing more material affordances than formal affordances. This results in the feeling of having many things to do (places to go, objects to fiddle with) without having any sense of why any one action would be preferable to another. For example, *Zork Grand Inquisitor* offers a rich world to navigate and many objects to collect and manipulate. Yet, since there is no unity of action, there is no way to relate current actions to the eventual goal of defeating the Grand Inquisitor. This leaves the player in the position of randomly wandering about trying strange juxtapositions of objects. This detracts from the sense of agency – though

the player can take action, this action is often not tied to a high-level player intention. Notice that adding more material opportunities for action would not help the matter. The problem is not a lack of options of things to do, the problem is having insufficient formal constraint to decide between choices.

First-person shooters such as *Quake* induce agency by providing a nice balance between material and formal constraints. The proto-plot establishes the following formal constraints (dramatic probabilities):

1. Everything that moves will try to kill you.
2. You should try to kill everything.
3. You should try to move through as many levels as possible.

From these three principles, all the rest of the action follows. The material affordances perfectly balance these formal affordances. The player can run swiftly and smoothly through the space. The player can pick up a wide array of lethal weapons. The player can fire these weapons at monsters and produce satisfying, gory deaths. The monsters' behavior is completely consistent with the "kill or be killed" ethos. Everything that one would want to try and do given the formal constraints is doable. There are no extraneous actions available (for example, being able to strike up a conversation with a monster) that are not dictated by the formal constraints.

Note that though these example games are not specifically interactive drama, the model can still be used to analyze player agency within these games. Though the model is motivated by interactive drama, it can be used to analyze the sense of agency in any interactive experience by analyzing the experience in *terms of the dramatic categories* offered by the model. For example, though *Quake* has neither plot nor characters in the strict sense, there are top-down player expectations established by a "proto-plot". This "proto-plot" is communicated by the general design of the spectacle (e.g. the design of the creepy industrial mazes) as well as the actions of the characters, even if these characters do have primitive diction and thought.

In order to invoke a sense of agency, an interactive experience must strike a balance between the material and formal constraints. An experience that successfully invokes a sense of agency inhabits a "sweet spot" in design space. Trying to add additional formal constraints (more plot) or additional material constraints (more actions) to a balanced experience is likely to move it out of the sweet spot.

I would like to conclude this section with a brief clarification of my use of Aristotle's causal terminology (this clarification will appear in [Mateas 2003c]). Laurel notes that my statements "formal cause is the authorial view of the play" and "material cause is the audience view of the play" are, strictly speaking, a misuse of the Aristotelian causal nomenclature [Laurel 2003]. The actual work of authoring is correctly understood as an efficient cause, while Aristotle proposes no causal role for the audience. But what I mean to highlight by these statements is not the author or audience viewed as a cause, but rather what sort of information is directly available to author vs. audience. The author, through the act of authoring (efficient cause), arranges the elements both materially and formally. But while the material arrangement of the elements is more or less available to the audience, the formal arrangement is not. The author knows things about the play, such as why a character must be *this* character for *this* whole action (formal cause), that the audience does not. The audience must work from what is directly available to the senses, and hopefully, by following the chain of material causation, eventually recapitulate the chain of formal causation. So in referring to the "authorial view" and "audience view," I am attempting to highlight this asymmetry in knowledge between author and audience. The chain of formal cause is available to the author in a way that it

is not available to the audience. And the chain of material cause is in some sense designed *for the audience* as it is the ladder they must climb in order to understand the whole action.

Similarly a player in an interactive drama becomes a kind of author, and thus, as an efficient cause, contributes both materially to the plot and formally to elements at the level of character on down. But these contributions are constrained by the material and formal causes (viewed as affordances) provided by the author of the interactive drama. Hopefully, if these constraints are balanced, the *constrained freedom* of the player will be productive of agency. In these discussions, I elided efficient cause and went straight for a discussion of the material and formal causes that the act of authoring puts in place.

Clarification of the Conceptual Experiment

This neo-Aristotelian theory clarifies the conceptual experiment we are undertaking with *Façade*. The goal is to create an interactive dramatic experience with the experiential properties of traditional drama, namely enactment, intensity, catharsis, unity and closure (these experiential properties are not independent; for example, intensity and unity are related to each other as are catharsis and closure). The Aristotelian analytic categories describe the structure (parts and relationships) of a story experience that induces these experiential properties. The way in which interaction has been incorporated into this model clarifies what is meant by *interactive* dramatic experience. Here, interaction means *first-person* interaction as a character within the story. Further, the essential experiential property of interactivity is taken to be agency. The interactive dramatic experience should be structured in such a way as to maximize the player's sense of agency within the story. The model provides prescriptive structural guidance for maximizing agency, namely, to balance material and formal constraints. So the conceptual experiment of *Façade* can now be more precisely stated as follows: build a first-person, interactive dramatic world that, in addition to the classical experiential properties of Aristotelian drama, also provides the player with a strong sense of agency.

Relationship to Immersion and Transformation

Agency was taken as the fundamental Murray category to integrate with Aristotle. In this section, I examine what the new, integrated model has to say about the other two categories, immersion and transformation.

Immersion

Murray suggests three ways of inducing immersion: structuring participation with a mask (an avatar), structuring participation as a visit, and making the interaction conventions (the interface mechanics) seamless. These three mechanisms can be viewed in turn as a way to provide material and formal constraints, as a design suggestion for balancing the constraints, or as a design suggestion for providing effective material constraints at the level of spectacle. Agency is a necessary condition for immersion.

An avatar can provide both material and formal constraints on a player's actions. The avatar can provide character exposition through such traits as physical mannerisms and speech patterns. This character exposition helps the player to recapitulate the formal, plot constraints. Through both input and output filtering (e.g. the characters in *Everquest*, [Mateas 1997]), the avatar can provide material constraints (affordances) for action.

A visit is one metaphor for balancing material and formal constraints when the material opportunities for action are limited. From the formal side, the conventions of a

visit tell the player that they won't be able to do much. Visits are about just looking around, possibly being guided through a space. Given the limited expectations for action communicated by the formal constraints, the game designer can get away with providing limited material means for action (and in fact, must *only* provide limited means).

The mechanics provide the material resources for action at the level of spectacle (the interface can be considered part of the spectacle). Providing a clean, transparent interface insures that agency (and thus immersion) will not be disrupted.

Transformation

Most of Murray's discussion of transformation examines transformation as variety, particularly in the form of kaleidoscopic narratives, which can be reentered multiple times so as to experience different aspects of the story. Agency, however, requires that a plot structure be present to provide formal constraints. An open-ended story without a clear point of view may disrupt the plot structure too much, thus disrupting agency. However, transformation as variety is necessary to make interaction really *matter*. If, every time a player enters the dramatic world, roughly the same story events occur regardless of the actions taken by the player, the player's interaction would seem inconsequential; the player would actually have no real effect on the story.

One way to resolve the apparent conflict between transformation and agency is to note that agency is a first-person experience induced by making moment-by-moment decisions within a balanced (materially and formally) interactive system, while transformation as variety is a third-person experience induced by observing and reflecting on a number of interactive experiences. Imagine an interactive drama system that guides the player through a fixed plot. As the player interacts in the world, the system, through a number of clever and subtle devices, moves the fixed plot forward. Given that these devices are clever and subtle, the player never experiences them as coercive; the player is fully engaged in the story, forming intentions, acting on them, and experiencing agency. Then imagine an observer who watches many players interact with this system. The observer notices that no matter what the players do, the same plot happens (meaning that roughly the same story events occur in the same order, leading to the same climax). By watching many players interact with the system, the observer has begun to discern the devices that *control* the plot *in the face of* player interaction. This observer will conclude that the player has no true agency, that the player is not able to form any intentions that actually matter within the dramatic world. But the first-time player within the world *is* experiencing agency. The designer of the dramatic world could conclude that since they are designing the world for the player, not for the observer, that as long as the player experiences a true sense of interactive freedom, that is, agency, transformation as variety is not an important design consideration.

The problem with this solution to the agency vs. transformation dilemma becomes apparent as the player interacts with the world a *second* time. On subsequent replays of the world, the player and the observer become the same person. The *total* interactive experience consists of both first-person engagement within the dramatic world and third-person reflection across multiple experiences in the world. In order to support the total experience, the system must support both first-person engagement and third-person reflection; must provide agency *and* transformation as variety.

A dramatic world supporting this total experience could provide agency (and the concomitant need to have a plot structure providing formal constraints) *and* transformation by actively structuring the player experience such that each run-through of the story has a clean, unitary plot structure, but multiple run-throughs have different,

unitary plot structures. Small changes in the player's choices early on result in experiencing a different unfolding plot. The trick is to design the experience such that, once the end occurs, any particular run-through has the force of dramatic necessity. The story should have the dramatic probabilities smoothly narrowing to a necessary end. Early choices may result in different necessary ends – later choices can have less effect on changing the whole story, since the set of dramatically probable events has already significantly narrowed. Change in the plot should not be traceable to distinct branch points; the player should not be offered an occasional small number of obvious choices that force the plot in a different direction. Rather, the plot should be smoothly mutable, varying in response to some global state that is itself a function of the many small actions performed by the player throughout the experience. The *Façade* architecture, an overview of which is provided in Chapter 3, and the accompanying authorial idioms for character behavior (Chapter 6) and story sequencing (starting page 163 of Chapter 8), offers one approach for supporting this variety within unity.

Technical Agenda

In addition to clarifying conceptual and design issues in interactive drama, the neo-Aristotelian model informs a technical agenda of AI research necessary to enable this kind of experience.

The primary heuristic offered by the model is that to maintain a sense of player agency in an interactive experience, material and formal constraints must be balanced. As the sophistication of the theme and plot of an experience increases, maintaining this balance will require characters whose motivations and desires are inferable from their actions. In addition, these characters will have to respond to the player's actions. Believable agents, that is, computer controlled characters with rich personality and emotion, will be necessary to provide these characters. In a domestic drama like *Façade*, in which the plot centers around relationships, trust, betrayal, infidelity, and self-deception, language is necessary to communicate the plot. In order to convey the formal constraints provided by the plot, the characters must have a rich repertoire of dialog available. In addition, the player must be able to talk back. One can imagine a system in which the characters can engage in complex dialog but the player can only select actions from menus or click on hotspots on the screen; this is in fact the strategy employed by character-based multimedia artwork and contemporary adventure games. But this strategy diminishes agency precisely by unbalancing material and formal constraints. The characters are able to express complex thoughts through language. However, the player is not able to influence these thoughts except at the coarse level provided by mouse-click interactivity. Since part of the conceptual experiment of *Façade* is to maximize agency in interaction, *Façade* must support player dialog and thus must provide an AI solution for a limited form of natural language dialog.

The function of interactive characters is primarily to communicate material and formal constraints. That is, the player should be able to understand why characters take the actions they do, and how these actions relate to the plot. Sengers [Sengers 1998a] provides a nice analysis of how this focus on agents as communication vs. agents as autonomous, independent entities, results in changes in agent architectures. When the focus changes from "doing the right thing" (action selection) to "doing the thing right" (action expression), the technical research agenda changes [Sengers 1998b]. The neo-Aristotelian model indicates that action expression is exactly what is needed. In addition, an interactive drama system must communicate dramatic probability (likely activity given the plot) while smoothly narrowing the space of dramatic probability over time. This

means that story action must be coordinated in such a way as to communicate these plot level constraints. Thus it is not enough for an individual character's actions to be "readable" by an observer. Multiple characters must be coordinated in such a way that their joint activity communicates both formal and material (plot and character level) affordances. As will be seen in Chapter 3, this focus on communicating affordances changes the standard architectural assumptions regarding the relationship between plot and character.

Critiques of Interactive Drama

Interactive drama, in its Aristotelian conception, currently inhabits a beleaguered theoretical position, caught in the cross-fire between two competing academic formations: the narrativists and the ludologists. The narrativists generally come out of literary theory, take hypertext as the paradigmatic interactive form, and use narrative and literary theory as the foundation upon which to build a theory of interactive media. Ludologists generally come out of game studies (e.g. [Avedon & Sutton-Smith 1971]), take the computer game as the paradigmatic interactive form, and seek to build an autonomous theory of interactivity (read: free of the English department), which, while borrowing from classical games studies, is sensitive to the novel particularities of computer games (this is sometimes described as a battle against the colonizing force of narrative theory [Eskelinen 2001]). Both camps take issue with an Aristotelian conception of interactive drama, finding it theoretically unsophisticated, an impossible combination of game and narrative (though of course the camps disagree on whether this should be decided in favor of game or narrative), and technically impossible. Gonzalo Frasca, an able proponent of ludology, offers three specific objections to the neo-Aristotelian conception of drama in [Frasca 2003], namely: neo-Aristotelian interactive drama creates an impossible-to-resolve battle between the player and the system, confuses first and third-person perspectives, and is technically impossible. My responses to Frasca's comments here will appear in [Mateas 2003b]. Frasca's critique is representative of ludological critiques of neo-Aristotelian interactive drama, with similar critiques appearing in[Aarseth 1997].

A Specific Ludological Critique

Frasca argues that a conception of interactive drama that attempts to create a strong sense of closure with a well-formed dramatic arc introduces a battle for control between the player and system. If the system decides the ending, we have guaranteed closure without interactive freedom; if the user decides the ending we have guaranteed freedom but possibly no closure. Further, if the player is playing a prescribed role, such as Gandhi, we either have to limit interactive freedom to maintain the player's role (and story arc) or provide interactive freedom at the expense of the role (and story arc). Both these arguments have the following form: story means fate, interactivity means freedom (doing whatever you want), therefore interactivity and story can't be combined. However, the whole point of the neo-Aristotelian theory presented in this chapter is to replace the vague and open-ended term *interactivity* with the more specific term *agency*, and to then argue the conditions under which a player will experience agency: a player will *experience agency when material and formal constraints are balanced*. This is not the same as "a player will experience agency when they can take arbitrary action whenever they want". So in the case of choosing the ending of an interactive story, the player does

not need the ability to make arbitrary endings happen in order to feel agency. A small number of authorially-determined ending configurations can still produce a strong feeling of player agency if reached through sequences of player actions within a materially and formally balanced system. Similarly, a Gandhi story can still produce a sense of agency without providing Gandhi with a chain gun or rocket launcher. If an interactive Gandhi story left weapons and power-ups lying about, but used some heavy handed interaction constraint (like the cursor turning red and beeping) to prevent the player from picking them up, then the experience would certainly be offering material affordances ("here's a gun for you to pick up – oops, not really") not balanced by the formal affordances (the dramatic probabilities of the Gandhi story), resulting in a decrease in the feeling of player agency. If, however, the Gandhi world never provided access to such weapons, and given the plot it never made sense to think of using such weapons, the player would still experience agency, even in the absence of access to plasma cannons. Interactive story designers do not have to be saddled with the impossible task of allowing the player to do whatever they want while somehow turning it into a well-formed story; creating a sense of both story and agency (interactivity) requires "merely" the hard task of balancing material and formal constraints.

Note that the neo-Aristotelian theory does not *prove* that if you build a system that materially balances more complex formal affordances, the player will experience both agency and "storyness". But neither do Frasca's arguments *prove* that this combination of agency and "storyness" is impossible. This is an empirical question. But the neo-Aristotelian theory has the advantage of providing a constructive plausibility argument that can inform the technical research agenda required to search for an empirical answer.

Frasca also argues that neo-Aristotelian interactive drama confuses the first-person gaming situation with the third-person narrative situation. A narrative is an already accomplished structure that is told to a spectator. A game is an evolving situation that is being accomplished by an interactor. Since an already accomplished static structure is not the same thing as an evolving, dynamic situation, then, the argument goes, narrative and game are fundamentally dichotomous. What this argument denies, however, is the possibility for hybrid situations, such as the the storytelling situation, in which a storyteller constructs a specific story through interaction with the audience. In this situation, the audience is both spectator and interactor, and the evolving story only becomes an already accomplished structure at the end, yet still has story properties (e.g. interpreted in accord with narrative conventions) in its intermediate pre-completed forms. Aristotelian interactive drama is similar to this storytelling situation; through interaction the player carves a story out of the block of narrative potential provided by the system.

Finally, Frasca argues against neo-Aristotelian interactive drama on the grounds of technical impossibility. It is very difficult for a human author to write a single drama. It would be even more difficult to write multiple dramas, in real-time, in response to player interaction. Since the current state of AI is nowhere near the point of producing systems that can write good linear plays on their own, then certainly interactive drama is not possible. This argument, however, assumes that an interactive drama system must have the capability to construct stories out of whole cloth, denying human authorship of the AI system itself. But any AI system consists of knowledge (whether represented symbolically, procedurally or as learned probability distributions) and processes placed there by human authors, and has a circumscribed range of situations in which the system can function. The "only" thing an interactive drama system must be able to do is represent a specific space of story potential and move appropriately within this space of story potential in response to player interaction. As argued above, the system doesn't

need to handle arbitrary player actions, but only those that are materially and formally afforded by the specific story space. While still hard, this becomes a much easier problem than building a system that can do everything a human playwright can do and more.

Frasca has proposed an interesting alternative conception of interactive drama based on the dramatic theory of Augusta Boal [Boal 1985]. Frasca's "video games of the oppressed", rather than attempting to immerse the player in a seamless dramatic world, instead invite the player to reflect on and critique the rules of the world, and to communicate this critique to other players by authoring their own behaviors and adding them to the game [Frasca 2001]. For example, someone dealing with alcoholism in their family may create an alcoholic mother character for a *Sims*-like environment and make the character publicly available. Others may download the character, play with it, and offer their own comments and commentary on alcoholic families by posting new alcoholic family member characters. This is certainly a provocative direction to pursue. However, Frasca notes that this Boalian conception of interactive drama provides both a better theoretical *and practical* framework for constructing interactive pieces. But the Boalian technical agenda of building powerful social simulation environments in which non-programmers can use easy-to-learn languages to simulate complex social phenomena is as challenging a technical project as the neo-Aristotelian technical agenda of building dramatic guidance systems. If one is inclined towards making technical impossibility arguments, it is unclear which agenda should be labeled more impossible.

Narrativist Critiques of Interactive Drama

Narrativist[3] critiques of interactive drama, inherited from their critiques of interactive fiction, are concerned that the interactive freedom resulting from making the player a protagonist *in* the world disrupts narrative structure to the point that only simple-minded, "uninteresting" stories can be told. This position is often held by hypertext theorists, who feel that the proper function of interaction in narrative is to engage in structural experiments that push the limits of narrative form, resulting in the "…resolutely unpopular (and often overtly antipopular) aesthetics promoted by hypertext theorists"[Jenkins 2003]. This overtly antipopulist stance can be seen in hypertext theorists reactions to interactive fiction:

> Digital narratives primarily follow the trajectory of *Adventure,* a work considered venerable only by the techies who first played it in the 1970s, cybergaming geeks, and the writers, theorists, and practitioners who deal with interactivity. Hypertext fiction, on the other hand, follows and furthers the trajectory of hallowed touchstones of print culture, especially the avant-garde novel. [Douglas 2000:6-7] (quoted in [Montfort 2003]).

Bernstein specifically places *Façade* within the category of interactive fiction and makes similar arguments to Frasca's, specifically that a first person story inevitably introduces a disruptive battle between the system and the player, and that no AI system will ever be able to respond to the space of actions a player will want to take within a story [Bernstein 2003] (see also Stern's response with respect to *Façade* [Stern 2003]). Of course Berstein's conclusions are the opposite of Frasca's. Rather than remove all narrative structure to open up the space of interaction, Berstein wants to limit interaction by

[3] I use the term "narrativist" as opposed to the more natural "narratologist" to refer to a specific, anti-game, interactive narrative position. While the narrativist position is often informed by narratology, this is not to say that all narratologists are anti-game or that narratology is intrinsically opposed to game-like interaction.

making the reader a witness, a minor character on the periphery of the action. My response to this is similar to my response to Frasca. While I find hypertextual experiments in narrative structure conceptually and aesthetically interesting, I reject any attempt to establish such experiments as the only "culturally legitimate" approach to interactive narrative. And *Façade* is precisely a theoretical, technical and story design experiment in the problems and potentials of building a first-person dramatic story that is about adult relationships, not the heroic travel narrative that narrativists believe first-person interaction inevitably produces.

Middle Ground Positions

A number of theorists have assumed middle ground positions, attempting to find a place for both game elements and narrative elements in the study of games.

Jenkins [Jenkins 2003] argues that while not all games tell stories, a number of strategies are available for weaving narrative elements into a game world, including:

- evoked narratives, in which elements from a known linear narrative are included in the spatial design of the game (e.g. *Star Wars Galaxies*)
- enacted narratives, organized around the player's movement through space (e.g. adventure games),
- embedded narratives, in which narrative events (and their consequences) are embedded in a game space such that the player discovers a story as they progress through the game (e.g. *Half-Life*)
- emergent narratives, narratively pregnant game spaces enabling players to make their own stories (e.g. *The Sims*).

Interestingly, perhaps purposely restricting himself to the current technical state of the art in commercial game design, he does not mention the strategy of actively weaving a player's activity into a story.

Ryan [Ryan 2001], while acknowledging that not all games are productive of narrative, defends the use of narrative as an analytic category in game studies:

> The inability of literary narratology to account for the experience of games does not mean that we should throw away the concept of narrative in ludology; it rather means that we need to expand the catalog of narrative modalities beyond the diegetic and the dramatic, by adding a phenomenological category tailor-made for games.

Ryan's proposal hinges on the relationship between the diagetic and mimetic mode. What allows us to bring narrative analysis to bear on movies and plays is that they are virtually diagetic: an audience member, were they to reflect on and describe their experience, would produce a diagetic narrative. Ryan proposes extending this virtuality one step further, in which a game player, were they to reflect on their action in the game, would produce a dramatic plot. Thus gameplay is virtually mimetic, which is itself virtually diagetic.

Both the ludological and narrativist critiques of interactive drama open up interesting conceptual spaces. I find Frasca's conception of Boalian "videogames of the oppressed" extremely interesting, and hope that he pursues this idea. And the structural experiments of the hypertext community continue to create new modes of literary expression. I certainly don't believe that the conception of interactive drama described in this chapter is the only proper conception of interactive story-like experiences. Nor do I believe that all interactive experiences must be assimilated to the concept of narrative. The ludologists commonly use examples such as chess, *Tetris* or *Space Invaders* in their

analyses, and I agree that such games are most profitably studied using non-narrative analytic tools (but conversely, denying any story-like properties to games such as *The Last Express*, *Grim Fandango*, or *Resident Evil* also does not seem profitable). However, I reject the notion that games and stories are fundamentally irreconcilable categories, that providing the player with an experiences of both agency and story structure is impossible. The neo-Aristotelian theory, and the concrete system that Andrew and I are building, are a theoretical and empirical investigation within this hybrid space of interactive story.

Façade Design Goals

This chapter has situated interactive drama within the space of different approaches to interactive narrative, and has further refined the notion of interactive drama by means of the neo-Aristotelian poetics. This section concludes with a description of our specific design goals for *Façade*.

Project Goals

The project goals are the overarching goals for the project, independent of the particular interactive story expressed within the system.

Artistically Complete

The player should have a complete, artistically whole experience. The system should not be a piece of interactive drama technology without a finished story, nor only a fragment of a story. The experience should stand on its own as a piece of art, independent of any technical innovations made by the project.

Animated characters

The characters will be represented as real-time animated figures that can emote, have personality and can speak.

Interface

The player will experience the world from a first-person 3D perspective. The viewpoint is controlled with the keyboard and mouse.

Dialog

Dialog will be the primary mechanism by which a player interacts with characters and influences how the story unfolds. To achieve dialog, the player types text that is visible on screen; the computer characters communicate with spoken speech. The conversation discourse is real-time; that is, if the player is typing, it is as if they are speaking those words in (pseudo) real-time. The system should be robust when responding to inappropriate and unintelligible input. Although the characters' natural language capabilities are narrowly focused around the topic of the story, the characters have a large variety of responses to off-the-wall remarks from the player.

Interactivity and plot

The player's actions should have a significant influence on what events occur in the plot, which are left out, and how the story ends. The plot should be generative enough that it

supports replayability. Only after playing the experience six or seven times should the player begin to feel they have "exhausted" the interactive story. In fact, full appreciation of the experience requires the story be played multiple times.

Change in the plot should not be traceable to distinct branch points; the player will not be offered an occasional small number of obvious choices that force the plot in a different direction. Rather, the plot should be smoothly mutable, varying in response to some global state that is itself a function of the many small actions performed by the player throughout the experience.

Even when the same plot plays out multiple times, the details of how the plot plays out, that is, the exact timing of events and the lines of dialog spoken, should vary both as a function of the player's interaction and in response to "harmless" random variation, that is, random variation that expresses the same thing in different ways.

Distributable

The system will be implemented on a platform that is reasonably distributable, with the intention of getting the interactive experience into the hands of as many people as possible. It should not just be an interesting demo in a closed door lab, but be experienced by people in the real world. At the time of this writing, all of the *Façade* architecture has been implemented, and the first part of the story has been written using the architecture. This has provided enough experience to validate the architecture and story design. Authoring work continues, with the goal of publicly releasing *Façade* in Fall 2003.

Story Requirements

The story requirements describe the properties that the story itself should have. These are not intended to be absolute requirements; that is, this is not a description of the properties that all interactive stories must have. Rather, these requirements are the set of assumptions grounding the design of our particular interactive story.

Short one-act play

Any one run of the scenario should take the player 10 to 15 minutes to complete. We focus on a short story for a couple of reasons. Building an interactive story has all the difficulties of writing and producing a non-interactive story (film or play) plus all the difficulty of supporting true player agency in the story. In exploring this new interactive art form, it makes sense to first work with a distilled form of the problem, exploring scenarios with the minimum structure required to support dramatically interesting interaction. In addition, a short one-act play is an extreme, contrarian response to the many hours of game play celebrated in the design of contemporary computer games. Instead of providing the player with 40 to 60 hours of episodic action and endless wandering in a huge world, we want to design an experience that provides the player with 10 to 15 minutes of emotionally intense, tightly unified, dramatic action. The story should have the intensity, economy and catharsis of traditional drama.

Relationships

Rather than being about manipulating magical objects, fighting monsters, and rescuing princesses, the story should be about the emotional entanglements of human relationships. We are interested in interactive experiences that appeal to the adult, non-computer geek, movie-and-theater-going public.

Three characters

The story should have three characters, two controlled by the computer and one controlled by the player. Three is the minimum number of characters needed to support complex social interaction without placing the responsibility on the player to continually move the story forward. If the player is shy or confused about interacting, the two computer controlled characters can conspire to set up dramatic situations, all the while trying to get the player involved.

The player should be the protagonist

It was our original intention that the player should experience the change in the protagonist as a personal journey. The player should be more than an "interactive observer", not simply poking at the two computer controlled characters to see how they change. Unfortunately, over time we have had to cut the content that would have most directly served to make the player feel like a protagonist, specifically the "love story" subplot in which a romance develops between one of the characters and the player. While the player is still more than an "interactive observer", she is not the primary protagonist, but rather more like an equal with Grace and Trip.

Embodied interaction should matter

Though dialog should be a significant (perhaps the primary) mechanism for character interaction, it should not be the sole mechanism. Embodied interaction, such as moving from one location to another, picking up an object, or touching a character, should play a role in the action. These physical actions should carry emotional and symbolic weight, and should have a real influence on the characters and their evolving interaction. The physical representation of the characters and their environment should support action significant to the plot.

Action takes place in a single location

This provides unity of space and forces a focus on plot and character interaction.

The player should not be over-constrained by a role

The amount of non-interactive exposition describing the player's role should be minimal. The player should not have the feeling of playing a role, of actively having to think about how the character they are playing would react. Rather, the player should be able to be themselves as they explore the dramatic situation. Any role-related scripting of the interactor [Murray 1998] should occur as a natural by-product of their interaction in the world. The player should "ease into" their role; the role should be the "natural" way to act in the environment, given the dramatic situation.

The Story

Façade is a domestic drama in which you, the player, using your own name and gender, play the character of a longtime friend of Grace and Trip, an attractive and materially successful couple in their early thirties. Tonight is a bit of a reunion; you all first met in college, but haven't seen each other for a couple of years. Shortly after arriving at Grace and Trip's apartment, the evening turns ugly as you become entangled in the high-conflict dissolution of Grace and Trip's marriage. Their marriage has been sour for years;

deep differences, buried frustrations and unspoken infidelities have killed their love for each other. No one is safe as the accusations fly, sides are taken and irreversible decisions are forced to be made. How the façade of their marriage cracks, what is revealed, and the final disposition of Grace and Trip's marriage and their friendship with you depends on your actions. By the end of this intense one-act play, the player has changed the course of Grace and Trip's lives – motivating you to re-play the drama to find out how your interaction could make things turn out differently the next time. The story's controlling idea: To be happy you must be true to yourself.

The details of the *Façade* story design appear in [Chapter 8]. The story design, including the backstory, a description of the *Façade* story values (tension and affinity), a categorization of the different types of beats appearing in the story, and a description of the beat collection for the first third of the story, appear on page 156. Sample annotated story traces, providing examples of player interaction within the story, appear on page 170.

CHAPTER 3
THE *FAÇADE* ARCHITECTURE

Chapter 2 describes our conceptual goals for *Façade*, provides a high-level description of the player experience, and lays out a theoretical foundation informing the conceptual goals and desired experience. This chapter provides a brief introduction to the AI architecture that satisfies these conceptual goals and enables this player experience. Later chapters provide detailed descriptions of individual architectural components.

Some of the material in this chapter, particularly the discussion of the problem of strong autonomy and the development of the dramatic beat as an architectural entity, was first presented in [Mateas & Stern 2000; Mateas & Stern 2002].

Autonomy and Story-Based Believable Agents

Most work in believable agents has been organized around the metaphor of strong autonomy. Such an agent chooses its next action based on local perception of its environment plus the internal state corresponding to the goals and possibly the emotional state of the agent. All decision making is organized around the accomplishment of the individual, private, goals of the agent. Using autonomy as a metaphor driving the design of believable agents works well for believable agent applications in which a single agent is facilitating a task, such as instructing a student (e.g. [Lester & Stone 1997]), or giving a presentation (e.g. [Andre, Rist & Mueller 1998]), or in entertainment applications in which a user develops a long-term relationship with the characters by "hanging-out" with them (e.g. [Stern, Frank & Resner 1998]). But for believable agents used as characters in a story world, strong autonomy becomes problematic. Characters in a story world are not there to believably convey their personalities, but rather to have the right characteristics to take the actions required to move the story forward. That is, knowing which action to take at any given time depends not just on the private internal state of the agent plus current world state, but also on the current story state. And the current story state includes information about all the characters involved in the story, plus the entire past history of the interaction considered *as a story*, that is, as a sequence of actions building on each other and moving towards some end. The global nature of story state is inconsistent with the notion of an autonomous character, which makes decisions based only on private goals, emotion states, and local sensing of the environment.

In order for believable agents to participate in a story, some of the agent decision making must depend on the global story state. Only a small amount of work has been done on the integration of story and character. This work divides the responsibility for state maintenance between a drama manager, which is responsible for maintaining story state, and the believable agents, which are responsible for maintaining character state and making the moment-by-moment behavior decisions [Weyhrauch 1997; Blumberg & Galyean 1995; Assanie 2002]. Given this division of responsibilities, one which *Façade* maintains, the natural question is *how* the drama manager intervenes in character state, that is, by what mechanism does the global story state maintained by the drama manager result in changes in character behavior. One can imagine a spectrum of character autonomy, ranging from an extreme at which the characters maintain strong autonomy (call this the *strong autonomy* position), with only occasional guidance from the drama

manager, to an extreme at which the drama manager continuously intervenes in every character decision (call this the *strong story* position).

The *strong story* position is problematic, particularly for real-time, animated agents. A single decision-making process would have to decide every little action at every moment for all characters in the story, as well as make story-level decisions. The ususal data hiding and modularity arguments apply here: such a program would be hard to write and understand, and consequently unforeseen side effects would arise from cross-talk between "agents". Additionally, character authoring would become conceptually difficult for the author. The very idea of "character" includes the notion of an independent (or quasi-independent) entity pursuing its own goals or desires; a monolithic interactive drama architecture[4] provides no code-level support for this conception (the architecture would provide poor authorial affordances for characters – see Chapter 7).

The *strong autonomy* position, however, is quite seductive. It allows all of the work on individual believable agents to immediately be harnessed within story worlds, and provides a clean separation of concerns between character and story. The two components communicate via a narrow-bandwidth, uni-directional interface flowing from drama manager to agent. The messages sent across this interface consist of goals that characters should assume or perhaps specific actions they should perform. The character is still responsible for most of the decision making. Occasionally the drama manager will modify one or more of the characters behaviors (by giving them a new goal or directly instigating a behavior) so as to move the plot along. In the absence of the drama manager, the character would still perform its normal autonomous behavior. One could author fully autonomous believable agents, which are able to convey their personalities in the absence of any story, drop them into a story world being managed by a drama manager, and now have those characters participate in the story under the drama manager's guidance [Assanie 2002].

The strong autonomy position makes several assumptions regarding the nature of interactive drama and believable agents: drama manager decisions are infrequent, the internal structure of the believable agents can be reasonably decoupled from their interaction with the drama manager, and multiple-character coordination is handled within the agents. We will explore each of these assumptions.

Infrequent guidance of strongly autonomous believable agents means that most of the time, behavior selection for the believable agents will occur locally, without reference to any (global) story state. The drama manager will intervene to move the story forward at specific points; the rest of the time the story will be "drifting", that is, action will be occurring without explicit attention to story movement. Such infrequent guidance may be appropriate for drama managers that sequence large-granularity story units. For example, Weyhrauch's drama manager *Moe* [Weyhrauch 1997] is designed to sequence plot points, that is, to choose the next scene, which accomplishes a plot point, so as to maximize the potential story quality given future player interaction. As scene changes happen infrequently within a story, scene sequencing decisions are of necessity infrequent. Within a scene, another architectural component, a refiner, is responsible for managing the playing out of the individual scene. And it is precisely at the level of the detailed activity *within* a scene that the assumption of infrequent, low-bandwidth guidance become violated. The smallest units of story structure are not large-grained pieces of the

[4]An interactive drama system consisting of a single program is not necessarily a "monolithic" architecture. If, within the single program, believable agents are separate code entities whose decision making is scheduled by the program, then the architecture still provides support for distinct characters, and is thus not an instance of the *strong story* extreme.

scene, but rather are small pieces of dialog and action, sometimes as small as a single action/reaction pair (the dramatic beat – described in the next section). The refiner, or "scene manager", will thus need to continuously guide the autonomous decision making of the agent. In a strongly autonomous agent, this frequent guidance from the drama manager is complicated by the fact that low-bandwidth guidance (such as giving a believable agent a new goal) interacts strongly with the moment-by-moment internal state of the agent (e.g. the collection of currently active goals and behaviors), leading to surprising, and usually unwanted, behavior. In order to reliably guide an agent, the scene-manager will have to engage in higher-bandwidth guidance involving the active manipulation of internal agent state (e.g. editing the currently active goal tree). Authoring strongly autonomous characters for story-worlds is not only extra, unneeded work (given that scene-level guidance will need to intervene frequently), but actively makes guidance more difficult, in that the drama manager will have to compensate for the internal decision-making processes (and associated state) of the agent.

Thinking of believable agents as strongly autonomous, independent characters leads to a style of agent authoring focused on the goals, motivations, behaviors and emotional states of the agent independent of their participation within a story context or their interactions with other agents. The internal structure of these agents is decoupled from consideration of how they will be guided by a drama manager. But, as mentioned above, any goal or behavior-level guidance will strongly interact with the agent's internal decision making processes and state. Reliable guidance will be greatly facilitated by building hooks into the agents, that is, goals and behaviors that are specifically designed to be activated by the drama manager, and which have been carefully crafted so as to override the agent's autonomous behavior in an appropriate manner. But to the extent that authoring story-based believable agents requires special attention to guideability, this brings into question how useful it is to think of the believable agents as strongly autonomous in the first place.

As the drama manager provides guidance, it will often be the case that the manager will need to carefully coordinate multiple characters so as to make the next story event happen. For example, it may be important for two characters to argue in such a way as to reveal specific information at a certain moment in the story. In a sense, the real goal of these two characters is to conspire towards the revelation of a specific piece of information by arguing with each other. But an author who thinks of the characters as autonomous will tend to focus on the individual character goals rather than story-level goals. To make a story-level goal happen, the character author will have to somehow coordinate the individual character goals and behaviors so that as the characters individually react to each other, the resulting interaction "just happens" to achieve the story goal. An alternative to this is to back away from the stance of strong autonomy and provide special goals and behaviors within the individual agents that the drama manager can activate to create coordinated behavior (a specific instance of providing special hooks). But even if the character author provides these special coordination hooks, coordination is still being handled at the individual goal and behavior level, in an ad-hoc way, on a case-by-case basis. What one really wants is a way to directly express coordinated character action at a level above the individual characters, a mechanism providing a mid-point on the spectrum between *strong autonomy* and *strong story*.

Integrating Plot and Character with the Dramatic Beat

Contemporary "how to" books on writing stage and screen plays [Egri 1946; McKee 1997] further develop Aristotle's analysis, moving from the large-scale categories such as plot, character, thought, etc., to a detailed analysis of the structure of individual scenes. In this theory of dramatic writing, stories are thought of as consisting of events that turn (change) values [McKee 1997]. A value is a property of an individual or relationship, such as trust, love, hope (or hopelessness), etc. In fact, a story event is precisely any activity that turns a value. If there is activity – characters running around, lots of witty dialog, buildings and bridges exploding, and so on – but this activity is not turning a value, then there is no story event, no dramatic action. Thus one of the primary goals of an interactive drama system should be to make sure that all activity turns values, and is thus a story event. Of course these values should be changed in such a way as to make some plot arc happen that enacts the story premise. The premise is the controlling idea of the story, such as "Goodness triumphs when we outwit evil", or "To be happy you must be true to yourself".

In dramatic theory, the smallest unit of value change is a called a beat. Any activity below the level of the beat is not associated with value change. Roughly, a beat consists of an action/reaction pair between characters. For example, in the case where action is being carried by dialog, a beat could simply consist of one character speaking a line of dialog, and another character reacting. Generally speaking, in the interest of maintaining economy and intensity, a beat should not last longer than a few actions or lines of dialog.

Beats Become Architectural Entities

The *Façade* drama manager's primary goal is to make sure that *activity* in the story world is dramatic *action*, that is, that it turns values. The beat is precisely the level of abstraction where character activity turns into dramatic action, that is, where character and plot meet. In *Façade*, beats become first class architectural entities, providing a representational unit within the architecture that explicitly coordinates detailed character activity in order to achieve dramatic action. Beats consist of *both* the declarative knowledge needed to sequence beats in a dramatically interesting way and the procedural knowledge necessary for the characters to jointly carry out the dramatic action within the beat.

The declarative knowledge used to sequence beats includes:

- Preconditions – tests that determine when a beat is potentially applicable.
- Priority tests – tests that determine, for beats with satisfied preconditions, how important the beat is relative to other satisfied beats.
- Weight tests – tests that determine the probability of selecting this beat vs. other satisfied beats in the same priority tier.
- Effects – a description of how the beat, if it successfully completes, will change story value(s).

The drama manager uses this knowledge to maintain a probability distribution over satisfied beats, that is, beats with one or more precondition evaluating to true. A beat is drawn from the (changing) distribution whenever a beat sequencing choice must be made.

The various tests are made against story memory (and potentially other memories); story memory maintains state such as the current story values, information about the beats that have already been sequenced, and the current status of the characters' relationship with the player.

Once a beat has been selected, the characters try to accomplish the beat by making use of beat specific character behaviors expressed in the reactive planning language ABL (ABL is described in Chapter 5). The beat behaviors form a micro-story machine that guides the characters in the moment-by-moment activity necessary to accomplish the "mini-story" of the beat, while remaining flexible to the moment-by-moment activity of the player. The drama manager, with its collection of beats, forms a macro-story machine that abstracts above the moment-by-moment activity to guide the story at the level of beat sequencing; by selecting beats, the drama manager "turns on" specific micro-story machines to handle the detailed character activity and player interaction within the beat. While a beat is executing, it is possible that the player may act in such a way that the beat behaviors can no longer make the beat happen. In such a case, player interaction has moved outside the boundaries determined by the capabilities of the micro-story machine; the beat aborts and another beat is selected.

The Function of Beats

Beats serve several functions within *Façade's* architecture. First, beats are the smallest unit of dramatic value change. They are the fundamental building blocks of the interactive story. Second, beats are the fundamental unit of character guidance. The beat defines the granularity of plot/character interaction. Finally, the beat is the fundamental unit of player interaction. The beat is the smallest granularity at which the player can engage in meaningful (having meaning for the story) interaction. A player's activity is interpreted as having affected the story only to the extent that this activity participates in a beat (of course, the detailed "sub-story" player activity within a beat influences the performance of the beat).

A Response to the Problem of Autonomy

Coordinating multi-character activity via beat specific behaviors that are swapped in and out on a beat-by-beat basis addresses the problem of strong autonomy in the context of story-based believable agents. It provides a concrete mechanism for falling at a mid-point in the spectrum between *strong autonomy* and *strong story*. In this architecture, the individual characters are no longer strongly autonomous. In the absence of the drama manager, the characters will not take action (or perhaps will only have very simple reactions to the environment). The beat-level of the drama manager provides frequent guidance to the characters by giving them reactive joint behaviors to carry out. The decision about which beat to sequence next is made based on the global story state. Multiple characters are coordinated at the beat-level; character authors are not forced to provide ad-hoc coordination within individual characters. Since the characters contain only low-level goals and behaviors, there is no complex character state complicating dramatic guidance. There is no longer a tension between authoring self-contained autonomous characters that have independent motivations, and providing those characters with the appropriate "hooks" to support guidance by an external process. Instead, the characters become libraries of character-specific ways of accomplishing low-level tasks; all higher-level motivation is provided by the drama manager. Thus this architecture addresses the tension between autonomy and dramatic guidance by backing away from

strong autonomy on the part of characters and instead factoring high-level character behavior across the units of dramatic guidance. Comparing this approach with earlier Oz Project work, if the *strong autonomy* end of the spectrum is labeled 0, and the *strong story* end of the spectrum is labeled 100, then the *Façade* architecture falls around 70, while earlier Oz project work falls around 30 [Bates, personal communication].

Joint Goals and Behaviors

The procedural knowledge associated with a beat is expressed as a collection of behaviors written in the behavior language ABL. Many of these behaviors are joint behaviors, an ABL construct that provides explicit coordination support for multiple characters. These joint behaviors describe the coordinated activity required of *all* the characters in order to carry out the beat. As discussed above, it is possible to write individual character behaviors that use ad-hoc communication (either in the form of sensing, or some form of direct, out-of-band message passing) to achieve multi-character coordination. It is difficult, however, for a behavior author to understand ahead of time all the synchronization problems that can occur; as unforeseen synchronization problems appear during play-testing, repeated patching and re-authoring of the behaviors will be necessary. In addition, the behavior author will have to separately solve the coordination problems of each new behavior involving multiple characters. In contrast, multi-agent coordination frameworks such as joint intentions theory [Cohen & Levesque 1991] or shared plans [Grosz & Kraus 1996] provide a systematic analysis of all the synchronization issues that arise when agents jointly carry out plans. These formal analyses have been used as the communication requirements for multi-agent coordination frameworks such as STEAM, an extension of SOAR (implemented as a collection of SOAR productions) providing explicit support for multi-agent coordination [Tambe 1997]. In STEAM, when a joint plan is carried out, the architecture automatically takes care of all the necessary message passing to guarantee coordination. ABL provides a similar extension of the reactive planning language Hap, [Loyall & Bates 1991; Loyall 1997], a language specifically designed for the authoring of believable agents.

Since beats hand the characters joint behaviors that are designed to accomplish that specific beat, most (perhaps all) of the high level goals and behaviors that drive a character will no longer be located within the character at all, but rather will be parceled out among the beats. Given that the purpose of character activity within a story world is to create dramatic action, this is an appropriate way of distributing the characters' behavior. The beat is precisely the smallest unit of dramatic action (the smallest unit that turns values). The character behavior is now organized around the dramatic functions that the behavior serves, while still maintaining distinct character decision making processes, avoiding the *strong autonomy* (behavior organized around a conception of the character independent of the dramatic action) or *strong story* (character action is described *only* in terms of story structure) extremes. Since the joint behaviors associated with beats are still reactive behaviors, there is no loss of character reactivity to a rapidly changing environment. Low-level goals and behaviors (e.g. locomotion, emotion expression, personality moves, etc.) are still contained within individual characters. These low-level behaviors provide a library of character-specific actions that are available to the higher-level behaviors handed down by the beats.

Finally, before leaving this section on joint behaviors, it is important to point out that in *Façade*, joint goals and behaviors are written to carry out individual beats. It is the

beats rather than the joint behaviors that are responsible for maintaining the story arc. Thus, within *Façade's* architecture, it won't be necessary to write huge, complex joint behaviors that tightly coordinate multiple characters for minutes on end while responding appropriately to every possible contingency. Rather, joint behaviors only coordinate characters for the duration of a beat (or perhaps several beats) so as to incrementally advance specific story values. This beat-level modularity helps tame the combinatorial complexity associated with tightly coordinating character activity across time.

Natural Language Processing

In *Façade*, much of the story action is carried by dialog between the player and the characters. Given the kind of story we want to tell, an adult domestic drama structured according to the principles of traditional, Aristotelian narrative, the use of language in the story is unavoidable. However, natural language is an exceedingly complex phenomenon. There is a long and continuing history in AI of complex attacks on specific, narrow aspects of natural language use. There exist no general theories, techniques or systems that can handle the syntactic, semantic and pragmatic breadth of the language use that occurs in *Façade*. Instead, *Façade* makes use of specific (non-general), a-theoretical, author-intensive techniques to understand natural language typed by the player. *Façade* does no natural language generation for the characters. All the possible lines of dialog the characters might speak have been pre-written[5]. We use pre-written dialog in order to maintain tight control over the quality of language. Further, since the state of the art in speech synthesis is not capable of producing high quality, rich, emotive vocal performances, the vocal performance of all lines of dialog is prerecorded by voice actors.

Our approach in *Façade* is to view the natural language understanding problem as a dialog management problem. Dialog management focuses on the pragmatic effects of language (what a language utterance does to the world) rather than with the syntax (the form of surface text) or semantics (meaning) of language. Dialog management views a language use situation as consisting of a discourse context within which conversants exchange speech acts. A conversant's surface utterance is interpreted *as* a speech act in a manner dependent on the current discourse context. That is, the same surface utterance can be interpreted as a different speech act in different discourse contexts. When a speech act is recognized, it changes the shared discourse context, perhaps moving to the next step of a social script, or in some other way changing the internal mental state of the conversation partner. Typical generic examples of speech acts include inform(<content>) and question(<content>), meaning, respectively, "I've told you about <content> and you should update your mental state accordingly and acknowledge that you've updated it" and "I'm asking you a question about <content> and you are socially obligated to answer the question". Though dialog management per se is concerned only with pragmatics, syntax and semantics still play a role. A surface utterance must somehow be turned into a speech act; this means there must be some means of manipulating the syntax of the surface utterance. And speech acts are often *about* something (like the <content> in the two speech acts above); the aboutness *is* the semantics of the utterance. In *Façade*, discourse management is divided into two phases: phase I maps surface text into speech acts (here called discourse acts), while phase II maps discourse acts into one or more character responses.

[5] There is a small amount of phrase substitution, in which an individual word or phrase is substituted within a sentence. This is used primarily to handle different player names.

Phase I: Surface Text to Discourse Acts

In the spirit of shallow semantic parsing, forward chaining rules map surface text to discourse acts. Some rules map patterns of surface text directly to intermediate meaning representations, while other rules combine intermediate meanings to form more complex meanings. Eventually the process of mapping islands of surface text into intermediate representations, and combining these representations, produces the final meaning, consisting of one or more discourse acts.

Rules are written in a custom rule language that compiles to Jess [Friedman-Hill 1995-2002], a java implementation of the CLIPS rule language [NASA 1985-2002]. Our custom rule language looks just like Jess with the addition of an embedded template description language that allows compact descriptions of surface text patterns to appear on the left hand sides of rules.

Discourse act representations are relatively flat. Rather than consisting of complex, symbolic constructions supporting compositional semantics, *Façade* discourse acts are simple frames whose slots tend to be filled with atomic tokens. For example, the discourse act ReferTo, produced when the player makes a reference to something, has only two slots: character, which takes a token representing the character the ReferTo was directed towards (if any), and object, which takes a token representing either a physical object (e.g. WeddingPicture) or an abstract object such as a topic (e.g. RockyMarriage). The phase I pipeline would produce a discourse act such as ReferTo(character: Grace, object: WeddingPicture) by:

1. Recognizing a surface text pattern that indicates an utterance is directed towards Grace. There are a number of template rules looking for different ways of directing an utterance towards Grace.
2. Recognizing a surface text pattern that indicates the wedding picture. There are a number of template rules looking for different ways of referring to the wedding picture.
3. Chaining together the intermediate semantic representations for DirectedTowards(character: Grace) and PhysicalObject(obj: WeddingPicture) to produce the ReferTo frame.

The embedded template description language includes mechanisms for specifying that Wordnet [Fellbaum 1998] expansions should be employed during template matching to map synonyms into a single canonical term as well as to move from more specific to more general terms. For example, if the player asked “Can I have a glass of Chablis?”, the Wordnet thesaurus can be used to map “Chablis” to the more general “wine” or even more general “alcoholic beverage”. In this way, the mapping rules don't need to know anything about “Chablis” (and the hundreds of other words that denote specific alcoholic beverages), only about “alcoholic beverages”.

Beats are the primary representation of discourse context. The beat organizes activity as dramatic action, that is, action that changes values. Dialog, as one of the activities (perhaps in *Façade*, the primary activity) that changes values, *is* action. It therefore makes sense that the current beat defines the most immediate dialog context. Specific mapping rules for the recognition of speech acts can be associated with each beat. More general mapping rules are active across sequences of beats or active all the time. The beat-specific rules are activated when a beat is selected and deactivated when the beat completes (either by aborting or completing successfully). These beat-specific rules may live side-by-side with more general rules or may override some or all of the more general rules. Since beats are the primary representation of discourse context, the beat-specific rules provide the mechanism by which surface text can be recognized as discourse acts in

a context-specific way. The current beat defines where we are in the story and what kinds of surface text responses we might expect.

Phase II: Discourse Acts to Reactions

Once phase I has recognized one or more discourse acts in the surface text, phase II determines what the (joint) reaction will be to discourse acts. Where phase I determines, in a context specific way, *which discourse acts* were produced by the player, phase II determines *what effect* these acts will have, including, potentially, how they change the discourse context. At any point in time, one or more reaction contexts are active. Each of these active contexts contains one or more proposers, which propose potential reactions as a function of the discourse act(s). A priority is associated with each proposed reaction within a context as well as with each active context. Generally, the winning reaction is the highest priority reaction within the highest priority context. Contexts may also specify priority mappers that dynamically modify the priorities of reactions and contexts as a function of all the proposed reactions and active contexts.

The current beat determines which reaction contexts are active. This allows beats to change the interpretation of discourse acts. For example, in $Beat_1$, Grace and Trip's wedding picture may play no role in the beat's action. $Beat_1$ therefore doesn't provide a higher-priority, more specific context for proposing reactions in response to a reference to the wedding picture. If the player mentions the wedding picture during $Beat_1$, the lower priority general context would propose a more general response; since no higher priority context proposes a response, this lower priority response would be selected and performed. However, in $Beat_2$, mentioning the wedding picture might be interpreted as siding with Grace. $Beat_2$ would therefore activate a higher priority, more specific context that proposes a reaction that successfully completes the beat with Grace feeling a greater affinity for the player (Grace and Trip of course perform this affinity change so that the player knows about it).

The beat, as the representation of the current discourse context, thus provides both context-specific recognition of, and reaction to discourse acts. Recognized discourse acts in turn change the discourse context through the chosen reactions.

Overview of the Architecture

Figure 3-1 depicts *Façade's* interactive drama architecture. This architecture is a concrete instantiation of the generic Oz architecture in Figure 1-3 (page 7). The major components of the architecture are the story world, the believable agents within the world, the drama manager, the story memory, and the natural language processing system.

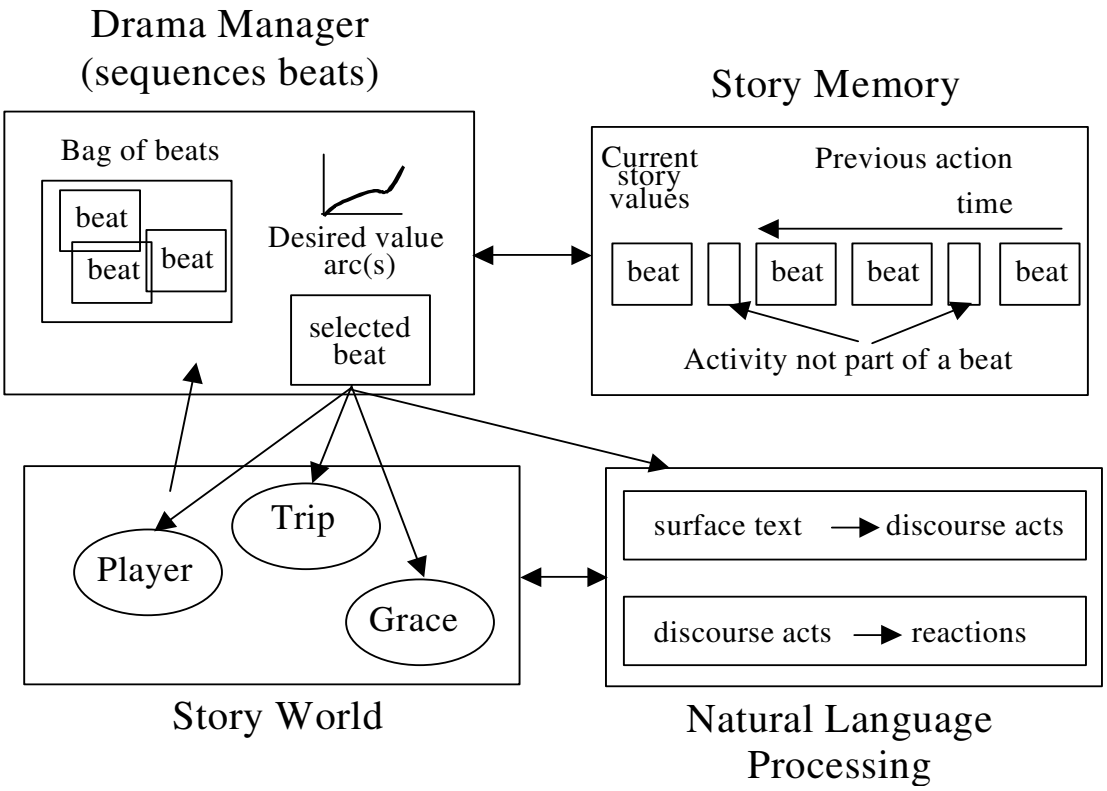

Figure 3-1. *Façade* interactive drama architecture

Story World

The story world is the graphical world within which the player and the believable agents (in this case, Trip and Grace) enact the story.

Each of the believable agents contains a library of low-level behaviors. The agents' activity within a beat is coordinated by the beat-specific higher level joint and individual behaviors associated with the beat. These high-level beat-specific behaviors make use of the low-level general behaviors.

Façade uses a custom non-photorealistic animation engine. The animation engine is responsible for modeling and rendering the physical space of the story, providing the interface through which the player interacts with the world, and providing the character bodies. The character bodies consist of the graphical representation of the character, the mechanisms for animating the graphical representation, and the sensory-motor interface through which the minds of the characters (ABL behaviors) can issue commands that effect the animation of the body (e.g. causing the right arm to make a gesture) and request information about the current world state. Character animation is achieved through a mixture of purely procedural animation (e.g. facial expressions), playback of animation data (e.g. a temporal sequence of numbers describing the progression of joint rotations necessary to make the right arm reach for an object), and layering and procedural tweaking of animation data. Behaviors control the animation of a character's body by issuing commands (e.g. "play this animation script" or "assume this facial expression") that effect the animation pipeline.

The player avatar is the system representation for the locus of player activity. In *Façade*, there is no graphical representation for the locus of activity; as the experience is first-person, the player never sees her own body in the world. Nevertheless, there is computational activity associated with the player; it is this computational activity that is the avatar. Compound sensors, which look for patterns in player activity, are written as ABL behaviors. These behaviors execute within an independent ABL agent, the player avatar. As the player compound sensors recognize patterns of activity, these patterns become available to the rest of the system, including the believable agents Grace and Trip, and can thus influence decision making.

As a beat progresses, the player's interaction within the beat will either be handleable within the scope of the beat, allowing the beat to accomplish its story action and complete, or will step outside the bounds of the beat, causing the beat to abort. The beat-specific behaviors are responsible for determining whether a beat has succeeded or must abort. In either case, the beat behaviors inform the drama manager, causing the drama manager to select a new beat.

Drama Manager

The drama manager is responsible for selecting beats in such a way as to move the story forward. The drama manager has a pool of possible beats from which to choose and a specification of desired value arc(s) describing how the story value(s) should change over time. The drama manager uses the declarative knowledge of the individual beats, plus the story arc, to make beat selection decisions. To select a beat, the drama manager constructs a probability distribution over the set of satisfied beats (beats in the pool with satisfied preconditions) and draws a beat from this distribution. The shape of the distribution is a function of how well the effects of the satisfied beats match the shape of the desired story arc, plus the outcomes of the satisfied beats' weight tests and priority tests. The selected beat passes beat-specific (joint) behaviors to the characters and activates beat-specific surface text rules and reaction contexts for natural language processing.

Information about the beat selection process (e.g. the pool of satisfied beats) is stored in story memory. This allows future beat selection decisions to condition on the details of previous decisions. When the drama manager is informed that a beat has terminated (either aborted or completed successfully), it writes this information to story memory as well, including updating story values for successful beats with story value effects. The vast majority of the matching done during the execution of various beat tests is done against the story memory.

Natural Language Processing

When the player enters text it is passed to the natural language processing system for interpretation. The first processing phase maps the surface text to one or more discourse acts. For example, "I like your couch, Grace" results in the construction of two discourse acts: ReferTo(character: Grace, object: couch) and Praise(character: Grace). The second processing phase takes the one or more discourse acts resulting from the first phase and chooses a reaction. If the two discourse acts are the ReferTo and Praise above, proposers in the various active contexts will propose multiple reactions, some reacting to the ReferTo and some reacting to the Praise. The highest priority reaction from the highest priority context is chosen and handed back to the characters. For example, it may be the

case that in this particular beat a Praise is interpreted as disagreeing with Grace (Grace is playing a head game in which she wants the player to say that her latest decorating scheme doesn't work, thus poking a small hole in Trip's "perfect life, perfect home" charade); the highest priority reaction is a transition out of the beat that lowers the affinity between Grace and the player.

CHAPTER 4
AI AND ART

This chapter begins the theoretical formulation of my own practice of AI-based cultural production: Expressive AI. Before proceeding, I would like to explain *why* Expressive AI is a part of this thesis. If either an AI-researcher or an artist decided that she wanted to build an AI-based art work, but didn't change her ways of working and thinking, her habits of mind, I firmly believe she would fail. This failure would manifest both as a failure to produce a compelling work of art and as a failure to advance AI (through the exploration of new techniques, representations and architectures) in a manner useful for artistic expression. Building something like *Façade* (or the other AI-based pieces described in the appendices) requires a new way of working and thinking. At a high level, this new practice can be described as one in which the practitioner functions as both an artist and an AI scientist. But this formulation is frustratingly vague. What does it *mean* to combine art practice and AI research? Expressive AI is my attempt to provide an answer to this question.

This chapter surveys the terrain of AI research vis-à-vis art practice. How do artists and cultural theorists view different AI research agendas? What differences exist between art practice and AI research? The answers to these questions lay the groundwork for the development of an integrated practice. Parts of this chapter first appeared in [Mateas 2000a; Mateas 2001a].

The Classical/Interactionist AI Debate

In recent years, discourse about AI's high-level research agenda has been structured as a debate between symbolist, classical AI (sometimes called Good Old Fashioned AI or GOFAI [Haugeland 1985]), and behavioral, or interactionist AI. The classical/interactionist distinction has shaped discourse both within AI and cognitive science [Brooks 1990; Brooks 1991; CogSci 1993], in cultural theoretic studies of AI [Adam 1998], and in hybrid practice combining AI and cultural theory [Agre 1997a; Sengers 1998a; Varela, Thompson & Rosch 1999]. This debate has shaped much contemporary practice combining AI and cultural production, with practitioners commonly aligning themselves with the interactionist camp. Because of this connection with cultural practice, it will be useful to position Expressive AI relative to this debate. In this section I briefly describe the classical/interactionist debate, and diagnose why it is that contemporary cultural practitioners would find the interactionist position particularly compelling. Then I describe how the goals of Expressive AI as a practice are distinct from the goals of both the classical and interactionist agendas.

Characterizing Classical and Interactionist AI

Classical AI is characterized by its concern with symbolic manipulation and problem solving [Brooks 1991]. A firm distinction is drawn between mental processes happening "inside" the mind and activities in the world happening "outside" the mind [Agre 1997a]. Classical AI's research program is concerned with developing the theories and engineering practices necessary to build minds exhibiting intelligence. Such systems are

commonly built by expressing domain knowledge in symbolic structures and specifying rules and processes that manipulate these structures. Intelligence is considered to be a property that inheres in the symbolic manipulation happening "inside" the mind. This intelligence is exhibited by demonstrating the program's ability to solve problems.

Where classical AI concerns itself with mental functions such as planning and problem solving, interactionist AI is concerned with embodied agents interacting in a world (physical or virtual) [Brooks 1991; Agre 1997a]. Rather than solving complex symbolic problems, such agents are engaged in a moment-by-moment dynamic pattern of interaction with the world. Often there is no explicit representation of the "knowledge" needed to engage in these interactions. Rather, the interactions emerge from the dynamic regularities of the world and the reactive processes of the agent. As opposed to classical AI, which focuses on internal mental processing, interactionist AI assumes that having a body embedded in a concrete situation is essential for intelligence. It is the body that defines many of the interaction patterns between the agent and its environment.

The distinctions between the kinds of systems built by classical and interactionist AI researchers is summarized in Table 4-1.

Classical AI	*Interactionist AI*
Narrow/deep	**Broad/shallow**
General	**Fits an environment**
Disembodied	**Embodied and situated**
Semantic symbols	**State dispersed and uninterpreted**
Sense-plan-act	**Reactive**

Table 4-1. Contrasting properties of classical and interactionist AI systems

Classical systems often attempt to *deeply* model a *narrow*, isolated mental capability (e.g. reasoning, memory, language use, etc.). These mental components duplicate the capabilities of high-level human reasoning in abstract, simplified environments. In contrast, interactionist systems exhibit the savvy of insects in complex environments. Interactionist systems have a *broad* range of *shallow* sensory, decision and action capabilities rather than a single, *narrow*, *deeply* modeled capability.

Classical AI seeks general solutions; *the* theory of language understanding, *the* theory of planning, etc. Interactionist AI starts with the assumption that there is a complex "fit" between an agent and its environment; there may not be generic solutions for all environments (just as many animals don't function well when removed from their environment).

Classical AI divorces mental capabilities from a body; the interface between mind and body is not commonly addressed. Interactionist AI assumes that having a body embedded in a concrete situation is essential for intelligence. Thus, interactionists don't buy into the Cartesian split. For them, it is the body that defines many of the interaction patterns between the agent and its environment.

Because of AI's historical affinity with symbolic logic, many classical systems utilize semantic symbols – that is, pieces of composable syntax that make one-to-one reference to objects and relationships in the world. The state of the world within which the mind operates is represented by a constellation of such symbols. Interactionist AI, because of its concern with environmental coupling, eschews complex symbolic representations;

building representations of the environment and keeping them up-to-date is notoriously difficult (e.g. the frame and symbol grounding problems). Some researchers, such as Brooks [Brooks 1990; Brooks 1991], maintain the extreme position that *no* symbolic representations should be used (though all these systems employ state – one can get into nasty arguments about what, precisely, constitutes a symbol).

In classical systems, agents tend to operate according to the sense-plan-act cycle. During sensing, the symbolic representation of the state of the world is updated by making inferences from sense information. The agent then constructs a plan to accomplish its current goal in the symbolically represented world by composing a set of operators (primitive operations the agent can perform). Finally, the plan is executed. After the plan completes (or is interrupted because of some unplanned-for contingency), the cycle repeats. Rather than employing the sense-plan-act cycle, interactionist systems are reactive. They are composed of bundles of behaviors, each of which describes some simple action or sequence of actions. Each behavior is appropriate under some environmental and internal conditions. As these conditions constantly change, a complex pattern of behavioral activation occurs, resulting in the agent taking action.

Interactionist AI's Affinity with Cultural Theory

Interactionist and classical AI are two technical research agendas within AI, each determining a collection of research problems and system-building practices. In this section, I examine the cultural theoretic association between interactionist AI and contemporary artistic practice.

Cultural theory is a diverse collection of literary, historical and sociological practices concerned with understanding the metaphors and meaning systems by which culture is composed. For cultural theorists, any cultural formation can be "read" in the same manner that one might analyze a text, seeking an understanding both of the dynamic and endlessly ramifying life the formation has within culture and the ways in which the formation is a historically contingent product of a specific cultural milieu. Cultural theory undermines the distinction between "fanciful" sign systems (e.g. literature, art), which are clearly understood as contingent, social constructions, and "true" sign systems (e.g. gender definitions, perspective vision), which are generally understood as being pre-cultural (and thus existing outside of culture). Politically, cultural studies is engaged in a project of emancipation. Social inequities are supported by unexamined beliefs (that is, "truths") about the nature of humanity and the world. For example, the inferior role of women in society is generally understood within cultural studies as being supported by the system of Enlightenment rationality (in addition to other meaning systems). By understanding the subjugating meaning system as culturally contingent, the absolute ground from which the system operates is undermined.

Cultural theory's affinity with interactionist AI is based in a critique of Enlightenment rationality. Starting with Descartes, Enlightenment thinkers developed a theory of rationality, defining thought in terms of abstract, preferably formal operations taking place in an inner mental realm divorced from the world of gross matter. This conception of intelligence, with the twist of embedding mental operations in a material base (the brain) while still maintaining a strong split between the inner mental world and the outer world, dominates the contemporary understanding of mind. In fact, this meaning system is so hegemonic as to make it difficult to conceive of any alternative. This is precisely the kind of situation cultural theorists love to interrogate; by revealing the historical and cultural relativity of the meaning system, and thus rendering it contingent, a space of alternatives is opened up. For the case of the Enlightenment conception of mind, this

analysis has focused on revealing the ways in which interaction with the world, and particularly the notion of an embodied actor marked with a specific racial and sexual identity, was systematically marginalized. In keeping with the political project of cultural theory, this marginalization of embodiment has been seen as theoretical support for the white, male subjugation of women and people of color. Interactionist AI, as a technical research agenda, seems to be reaching the some of the same conclusions as this cultural theoretic project. Some cultural theorists explicitly acknowledge this alignment [Adam 1998]. This results in some of the moral energy associated with the political component of the cultural theoretic project transferring to the technical agenda; interactionist AI is associated with freedom and human rights and classical AI with oppression and subjugation.

Much of contemporary art practice is no longer concerned with the modernist agenda of perfecting purely formal elements. Rather, this practice involves self-consciously questioning cultural forms, representational modes and tropes, exploring the boundaries of these forms, breaking the representation, questioning whose power is preserved by a representational mode, and hybridizing modes in order to create new ones, all from a position of extreme cultural self-consciousness. This self-conscious concern with meaning systems makes contemporary art practice and cultural theory natural allies, with many artists being informed by and participating in cultural theoretic analysis. And through this link with cultural theory, many artists inherit their attitude towards AI, aligning with interactionist AI (and bottom-up methods in general) while feeling a generalized distrust of classical AI, often accompanied with a sense of moral outrage acquired from cultural theory's political project. Additionally, classical AI is seen as a failed dream of rationalist hubris. Early AI research was accompanied by extremely optimistic claims that AI systems would exceed human intelligence by the end of the 20th century. When the predictions failed to materialize, resulting in the funding dry spell known as "AI Winter" in the mid to late 1980s, it became easy to read the growing interest in bottom-up techniques at the end of AI Winter as a pure progress narrative – new methods, full of promise, replacing the old, bad methods. For these reasons, many artists came to see *interactionist AI as peculiarly suited for cultural production*.

AI & Cultural Production

Expressive AI does *not* view interactionist AI, or any particular AI technical agenda, as possessing a privileged role in AI-based cultural production. This section disrupts this affinity, preparing the ground for the development of a practice that engages and transforms the whole of AI.

The Limitations of Agent as Metaphor

Within the AI community, the interactionist/classical debate is organized around the idea of an agent. Within AI, an agent is understood as an autonomous entity existing in an environment, able to sense and act on this environment. Historically, interactionist AI appeared as a reaction to recurring problems in classical AI in the design of complete agents, particularly robots [Brooks 1990; Brooks 1991]. In recent years, the AI research community has indeed begun converging on reactive techniques for agent design, proposing a number of reactive and hybrid (combining search and reactivity) architectures for robotic and virtual agents. However, AI-based cultural production is broader than agent design. For example, while both *Subjective Avatars* (Appendix A)and

Office Plant #1 (Appendix B)can be understood as agents, *Terminal Time* (Appendix C) is not an agent (at least it can't be understood as an agent without broadening the notion of agent until it is vacuous), and yet is indisputably an instance of AI-based cultural production. In fact, *Terminal Time* makes heavy use of classical techniques. An AI-based artist aligning herself too strongly with interactionist techniques may find that all her work becomes assimilated to the metaphor of agent, thus missing out on a rich field of alternative strategies for situating AI within culture.

Façade is an interesting case with respect to the agent metaphor. While certainly making use of the agent metaphor (the characters are believable agents), it locates a problem in the context of interactive story with the agent assumption of strong autonomy. *Façade* is a hybrid, combining the agent metaphor with a more centralized story generation framework.

Cultural Production vs. AI.

For the artist, even more important than recognizing the way that the metaphor of agency structures the interactionist/classical technical debate, is recognizing that *both* interactionist and classical AI share research goals at odds with the goals of AI-based cultural production. Table 4-2 summarizes some of the differences between cultural production and traditional AI research practice.

Cultural Production	*AI*
Poetics	**Task competence**
Audience Perception	**Objective measurement**
Specificity	**Generality**
Artistic abstraction	**Realism**

Table 4-2. Contrasting goals of cultural production and AI

Artists are concerned with building artifacts that convey complex meanings, often layering meanings, playing with ambiguities, and exploring the liminal region between opaque mystery and interpretability. Thus the purpose of, motivation behind, or concept defining any particular AI-based artwork will be an interrelated set of concerns, perhaps not fully explicable without documenting the functioning of the piece itself. In contrast, the focus in AI is on task competence, that is, on demonstrably accomplishing a well defined task. "Demonstrably accomplishing" means being able to show, either experimentally or by means of mathematical proof, that the AI system accomplishes the task. "Well defined task" means a simple, concisely defined objective that is to be accomplished using a given set of resources, where the objective often has "practical" (i.e. economic) utility. In classical AI, task competence has often meant competence at complex reasoning and problem solving. For interactionist AI, task competence has often meant moving around in complex environments without getting stepped on, falling off a ledge, or stuck behind obstacles. In describing *Office Plant #1* (*OP#1*) to AI researchers (and more generally, CS researchers), I often confront this distinction between poetics and task competence. A technical researcher tends to view *OP#1* as a sophisticated email indicator, useful for indicating to the user whether they should read their mail or not. That is, *OP#1* is viewed as a mechanism for facilitating the task of reading and answering

email. The notion that *OP#1* is really about creating a presence whose behavior should correlate with email activity while maintaining a sense of mystery, and whose "function" is to open a contemplative window onto its owner's daily activity, is only communicated to a technical practitioner with some difficulty.

The success of an AI-based artwork is determined by audience perception. If the audience is able to participate in the poetics defined by the artist, that is, engage in an interpretive process envisioned by the artist, then the piece is successful. AI tries to measure success objectively. How many problems could the program solve? How long did the robot run around before it got into trouble? How similar is the system's solution to a human's solution? The artist is concerned with the subjective experience of the audience, where the AI researcher strives to eliminate any reference to human perception of their artifact. All four AI-based artworks described above are intimately concerned with audience experience. *Subjective Avatars* structures a participant's perceptions so as to help her experience a virtual world from an alien subjective viewpoint. *OP#1* creates a variable sculptural presence reflecting its owner's daily activity. *Terminal Time* makes visible ideological bias in the construction of history by generating biased histories in response to audience feedback. *Façade* provides the player with a sense of agency within a dramatically structured story experience. There is no audience-free vantage point from which to consider these systems.

Artists build specific works. Each piece is crafted so as to establish a specific poetics, so as to engage the audience in specific processes of interpretation. The artist explores meaning-making from the vantage point of his or her particular cultural situation. AI, like most sciences, tries to create general and universal knowledge. Even interactionist AI, while stressing the importance of an agent's fit to its environment, seeks general principles by which to describe agent/environment interactions. Where AI conceives of itself as searching for timeless truths, artists participate in the highly contingent meaning systems of a particular cultural milieu. Even those AI practitioners engaged in the engineering task of building "smarter" gizmos here and now, and who would probably demure from the "timeless truth" characterization of AI practice, are still committed to building generally applicable engineering tools. *Subjective Avatars* provides an example of Expressive AI's focus on specificity. The characters in *Subjective Avatars* were built using Hap, a language designed to facilitate the crafting of specific, unique characters [Loyall & Bates 1991]. This is in contrast to both ALife and top-down approaches to character, which attempt to define universal character frameworks in which specific characters are "tuned-in" by adjusting parameters in the model [Mateas 1999b]. *Façade* extends the character specificity of Hap by supporting the crafting of specific story "chunks" (beats).

Finally, artists engage in abstraction. That is, they are not so much concerned with building exact replicas of parts of the world (mimesis), as with creating meaning systems that make reference to various aspects of the lifeworld (the amalgam of the physical world plus culture). Conversely, much of AI research is motivated by realism. A classical researcher may claim that their program solves a problem the way human minds really solve the problem; an interactionist AI researcher may claim that their agent *is* a living creature, in that it captures the same environment/agent interactions as an animal. The first time I presented *Terminal Time* to a technical audience, there were several questions about whether I was modeling the way that real historians work. The implicit assumption was that the value of such a system lies in its veridical model of human behavior. In fact, the architectural structure of *Terminal Time is* part of the concept of the piece, not as a

realist portrait of human behavior, but rather as a caricature of certain institutionalized processes of documentary film making.

A Random Walk Around the AI Landscape

This chapter has so far organized the discussion around the binary opposition of Classical vs. Interactionist AI. While this distinction has been strongly represented (perhaps over-represented) in discussions of AI and culture, it by no means exhausts the discourse in and about AI. Many other distinctions, disputes and historical currents flow around the AI landscape, all of which can serve as conceptual resources for the AI-based cultural practitioner.

Neats vs. Scruffies

Abelson [Abelson 1981] introduced the terms *neat* and *scruffy* to describe two styles of AI research, styles that are deeply intertwined with the personality, worldview, and value systems of the individual researcher. These terms are now broadly used in AI to describe these two styles of work.

> The one tendency points inside the mind, to see what might be there. The other points outside the mind, to some formal system which can be logically manipulated. Neither camp grants the other a legitimate claim on cognitive science. One side says, "What you're doing may seem to be science, but it's got nothing to do with cognition." The other side says, "What you're doing may seem to be about cognition, but it's got nothing to do with science." …
>
> …Indeed, the stylistic division is the same polarization that arises in all fields of science, as well as in art, in politics, in religion, in child rearing – and in all spheres of human endeavor. Psychologist Silvan Tomkins (1965) characterizes this overriding conflict as that between characterologically left-wing and right-wing world views. The left-wing personality finds the sources of value and truth to lie within individuals, whose reactions to the world define what is important. The right-wing personality asserts that all human behavior is to be understood and judged according to rules or norms which exist independently of human reaction. A similar distinction has been made by an unnamed but easily guessable colleague of mine[6], who claims that the major clashes in human affairs are between the "neats" and the "scruffies". The primary concern of the neat is that things should be orderly and predictable while the scruffy seeks the rough-and-tumble of life as it comes. [Abelson 1981:1].

Abelson argues that the proper way forward in AI and cognitive science is to somehow merge these two styles, to remain open to the complexities of the messy reality of human experience while developing principled and elegant characterizations of this reality. But this fusion is difficult.

> The strategy of moving leftward from the right suffers from a seemingly permanent limitation on the kinds of content and process you are willing to consider. If what really matters to you is the formal tractability of the domain of investigation, then your steps are likely to be small and timid…

[6] Roger Schank

> … What are the difficulties in starting from the scruffy side and moving toward the neat? The obvious advantage is that one has the option of letting the problem area itself, rather than the available methodology, guide us about what is important. The obstacle, of course, is that we may not know how to attack the important problems. More likely, we may think we know how to proceed, but other people may find our methods sloppy. We may have to face accusations of being ad hoc, and scientifically unprincipled, and other awful things. [Abelson 1981:1-2]

As research styles, neat and scruffy cut across technical approaches. Within classical AI, the neats are the logicists (e.g. [McCarthy 1968; McCarthy & Hayes 1969), who argue that broad, hard reasoning problems, such as the inferences made in everyday, common sense reasoning, can only be solved be developing logical frameworks with the right formal properties. Yale School AI (Abelson was part of this lab) [Schank & Reisbeck 1981] are the canonical scruffies of classical AI, developing systems that generate stories [Meehan 1976], understand stories [Cullingford 1981; Wilensky 1981; Dyer 1983], engage in ideologically biased reasoning [Abelson & Carroll 1965; Carbonell 1979], model human memory [Kolodner 1984], and other knowledge rich tasks.

Interactionist AI also has its neats and scruffies. The original move to treat the world as its own representation, to strongly couple an intelligent system to the particularities of physical interaction, can be seen as a scruffy maneuver. But neat work in this area then re-abstracts from the (intrinsically messy) particularities of physical interaction to study the properties of behaviors in themselves, for example, to study the composability of behaviors through formal behavior description languages (e.g. [Safiotti, Konolige & Ruspini 1995]).

While my own research (and life) approach is scruffy, I am not trying to argue that only scruffy approaches have value and that neat approaches are intrinsically wrong-headed. I agree with Abelson that progress is best made by somehow fusing aspects of the scruffy and neat styles. For instance, *Terminal Time* makes use of formal reasoning techniques, specifically a form of higher-order logic programming (an implementation of [Elliott & Pfenning 1991]) (neat), to engage in ideologically biased reasoning about historical events (scruffy), where historical events are represented as collections of higher-order predicate calculus assertions (neat) written in an ontology based on the Upper Cyc Ontology [Lenat 1995] (Cyc being a scruffy instantiation of the neat logicist common sense reasoning agenda), all within the context of an audience-interactive exploration of ideological bias in representations of history (scruffy). Scruffy and neat tendencies coexist and interact in complex ways within AI, within the body of work of a single individual, and sometimes within an individual system/artwork.

Alternative AI

Table 4-1 provides a schematic description of the distinctions between classical and interactionist AI. But in arguments within and about the culture of AI, interactionist AI is sometimes lumped in the larger category of alternative AI, where alternative AI is also contrasted with classical AI (again with classical AI as the fall guy). The alternative AI camp includes such research agendas as neural networks, behavior-based (interactionist) AI, and approaches from artificial life such as genetic algorithms. The common thread running through these approaches is that they are all non-symbolic, all make use of some notion of emergence, and are all reactions to perceived failures in the classical AI tradition. However, these different approaches also make quite different methodological and theoretical commitments. Where the extreme forms of interactionist AI eschew any

form of representation, work in neural networks (NNs) explores distributed representation, while the performance of genetic search is highly dependent on the details of the genomic representation (learning performance for NNs is also highly dependent on the representation presented to the input layer). Work in both interactionist AI and NNs makes use of the notion of complex dynamical systems [Beer 1995; Smolensky 1998], viewing their respective systems primarily as complex physical systems rather than computational systems; the dynamical systems view is not as useful in genetic algorithms (GAs). While interactionist AI eschews search, GAs *are* a parallel hillclimbing search, and NNs make use of search during their training phase (searching the space of connection weights for a collection of weights that minimizes the error term).

Thus the camp of alternative AI is fragmented, with complex relationships among the various approaches within the alternative camp as well as with the concepts and commitments within classical AI. Any simple story that posits alternative AI as the hip, new response to bad, old classical AI prematurely closes off further inquiry into the rich complexities of the intellectual currents running through AI.

Learning vs. Authoring

The notion of intelligent machines that modify their own operations over time, that is, learning machines, has always been a part of AI work, with architectures and systems such as neural networks [Rosenblatt 1959], simulated evolution [Bledsoe 1961; Holland 1962], learning symbolic relational descriptions [Winston 1970; Vere 1975], and learning action policies from time-delayed evaluations [Samuel 1959] providing early examples. In recent years, machine learning has splintered off as its own subdiscipline concerned with making inferences from relatively unstructured (e.g. tuples) noisy data. Contemporary work focuses on three kinds of inferences: function approximation (classification and regression), building action policies for agents (reinforcement learning), and discovering structure in data (e.g. clustering, principle component analysis, etc.). Where much of classical AI studied complex reasoning in highly structured, noise free domains, machine learning focuses on simple reasoning in relatively unstructured, noisy domains.

One can make a distinction between systems that make heavy use of hand-authored knowledge (whether procedural or declarative), and systems that make heavy use of learning techniques. This distinction cuts across the other distinctions we've made so far. There are neat and scruffy versions of machine learning, and learning approaches situated within both interactionist (e.g. reinforcement learning) and classical (e.g. case-based reasoning) AI. Within machine learning the neats are currently in ascendancy, with purely statistical approaches, preferably approaches amenable to various convergence and bounds proofs, dominating.

Partisans of learning feel that only distinctions (representations, action policies) arising directly from data have value – systems employing a lot of hand authored knowledge are considered hacks that work only because of brute force knowledge engineering. Partisans of hand-authored knowledge argue that the process of authoring knowledge provides a valuable analysis of the structure of and interrelationships between the concepts in question; where the distinctions learned by statistical learning algorithms are often opaque to human observers. Further, there are no purely statistical learning approaches for many knowledge rich tasks, such as story generation (though a scruffy, classical learning approach may work, e.g. [Turner 1991]).

An AI-based art practice may make use of both learning and knowledge-rich approaches. In the context of interactive drama, behavior authoring for characters is

preferred, since the author wants tight control over the way the personality is expressed through all the behaviors. Thus characters in *Façade* are authored in a reactive planning language, ABL. *Office Plant #1*, on the other hand, makes use of statistically trained text classifiers to classify incoming email. Where we as human authors know the distinctions we want the classifiers to make, for example chatty vs. non-chatty email, and can recognize a chatty email when we see one, we don't have ready access to the knowledge we employ to make this distinction. By tuning the parameters of a generative model, a statistical text classifier can learn to make the distinction on its own (within the limits of the model the classifier starts with, typically for text learning a bag-of-words model). However, *Office Plant #1* then makes use of authored knowledge (rules that map the output of the classifiers onto node activations in a fuzzy cognitive map, and hand-tuned weights within the fuzzy cognitive map) to take action in response to incoming email.

Emergent vs. Rule-Based Systems

Arguments for alternative AI often hinge on emergence; classical AI systems can only do exactly what has been put into them, with no room for generativity or surprise, while alternative AI systems exhibit emergence, generating "stuff" (behaviors, responses) that was not put into them. Classical AI systems are dismissed as merely following rules that have been written by a human programmer, and thus incapable of having properties such as creativity or flexibility. As Chalmers points out [Chalmers 1996: 329], such arguments are vague and underspecified. Neural networks follow rules for summing activation energies and adjusting connection weights, genetic algorithms follow rules for combining genomes and evaluating the resulting phenome, etc., so, though rule-following arguments implicitly identify rules with symbolic rules manipulating high-level conceptual representations, the demarcation between rule-following and merely lawful activity is rarely spelled out. But even without resolving the semantic difficulties in the definition of a rule and accepting the conflation of the multitudinous approaches in classical AI to an intuitive notion of high-level symbolic rule, the distinction between emergent systems and non-emergent classical systems is far from clear. To claim that a classical system exhibits no surprise, no generativity, no flexibility, is to claim that the classical system is transparent – that in some detailed way a human observer can readily determine what the system's response will be to some given input. But, as Weizenbaum argues in his well known critique of AI [Weizenbaum 1976], programs are not inherently transparent and comprehensible, they rather quite quickly become opaque and incomprehensible (Weizenbaum is writing in the context of research in classical AI, so this statement pertains to classical systems). Within Weizenbaum's critique (to which we will return on page 193), the problem with opaque programs is that, as a technophilic society computerizes more and more social functions, the society falls under the sway of a totalizing instrumental reason that even in principle isn't open to democratic debate, since it is no longer possible to have a detailed understanding of how these computerized social functions operate. In the context of our argument here, what is interesting is that "simple, rule-following" classical programs are *not* simple, unsurprising and transparent:

> The layman … believes that the very fact that a program runs on a computer guarantees that some programmer has formulated and understands ever detail of the process which it embodies. But his belief is contradicted by fact. A large program is, to use an analogy of which Minsky is also fond, an intricately connected network of courts of law, that is, of subroutines, to which evidence is transmitted by other

> subroutines… The verdicts rendered by these courts may, indeed, often do, involve decisions about what court has "jurisdiction" over the intermediate results then being manipulated. The programmer thus cannot even know the path of decision making within his own program, let alone what intermediate or final results it will produce. Program formulation is thus rather more like the creation of a bureaucracy… [Weizenbaum 1976:234]

AI researchers build systems not to prove that the system does what they say it will do, but indeed to find out *what it will do*; system building, and the surprise thereof, is the only way to understand the repercussions of an idea.

To push quite specifically on the "rule-based systems are rigid and simplistic" argument, consider Soar, an archetypal rule-based cognitive modeling system [Newell 1990]. A Soar program, despite the fact that its operations consist entirely of selecting and applying rules from a rule base, is not doomed to rigid regurgitation of the knowledge put into it. On the contrary, the whole point of an architecture like Soar is to explore radical conditionality. Programs written in standard imperative languages tend to sprinkle conditionality rather sparsely through programs, with if-statements and loops scattered around in sequential code. For Soar (and other rule-based systems), on the other hand, every step of execution is conditional, potentially having done something different if conditions had not been the same [Newell 1990:163-164]. With this massive conditionality, rules interact in complex ways: "Thus the issue for the standard computer is how to be interrupted, whereas the issue for Soar and Act* (and presumably for human cognition) is how to keep focused." [Newell, Rosenbloom & Laird 1989] Sengers' experience of re-implementing the Oz Woggles [Loyall & Bates 1993] in Soar bears this out – the Soar Woggles were in some ways *more* reactive than the Hap Woggles, with the difficulty being to maintain sequential activity [Sengers, personal communication].

The point here is not to argue *against* alternative approaches and *for* classical approaches, but merely to point out that the behavior of all kinds of systems quickly becomes surprising. While notions of generativity serve as a useful lens for analyzing different architectures and approaches, it is not a simple matter of using "emergence" as a divining rod for identifying favored approaches.

Artistic Practice Transforms AI.

Artistic practice is potentially concerned with a broader set of issues than the issues of agency that structure the technical interactionist vs. classical AI debate. Artistic practice also operates from a different set of goals and assumptions than those shared by both interactionist and classical AI researchers. Finally, there are numerous additional distinctions (of which this chapter offers just a few) that characterize the intellectual strands, tensions, and camps operating within the terrain of AI. Any discussion of AI that neatly divides the field into one (or a few) distinctions, while bringing one set of issues to the surface, also distorts and obscures vast tracts of the terrain, rendering the intuitions, approaches, debates, and history lying within these tracts unavailable as conceptual resources. Thus, despite the affinity between cultural theoretic critiques of Enlightenment rationality and the technical project of interactionist AI, we should be wary of any position, implicit or explicit, that holds that some particular technical school of thought within AI is peculiarly suited to artistic practice. AI-based art is not a subfield of AI, nor affiliated with any particular technical school within AI, nor an application of AI. Rather it is a new interdiscipline, a new stance or viewpoint, from which all of AI is transformed and reconstructed.

Authorship

AI has traditionally been engaged in the study of the possibilities and limitations inherent in the physical realization of intelligence [Agre 1997a]. The focus has been on understanding AI systems as independent entities, studying the patterns of computation and interactions with the world that the system exhibits in response to being given specific problems to solve or tasks to perform. Both classical and interactionist AI reify the notion of intelligence. That is, intelligence is viewed as an independently existing entity with certain essential properties. Classical AI assumes that intelligence is a property of symbol manipulation systems. Interactionist AI assumes that intelligence is a property of embodied interaction with a world. Both are concerned with building something that *is* intelligent; that unambiguously exhibits the essential properties of intelligence.

In Expressive AI the focus turns to *authorship*. The AI system becomes an artifact built by authors in order to communicate a constellation of ideas and experiences to an audience. If classical AI builds brains in vats, and interactionist AI builds embodied insects, then Expressive AI builds *cultural artifacts*. The concern is not with building something that *is* intelligent independent of any observer and her cultural context. Rather, the concern is with building an artifact that *seems* intelligent and/or alive, that participates in a specific cultural context in a manner that is perceived as intelligent and/or alive. Expressive AI views a system as a performance of the author's ideas. The system is both a messenger for and a message from the author.

Metaphors Structuring AI-based Art Practice

The concept of an AI system as communication and performance is depicted in Figure 4-1.

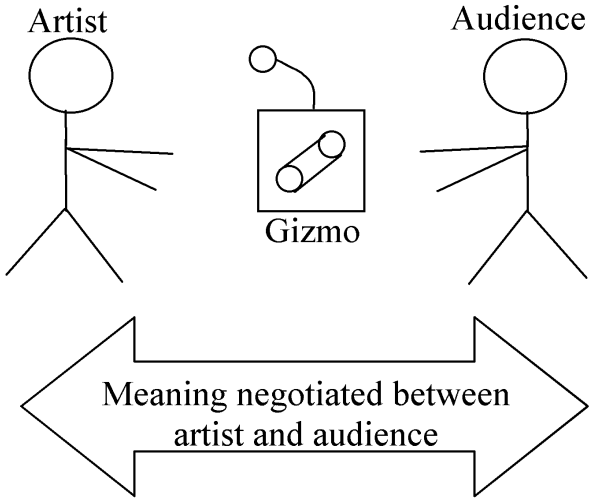

Figure 4-1. The conversation model of meaning making

The AI system (here labeled "gizmo") mediates between artist and audience. The gizmo structures the context within which the artist and audience negotiate meaning. The artist attempts to influence this negotiation by structuring the interpretive affordances of the gizmo, that is, by providing the audience with the resources necessary to make up a story

about what the gizmo is doing and what meanings the author may have intended to communicate. This relationship between gizmo, artist, and audience uses the conversation metaphor, where artistic practice is conceived of as a conversation between artist and audience mediated by the art "object" (the object can be something non-concrete, such as a performance).

The conversation metaphor is an example of what Agre [Agre 1997a] calls a theory-constitutive metaphor. Such a metaphor structures the theories and practices of a field. Every such metaphor has a center and a margin. The center is the set of issues brought into focus by the metaphor, those issues considered primary in the practice structured by the metaphor. The margin is the set of issues made peripheral by the metaphor, those issues playing only a secondary role in the practice, if considered at all. The practice may even assume that the margin will "take care of itself" in the process of focusing on the center.

The center of the conversation metaphor is the relationship between two subjects, the artist and the audience. A practice structured by this metaphor will focus on the negotiation of meaning between these two subjects. The margin is the internal structure of the gizmo itself. The conversation metaphor interprets the internal structure of the gizmo as an accidental byproduct of a focus on negotiated meaning; the structure "takes care of itself" in the process of focusing on the negotiation of meaning between artist and audience.

The central and marginal concerns of the conversation metaphor reverse those found in AI research practice (Figure 4-2).

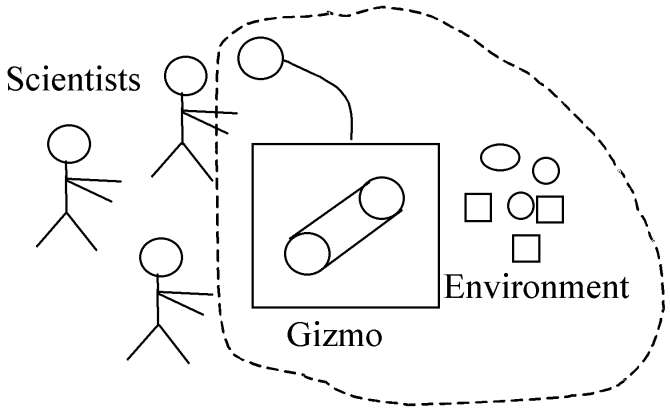

Figure 4-2. The construction model of AI research

AI research practice proceeds by means of the construction metaphor. The gizmo (in classical AI practice) or the gizmo + environment (in interactionist AI practice) is considered as a system complete unto itself, about which statements can be made without reference to the either the system builders or interpreters as subjects. Instead, system construction and interpretation is rendered as an objective process; construction is conditioned only by engineering concerns, and interpretation only by the requirements of empirical investigation. The active process of meaning making engaged in by a subject is marginalized.

Towards an Integrated Practice

In this chapter we've seen that there is no simple alignment between the needs of an art practice and any particular technical agenda within AI. AI does not spring univocally from some single cultural influence or master idea, but rather consists of complex flows of intuitions, approaches, debates and cultural influences. But there appear to be deep differences in the high-level goals and metaphors structuring AI and art. The challenge for the further development of Expressive AI is to find some way to unify these high-level agendas while remaining open to the full complexity of AI as a conceptual resource.

CHAPTER 5
A BEHAVIOR LANGUAGE

As described in Chapter 3, character behavior is written in the custom reactive planning language ABL (**A B**ehavior **L**anguage, pronounced "able"). ABL provides mechanisms supporting the real-time, moment-by-moment decision making necessary for animated believable agents. ABL is based on the Oz Project believable agent language Hap developed by A. B. Loyall, along with Bates and others in the Oz group [Loyall 1997; Bates, Loyall & Reilly 1992a; Bates, Loyall & Reilly 1992b; Loyall & Bates 1991]. The ABL compiler is written in Java and targets Java; the generated Java code is supported by the ABL runtime system.

ABL modifies Hap in a number of ways. The most significant addition in ABL is language support for multi-agent coordination. In *Façade*, character behavior is factored across beats and thus organized around dramatic activity. The support for multi-agent coordination allows the beat behaviors to directly describe Trip and Grace's joint dramatic activity. This chapter describes the ABL language, focusing on features of the language itself rather than on the expressive possibilities of these features. Chapter 6 discusses the expressive potential of ABL by describing the ABL idioms developed for *Façade*.

Hap

Since ABL re-implements and extends Hap, this section briefly describes the architecture of a Hap agent and the organization and semantics of the Hap language. The definitive reference on Hap is of course Loyall's dissertation [Loyall 1997]. Readers familiar with Hap should skip ahead to *Support for Joint Action* on page 73.

The architecture of a Hap/ABL agent appears in Figure 5-1. The agent has a library of pre-written behaviors. Each behavior consists of a set of steps, to be executed either sequentially or in parallel, which accomplish a goal. The current execution state of the agent is captured by the active behavior tree (ABT) and working memory. The ABT contains the currently active goals and behaviors. The ABT is a tree rather than a stack because some behaviors execute their steps in parallel, thus introducing parallel lines of expansion in the program state. The leaves of the ABT constitute the conflict set. The agent continuously executes a decision cycle, during which a leaf step is chosen for execution. As each step is executed, it either succeeds or fails. In a sequential behavior, step success makes the next step available for execution. If any step fails, it causes the enclosing behavior to fail. When the last step of a behavior succeeds, the enclosing behavior succeeds. In this way, success and failure propagate through the ABT.

The four basic step types are introduced below. For now, note that one of the step types is act, which performs a physical action with the agent's body, such as taking a step or grasping an object. The exact details of the execution of a physical action depend on both the agent's body and the world. For *Façade*, agents have virtual, animated bodies within a real-time, graphical, 3D story world.

Working memory contains any information the agent needs to keep track of during execution. This information is organized as a collection of working memory elements (WMEs). WMEs are like instances in an object-oriented language; every WME has a

type plus some number of typed fields that can take on values. WMEs are also the mechanism by which an agent becomes aware of sensed information. Sensors report information about changes in the world by writing that information into WMEs. Hap/ABL has a number of mechanisms for writing behaviors that are continuously reactive to the contents of working memory, and thus to sensed changes in the world. The details of sensors, like actions, depend on the specific world and agent body.

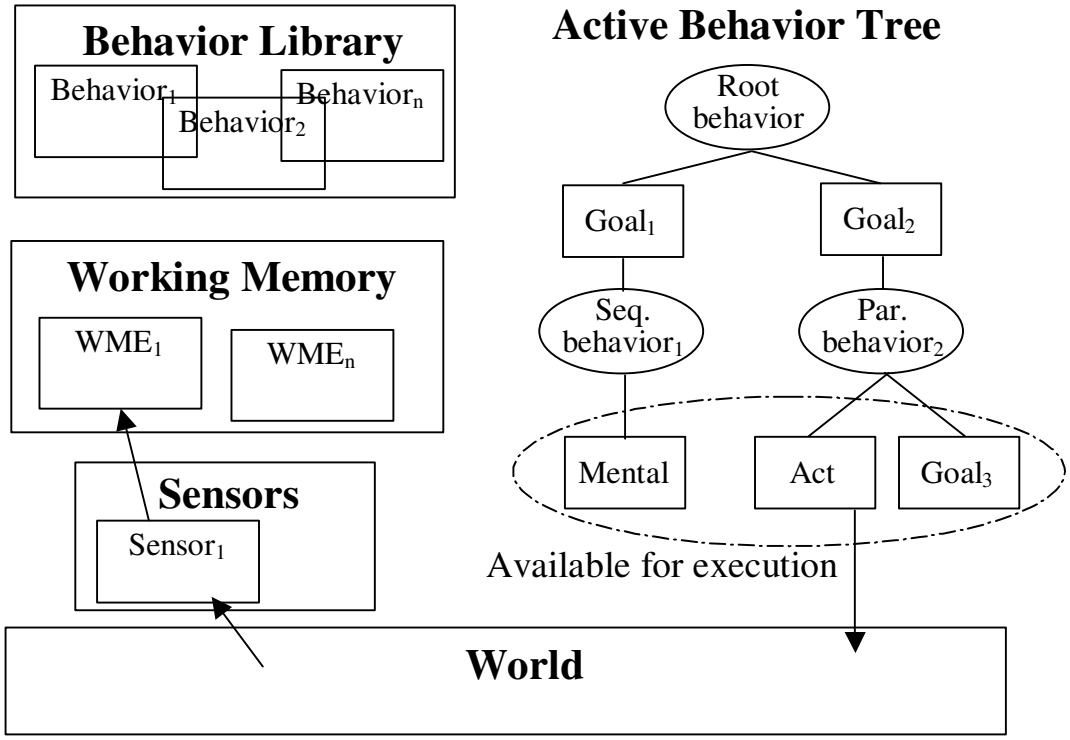

Figure 5-1. Architecture of a Hap/ABL agent

Example Behaviors

This section introduces Hap language features through a series of example behaviors. All examples use the Java-like ABL syntax.

An example sequential behavior appears in Figure 5-2.

```
sequential behavior AnswerTheDoor() {
    WME w;

    with success_test { w = (KnockWME) } wait;
    act sigh();
    subgoal OpenDoor();
    subgoal GreetGuest();
    mental_act { deleteWME(w); }
}
```

Figure 5-2. Example sequential behavior demonstrating the four basic step types

In this sequential behavior, an agent waits for someone to knock on a door, sighs, then opens the door and greets the guest. This behavior demonstrates the four basic step types,

namely wait, act, subgoal, and mental_act. Wait steps are never chosen for execution; a naked wait step in a sequential behavior would block the behavior from executing past the wait. However, when combined with a success test, a wait step can be used to make a demon that waits for a condition to become true. Success tests are continuously monitored conditions that, when they become true, cause their associated step to immediately succeed. Though in this example the success test is associated with a wait step to make a demon, it can be associated with any step type.

Success tests, as well as other tests to be described shortly, perform their test against working memory. In this example, the success test is looking for working memory elements (WMEs) of type KnockWME, which presumably is placed in the agent's working memory by a sensor when someone knocks on a door. Since there are no field constraints in the test, the test succeeds as soon as a KnockWME appears.

An act step tells the agent's body (sensory-motor system) to perform an action. For graphical environments such as *Façade*, physical acts will ultimately be translated into calls to the animation engine, though the details of this translation are hidden from the Hap/ABL program. In this example, the act makes the body sigh. Note that physical acts can fail – if the sensory-motor system determines that it is unable to carry out the action, the corresponding act step fails, causing the enclosing behavior to fail.

Subgoal steps establish goals that must be accomplished in order to accomplish the behavior. The pursuit of a subgoal within a behavior recursively results in the selection of a behavior to accomplish the subgoal.

Mental acts are used to perform bits of pure computation, such as mathematical computations or modifications to working memory. In the final step of the example, the mental_act deletes the KnockWME (by making a call to a method defined on ABL agents), since the knocking has now been dealt with. In ABL, mental acts are written in Java.

The next example, in Figure 5-3, demonstrates how Hap/ABL selects a behavior to accomplish a subgoal through signature matching and precondition satisfaction.

```
sequential behavior OpenDoor() {
   precondition { (KnockWME doorID :: door)
        (PosWME spriteID == door  pos :: doorPos)
        (PosWME spriteID == me  pos :: myPos)
        (Util.computeDistance(doorPos, myPos) > 100) }
   specificity 2;
   // Too far to walk, yell for knocker to come in
   subgoal YellAndWaitForGuestToEnter(doorID);
}

sequential behavior OpenDoor() {
   precondition { (KnockWME doorID :: door) }
   specificity 1;
   // Default behavior - walk to door and open
}
```

Figure 5-3. Example behaviors demonstrating preconditions

In this example, there are two sequential behaviors with the signature OpenDoor(), either of which could potentially be used to satisfy the goal OpenDoor(). The first behavior opens the door by yelling for the guest to come in and waiting for them to open the door. The second behavior (details elided) opens the door by walking to the door and opening it. When AnswerTheDoor() pursues the subgoal OpenDoor(), Hap/ABL

determines, based on signature matching, that there are two behaviors that could possibly open the door. The precondition of both behaviors is executed. In the event that only one of the preconditions is satisfied, that behavior is chosen to accomplish the subgoal. In the event that both preconditions are satisfied, the behavior with the highest specificity is chosen. If there are multiple satisfied behaviors with highest specificity, one is chosen at random. In this example, the first OpenDoor() behavior is chosen if the lazy agent is too far from the door to walk there ("too far" is arbitrarily represented as a distance > "100").

The precondition demonstrates the testing of the fields of a WME. The :: operator assigns the value of the named WME field on the left of the operator to the variable on the right.[7] This can be used both to grab values from working memory that are then used in the body of the behavior, and to chain constraints through the WME test.

The last example, in Figure 5-4, demonstrates parallel behaviors and context conditions.

```
parallel behavior YellAndWaitForGuestToEnter(int doorID) {
   precondition { (CurrentTimeWME t :: startT) }
   context_condition { (CurrentTimeWME t <= startT + 10000) }
   number_needed_for_success 1;

   with success_test { (DoorOpenWME door == doorID) } wait;
   with (persistent) subgoal YellForGuest(doorID);
}
```

Figure 5-4. Example parallel behavior demonstrating context conditions

In a parallel behavior, the steps are pursued simultaneously. YellAndWaitForGuestToEnter(int) simultaneously yells "come in" towards the door (the door specified by the integer parameter) and waits to actually see the door open. The persistent modifier on the YellForGuest(int) subgoal makes the subgoal be repeatedly pursued, regardless of whether the subgoal succeeds or fails (one would imagine that the behavior that does the yelling always succeeds). The number_needed_for_success annotation (only usable on parallel behaviors) specifies that only one step has to succeed in order for the behavior to succeed. In this case, that one step would be the demon step waiting for the door to actually open. The context condition is a continuously monitored condition that must remain true during the execution of a behavior. If the context condition fails during execution, then the behavior immediately fails. In this example, the context condition tests the current time, measured in milliseconds, against the time at which the behavior started. If after 10 seconds the door hasn't yet opened (the guest isn't coming in), then the context condition will cause the behavior to fail.

As failure propagates upwards through the subgoal chain, it will cause the first OpenDoor() behavior to fail, and eventually reach the OpenDoor() subgoal in AnswerTheDoor(). The subgoal will then note that there is another OpenDoor() behavior that has not yet been tried and whose precondition is satisfied; this behavior will be chosen in an attempt to satisfy the subgoal. So if the guest doesn't enter when the agent yells for awhile, the agent will then walk over to the door and open it.

[7] In ABL, a locally-scoped appropriately typed variable is automatically declared if it is assigned to in a WME test and has not been previously explicitly declared.

Step and Behavior Annotations

Much of the expressive power of Hap results from a rich set of step and behavior annotations that modify step and behavior processing. ABL introduces new annotations that build on the Hap annotations. The behavior idioms developed for *Façade*, described in Chapter 6, make use of both the original Hap annotations and the ABL annotations. For these reasons I briefly describe the Hap annotations here.

Success Test

The success_test annotation associates a continuously monitored test with a step. If the test becomes true during the execution of the step, the step immediately succeeds. Success tests are often coupled with wait steps in sequential behaviors in order to create demons that trigger on events, such as the behavior AnswerTheDoor() in Figure 5-2. It is also often useful to associate success tests with subgoals; this can be used to encode the behavior that if, while the subgoal is executing, the goal is spontaneously accomplished, the subgoal should immediately succeed.

Priority

In every decision cycle there are, in general, multiple steps in the conflict set to choose among in selecting the next step for execution. Associated with every step is a priority. When selecting a step from the conflict set, Hap/ABL prefers higher priority steps to lower priority steps. The default step priority is 0. Step priorities can be set with two annotations, priority and priority_modifier. Priority sets the absolute priority of a step. When used to set the priority of a subgoal, all steps in the subtree expanded at the subgoal inherit the priority. Priority_modifier specifies an increment or decrement for setting the priority of a step relative to the priority inherited from its enclosing behavior (which in turn is inherited from the priority of the parent subgoal). Programming in Hap/ABL is parallel programming, with all the resulting complexities, such as the unexpected side effects of parallel processes mixing, race conditions, etc. The priority annotations are one mechanism for controlling the mixing of parallel lines of expansion.

Persistence

Three related step annotations, persistent, persistent when_succeeds, and persistent when_fails, change what happens when a step succeeds or fails. When a step is marked persistent, it is not removed from the ABT when it succeeds or fails. Instead the step returns to the conflict set and is again available for execution. The when_succeeds and when_fails versions of the persistent annotation affect step processing similarly, but only when the step succeeds or fails respectively. Persistence annotations are useful for coding iteration. Additionally, long term character motivations can be captured as persistent high-level goals.

Ignore Failure

When a step fails, failure propagates up the ABT, causing the enclosing behavior to fail, the parent subgoal to look for another behavior to accomplish the goal (and potentially fail if there isn't one), etc. The ignore_failure annotation causes failure to be treated as success. Ignore_failure is typically used with subgoal steps to attempt the subgoal without requiring that it succeed.

Effect Only

The effect_only annotation is used to create truly optional steps. It can only be used meaningfully in parallel or collection[8] behaviors. When all the non-effect_only steps succeed, the behavior succeeds. The difference between ignore_failure and effect_only is that ignore_failure steps must still be executed to completion before the parent behavior can succeed; it's just that the ignore_failure step can never fail. An effect_only step not only doesn't have to complete for the enclosing behavior to succeed, it doesn't even have to be attempted. If, by chance, all the non-effect_only steps are chosen for execution before the effect_only step, then the behavior succeeds without ever trying the effect_only step. If the effect_only step is in the middle of executing (say, an expanded subgoal that has an executing subtree rooted at the goal) and all the non-effect_only steps succeed, the effect_only step is immediately aborted and the behavior succeeds. Effect_only is useful for creating purely optional parts of behaviors, which happen or not by chance, or parts of behaviors that should only execute for as long as the rest of the enclosing behavior executes.

Specificity

The behavior annotation specificity modifies behavior selection when executing a goal. A behavior is normally chosen by first using signature matching to locate a subset of behaviors from the behavior library that could potentially accomplish the goal, testing the preconditions of this subset of behaviors, and randomly selecting a behavior from those with satisfied preconditions. A specificity annotation associates an integer specificity with a behavior, as can be seen in the OpenDoor() behaviors in Figure 5-3. In the event that multiple matching behaviors have satisfied preconditions, the behavior with the highest specificity is chosen. The term "specificity" refers to the notion that, given two different satisfied preconditions, one of the preconditions is satisfied in a larger subset of all possible working memory states, and is thus more general, while the other is satisfied by a smaller subset of working memory states, and is thus more specific. By allowing authors to associate specificities with behaviors, Hap/ABL implements the heuristic of first trying the behavior whose precondition is most specific to the current state the agent finds itself in.

Number Needed for Success

Normally all the non-effect_only steps of a behavior must succeed for the behavior to succeed. Sometimes however, in a parallel or collection behavior, only a subset of the steps need to succeed, without knowing ahead of time which subset is necessary (if you knew ahead of time which subset actually needed to succeed, the other steps could be marked effect_only). For example, suppose two subgoals are being pursued in parallel, but once either one of them succeeds, the enclosing behavior should succeed (and thus abort the unfinished subgoal). The behavior annotation number_needed_for_success expresses this. It is used to specify the number of non-effect_only steps that must succeed in order for the behavior to succeed.

[8] A collection behavior is a variety of parallel behavior in which every step need only be *attempted* for the behavior to succeed. A collection behavior is thus the same as a parallel behavior with every step annotated with `ignore_failure`.

Breed Goal

Normally subgoals are rooted at the behavior containing the subgoal step. It is sometimes useful to spawn a goal to another part of the ABT – this allows the goal to continue to be pursued when the original subgoaling behavior disappears. In Hap, this functionality is provided by the breed_goal step. In ABL this is called spawngoal. In ABL, the spawngoal step is part of the support for reflection (described starting on page 93). By default goals are spawned at the ABT root, though they can be spawned at specific parallel behaviors. Goal spawning is used by the ABL idioms described in Chapter 6.

Conflicts

Because of the existence of parallel and collection behaviors, a Hap/ABL agent is usually pursuing many simultaneous activities. But not all activities can necessarily mix; some may conflict with each other. Physical acts may conflict because they make use of the same body resources, that is, the same sets of "muscles". For example, in a humanoid agent, an act that grasps an object with the right arm conflicts with an act that releases an object with the right arm. The body can't grasp and release an object simultaneously. Physical act conflicts are determined by the details of the sensory-motor system and world to which a Hap/ABL agent is connected. More abstractly, goals may conceptually conflict. For example, the goal to sleep and the goal to play conflict with each other. Even though it may be possible to interleave the physical actions of sleeping and playing, it doesn't make sense conceptually.

Hap allows authors to specify goal and act conflicts. When a step (goal or act) is executed, Hap checks to see if any conflicting steps are already executing. If there are conflicting steps executing, then the highest priority step wins; the lower priority step is suspended until the higher priority step completes. In the event of a priority tie, the already executing step wins. When an act is suspended it is aborted if it was in the middle of executing (doing something with the body) and removed from the conflict set. It is added back to the conflict set when all higher or same priority conflicting acts are removed from the ABT. When a goal is suspended, any executing acts within the subtree rooted at the goal are aborted, and all leaf steps of the subtree rooted at the goal are removed from the conflict set. These leaf steps are added back to the conflict set when all higher or same priority conflicting goals are removed from the ABT.

Decision Cycle

The Hap decision cycle appears in Figure 5-5 [Loyall 1997:57]. The ABL decision cycle, while substantially similar, has a number of changes including support for joint goals. The ABL decision cycle is described later.

1. If needed, adjust the ABT for actions that have finished.
2. Else if needed adjust the ABT based on changes in the world.
3. Else if needed adjust suspended goals.
4. Else pick a leaf step to execute, and execute it.

Figure 5-5. Hap decision cycle

Physical acts (act steps) take time to execute. While the physical act executes in the world (e.g. an arm moves or a leg takes a step), the act step is marked as executing and blocks the current line of expansion in the ABT. The first step of the decision cycle checks whether any currently executing physical acts have succeeded or failed. The effect of the success or failure of the act is propagated in the ABT. This includes possibly succeeding or failing the enclosing behavior (and all the ABT changes cascading from that, including unsuspending currently suspended goals and steps).

The second step of the decision cycle checks all continuously monitored conditions, that is, context conditions and success tests, and succeeds steps whose success tests evaluate to true and fails behaviors whose context conditions evaluate to false.

The third step of the decision cycle adjusts the suspended status of goals based on the authorially specified goal and act conflicts.[9]

The last step of the decision cycle picks a step from the conflict set (the leaves of the ABT) to execute. If there are multiple steps in the conflict set, the highest priority step is selected. If multiple steps share the highest priority, then a step in the current line of expansion, that is, a step on the most recently pursued path from root to leaf, is selected. This "prefer current line of expansion" heuristic helps to prevent Hap from thrashing by needlessly switching between multiple active behaviors. Finally, if there are still multiple steps to choose from, one is selected at random.

Note that parallelism in a Hap program is introduced by cooperative multitasking. Steps from a lower priority (or same priority) behavior can opportunistically mix with steps from a higher priority behavior when the higher priority behavior blocks on the execution of a physical act or wait step (the wait step presumably has a success test). While the higher priority behavior is blocked, the decision cycle can choose lower priority steps. If a Hap program consisted of nothing but subgoals and mental acts, and thus behaviors never blocked, the priority and current line of expansion mechanisms would serialize behavior execution.

Em

Em, a goal appraisal model of emotion [Neal Reilly 1996; Bates, Loyall & Reilly 1992a; Bates, Loyall & Reilly 1992b; Bates 1994], is integrated with Hap. Em maintains an emotion state, which includes a hierarchy of active emotions and the possibility of simultaneous conflicting emotions, as a byproduct of goal processing. A rich set of goal annotations can be used to customize the emotions generated as goals are attempted, succeed and fail. Emotion state can be queried within preconditions, context conditions and success tests. Behavior processing produces emotion and emotion modulates behavior processing. Sadly, Em is not currently re-implemented in ABL. This should not be read as an implicit commentary on Em. In my work on *Subjective Avatars* [Mateas 1997; Mateas 1998], I found Em to be a powerful resource. *Façade's* simple mood system, implemented in ABL, is briefly described in Chapter 6 on page 117.

Support for Joint Action

ABL's most significant contribution is to extend the semantics of Hap to support the coordination of multiple characters, through the addition of *joint goals and behaviors.*

[9] The ABL decision cycle doesn't include a distinct step for processing conflicts. Rather, suspension and unsuspension are processed as steps execute, succeed and fail.

The driving design goal of joint behaviors is to combine the rich semantics for individual expressive behavior offered by Hap with support for the automatic synchronization of behavior across multiple agents.

Introduction to Joint Goals and Behaviors

In ABL, the basic unit of coordination is the joint goal. When a goal is marked as joint, ABL enforces synchronized entry into and exit from the behaviors chosen to accomplish the goal. The keyword joint can be used to modify both goals and behaviors. The joint declaration tells ABL that entry into and exit from the joint behavior must be coordinated with team members. *Entry* into a behavior occurs when the behavior is chosen to satisfy a subgoal. *Exit* from the behavior occurs when the behavior succeeds, fails, or is suspended. The algorithm for executing a joint subgoal and coordinating entry appears in Figure 5-6.

When ABL executes a joint goal, a behavior is chosen for the goal using normal Hap behavior selection methods, with the additional constraint that the behavior must be joint (marked with the joint keyword).

Joint behaviors include a specification of the team members who must participate in the behavior. If a joint behavior is found for the joint goal, ABL marks the goal as negotiating and begins negotiating entry with team members specified in the joint behavior. Note that *negotiating* is a new state for joint goal steps. A normal (non-joint) goal can either be *not executing*, in which case it is a leaf step available for execution, *executing*, in which case a behavior has been chosen for the goal and added to the ABT, or *suspended*, in which case it is removed from the conflict set if it was not previously executed, or all leaf steps in the subtree rooted at the goal are removed from the conflict set if it was previously executed. In the *negotiating* state, the goal step is removed from the conflict set. That line of expansion is blocked until negotiation completes, but all other parallel lines of expansion are still pursued. If the negotiation takes awhile, perhaps because there are a large number of distributed teammates who are synchronizing during the negotiation, all negotiating agents continue to execute the decision cycle and engage in behavior. An intention-to-enter message is sent to all team members. The initiating message includes information about the goal signature and arguments.

1. The initiating agent chooses a *joint* behavior for the joint goal based on signature matching, precondition satisfaction, and specificities.
2. If a joint behavior is found for the joint goal, mark the goal as negotiating and broadcast an intention to enter the goal to all team members, otherwise fail the goal.
3. If all team members respond with an intention to enter the joint goal, add the joint behavior (and behavior children) to the ABT.
4. If any team member reports an intention to refuse entry to the joint goal, broadcast an intention to refuse entry and fail the behavior when all team members respond with an intention to refuse entry.

Figure 5-6. Agent initiating a joint behavior via joint subgoal execution

The goal remains in the *negotiating* state until all team members respond with an intention to enter or an intention to refuse entry. If all agents respond with intention-to-enter messages, this signals that all agents in the team have found appropriate behaviors in their local behavior libraries; the goal state is changed to *executing*, and the selected behavior and its steps are added to the ABT. If any agent responds with an intention to refuse entry, presumably because, given the goal signature and goal arguments, it couldn't find a satisfied joint behavior, the initiating agent sends all team members an intention to refuse entry. When all agents report that they intend to refuse entry, the initiating agent fails the joint behavior (whose steps never actually got a chance to execute). This causes the goal to attempt to find a different joint behavior with satisfied precondition, perhaps one with a different set of team members. Just as with a normal (non-joint) goal, if no such alternate behavior can be found, the goal fails.

Figure 5-6 shows the entry negotiation algorithm for the *initiator* of a joint goal, that is, the agent who originally executes the joint goal step, and who thus begins the joint behavior selection and negotiation process. The teammates of a joint goal initiator use a similar negotiation algorithm. The only difference is that for non-initiators, a joint goal with appropriate signature and arguments must be created and attached to the root collection behavior of the ABT.

The algorithm of coordinating exit from a joint behavior is shown in Figure 5-7.

1. An initiating agent broadcasts to all team members an intention to exit (either succeed, fail, or suspend) an executing joint goal.
2. All agents receiving an intention to exit respond by broadcasting to all team members their own intention to exit (succeed, fail, or suspend).
3. When all team members respond with the appropriate intention to exit, the joint goal is succeeded, failed or suspended as appropriate.

Figure 5-7. Agent exiting a joint behavior

For example, assume that a joint behavior has been successfully entered by a team. At this point each member of the team is executing a joint behavior from their local behavior library with the same signature, arguments, and team members. One of the team members, in executing their local joint behavior, encounters a condition where they should exit the behavior. Perhaps the last step of the behavior succeeds, causing the joint behavior and goal to succeed, or the context condition fails, causing the joint behavior and goal to fail, or a higher priority conflicting goal (either joint or non-joint) enters the ABT, causing the joint goal to suspend. The agent encountering this situation becomes the initiator of the intention to exit; this exit initiator is not necessarily the same agent as the entry initiator. The exit initiator marks the joint goal as negotiating and broadcasts the appropriate intention to exit to all team members. While the joint goal is in the negotiating state it blocks that line of expansion; all other lines of expansion in the ABT are still active.

As each team member receives an intention to exit, it marks its local version of the joint goal as negotiating and broadcasts an exit intention. Once exit intentions have been received from all team members, an agent exits the negotiating goal (succeeds, fails or suspends the goal).

Example of Basic Joint Goal and Behavior Support

This section provides a simple example, based on the Follow the Leader behavior of the Woggles [Loyall & Bates 1993], of the joint goal negotiation protocol in action. The Woggle world, an early demonstration system produced by the Oz Project, consists of a Dr. Seuss-like landscape inhabited by three Woggles - the shy Shrimp, the aggressive Wolf, and the friendly Bear (see Figure 5-8). As the Woggles play, fight and hang out with each other, the player is able to enter the Woggle world as a forth Woggle.

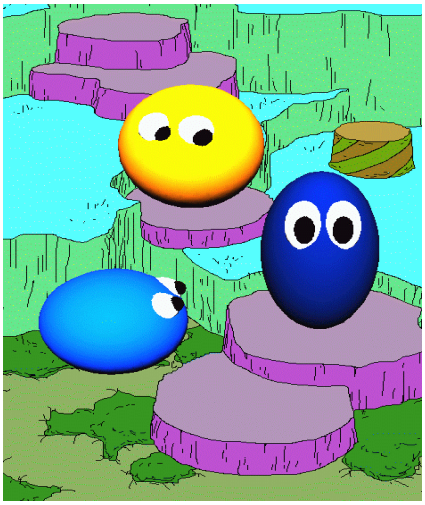

Figure 5-8. Close-up of three Woggles

The diagram in Figure 5-9shows the original behavior structure for a follower playing Follow the Leader.

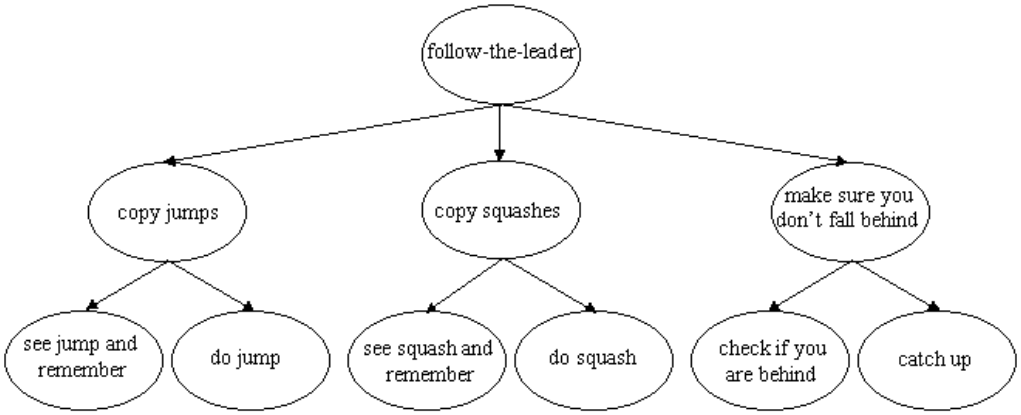

Figure 5-9. Original behavior structure for the follower

The behavior is decomposed into three sub-behaviors, one to copy the leader's jumps, one to copy the leader's squashes, and one to monitor whether the follower is falling too far behind the leader. Each of these behaviors is in turn decomposed into a sensing behavior that gathers information from the world (e.g. see jump, check if you are behind), and a behavior that acts on the sensed information (e.g. copy the jump recorded by the "see jump" behavior). Communication between behaviors takes place via memory elements posted to and matched from the agent's working memory.

The diagram in Figure 5-10 shows the original behavior structure for the leader playing Follow the Leader.

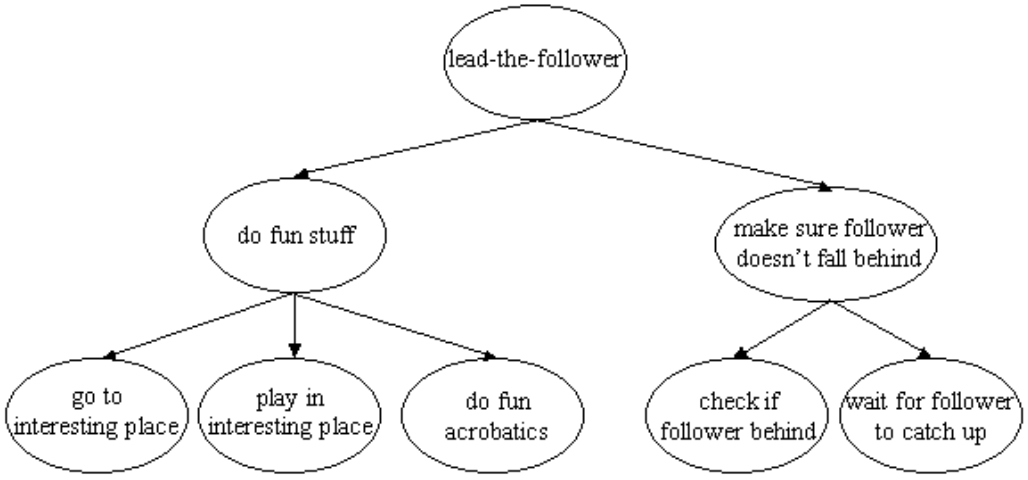

Figure 5-10. Original behavior structure for the leader

The top level behavior is decomposed into two sub-behaviors, one to do fun stuff (the hopping and squashing that the follower will copy) and one to monitor whether the follower has fallen behind. The "fun stuff" behavior is further decomposed into three different ways to have fun. The "make sure follower doesn't fall behind" behavior is decomposed into a sensing behavior that monitors the follower's activity, and a behavior that waits for the follower to catch up in the event that the follower did fall behind. Both Figure 5-9 and Figure 5-10 elide the sequential structure of the behaviors, showing only the persistent, parallel, subgoal structure. For example, the "lead-the-follower" behavior first chooses another Woggle to invite, moves over to the invitee, offers an invitation to play follow the leader (using Woggle body language), then, if the invitee signals that the invitation is accepted, starts the two parallel behaviors (fun stuff and monitor follower).

For two Woggles to play a game of follow the leader, one of the Woggles must first decide that it wants to be a leader and successfully invite the other Woggle to be a follower. The two Woggles then independently execute their respective behavior hierarchies. These two independent hierarchies coordinate via sensing, by mutually monitoring each other's physical activities. In addition to the Follow the Leader behavior hierarchy, both Woggles have a number of other behaviors executing in parallel. These behaviors are monitoring the world for certain actions, such as someone saying hi, a friend being attacked by someone else, someone inviting the Woggle to play a game, etc. If the follower pauses in the middle of the game to respond to one of these world events, perhaps suspending it's local Follow the Leader behavior hierarchy, the leader will experience this as the follower falling behind. If the follower takes too long to get back to the game, the leader will "time out" and the lead-the-follower behavior will fail (stop executing with failure). The leader will then start doing something else. However, unless similar timeouts have been placed in the right behaviors in the follower, the follower, after completing the interruption, will unsuspend and continue playing follow the leader. In fact, the original Woggle code *does not* have the appropriate timeouts in the follower, and so this condition can happen. So now the former leader is jumping around the world doing its own thing while the follower dutifully follows behind copying the leader's actions; the leader is not aware that the follower's actions are in any way related to the

leader, and the follower has no idea that the leader is no longer playing Follow the Leader. This is one example of the coordination failures that can happen even in a rather simple joint action when the joint activity is produced through the ad hoc synchronization of independent behavior hierarchies.

Using ABL's joint behaviors, the top of the leader's and follower's Follow the Leader behavior hierarchies are shown in Figure 5-11.

```
// Leader's version of FollowTheLeader
joint parallel behavior FollowTheLeader {
    teammembers Shrimp, Bear;
    precondition { <I'm a leader and feel like leading> }

    subgoal DoFunStuff();
    subgoal MakeSureFollowerDoesntFallBehind();
}

// Follower's version of FollowTheLeader
joint parallel behavior FollowTheLeader {
    teammembers Shrimp, Bear;
    precondition { <I'm a follower and feel like playing> }

    subgoal CopyJumps();
    subgoal CopySquashes();
    subgoal MakeSureYouDontFallBehind();
}
```

Figure 5-11. Joint behaviors for Follow the Leader

To simplify the example, consider just two of the Woggles, Shrimp and Bear. Since either can be a leader or follower, both Woggles have both the leader and follower versions of the behavior in their behavior libraries. One of them, say Bear, decides to play follow the leader – this decision is made by logic in some other behavior, perhaps a high level motivational behavior, resulting in the creation of a WME, LeaderWME, indicating that Bear wants to lead a game of follow the leader, a body language request to Shrimp to play, and the execution of joint subgoal FollowTheLeader().

The execution of the joint subgoal results in ABL trying to find a satisfied joint behavior to accomplish the goal. The preconditions distinguish between the leader and follower cases. If the behaviors didn't have preconditions testing LeaderWME, then the initiator of FollowTheLeader() might inadvertently select the follower version of the behavior. Sensing could also be used to distinguish the two cases, selecting the leader version if a body language request to play from another Woggle has not been recently seen, and the follower version if it has. Once Bear has selected the leader version of joint behavior FollowTheLeader(), the subgoal FollowTheLeader() is marked as negotiating and a request-to-enter is sent to Shrimp. Shrimp creates a joint subgoal FollowTheLeader() at the root of his ABT and selects the follower version of the behavior from his joint behavior library, again using preconditions to distinguish cases. Note that the preconditions can also be used to add personality-specific tests as to whether the Woggle feels like playing follow the leader. In Shrimp's case, for example, Shrimp may only feel like playing if he hasn't recently been picked on by another Woggle. Assuming that Shrimp's precondition is satisfied, Shrimp sends an intention-to-enter to Bear. Both Shrimp and Bear have received intentions-to-enter from all team members, so they each

add their respective selected behaviors to the ABT; they are now both playing follow the leader.

Once synchronized entry into FollowTheLeader() is established, they continue playing follow the leader until one of them exits the behavior. Perhaps Wolf threatens Shrimp in the middle of the game, causing Shrimp to engage in high priority fear reaction that suspends his local FollowTheLeader() goal. The goal is marked as negotiating and an intention to suspend is sent to Bear. Bear marks his goal as negotiating and sends an intention to suspend to Shrimp. They have both received intentions to suspend from all team members, so they each locally suspend their FollowTheLeader() goal. Similar exit negotiations ensure synchronization on goal success and failure. Every team member is guaranteed that if it is locally executing the joint goal FollowTheLeader(), all team members are executing the joint goal FollowTheLeader().

Joint goals and behaviors thus synchronize behavior execution across agents; the entry into a joint behavior is precisely a synchronization point. Joint and individual behaviors can be nested arbitrarily within the behavior hierarchy, depending on the granularity of the synchronization required. In the simple joint behaviors in Figure 5-11, only the FollowTheLeader() behavior is synchronized. However, smaller granularity behaviors could be synchronized. For example, jump and squash could be implemented as joint behaviors within the follow the leader behavior hierarchy. When a joint jump is entered, it would synchronize the leader and followers for the specific act of jumping. In essence, this would establish an automatic pattern of communication between the Woggles saying "now the leader is going to do a jump and all the followers should copy it". In addition, an agent can be committed to multiple simultaneous joint goals with different team members. For example, Shrimp could be a follower committed to a FollowTheLeader() goal with Bear while simultaneously committed to a Hassle() goal with Wolf (a goal coordinating Wolf and Shrimp in Wolf hassling Shrimp). As the two behaviors mixed together, Shrimp would keep a wary eye on Wolf, occasionally slinking or groveling, while trying to keep up in follow the leader with Bear.

Additional Annotations for Joint Goals and Behaviors

This section describes several step annotations that increase the expressive power of ABL with respect to joint goals.

Team Success

When the local joint goal for one team member succeeds, that team member initiates success negotiation for the joint goal. What should the rest of the team members do? On receipt of the initiator's intention to succeed, should they immediately succeed their local joint goal, thus aborting their still executing local joint behavior, or should they wait to signal that they intend to succeed until their local joint behavior and goal succeed? This choice can be controlled by the goal annotations team_needed_for_success and one_needed_for_success. If a joint goal has been initiated with team_needed_for_success, all team members' local copies of the joint goal must succeed before the team can succeed. Team members that are still executing their local copy of the joint behavior keep track of who they have received intentions to succeed from, but wait to send their own intention to succeed until their local behavior succeeds. The goals for waiting team members remain in the negotiating state, blocking that particular line of expansion, until all team members have succeeded and sent an intention to succeed. If a joint goal has been initiated with one_needed_for_success, then the first

team member to succeed will cause the rest of the team to succeed. Team members that are still executing their local copy of the joint behavior immediately succeed their joint goal on receipt of an intention to succeed and signal their own intention to succeed. This of course removes the subtree rooted at the joint goal, including the joint behavior, and aborts any executing steps within the subtree. As an example, if we wanted the FollowTheLeader() behavior above to succeed as soon as any one team member succeeded, the initiator's joint subgoal would read:
with (one_needed_for_success) joint subgoal FollowTheLeader(). These step annotations can only be legally used on joint subgoals.

An ABL program can set the default negotiation strategy to use in the absence of an explicit annotation with the compiler pragma joint_goal_success_negotiation.

Team Effect Only

The team_effect_only step annotation indicates that a step should be treated as effect_only with respect to the specific local copy of the joint behavior, but should continue to execute until the parent joint goal of the behavior negotiates success across the whole team. Thus team_effect_only has a combined effect: treat the step as optional for local behavior success, but continue to execute the step (or leave it available for execution in the conflict set if it isn't yet executing) while the parent joint goal is in the negotiating state. Team_effect_only can only be meaningfully used on steps of a parallel joint behavior.

Full Complexity of the Joint Negotiation Protocol

So far we've only been discussing a simplified version of the negotiation protocol. This section describes the protocol in its full complexity, and gives examples of the edge cases and race conditions that the protocol successfully handles. The bottom line of this section is this: joint goals and behaviors guarantee synchronized entry and exit across arbitrarily sized teams consisting of fully asynchronous, arbitrarily scheduled team members.

Asynchronous Agents and World

Both the text world and real-time graphical world versions of Hap scheduled the agents and world using a top-level loop as shown in Figure 5-12. Many games employ a similar top-level loop.

1. Lock down any sensed values (e.g. agent positions) in the world. All sensed values are determined by the most recent render.
2. Each agent, in some order, is a given a time slice in which to execute. The agent executes as many steps as it can in the time slice.
3. The world is given a time slice in which to update and render.

Figure 5-12. Top-level loop scheduling Hap agents and world

ABL makes no assumptions about the scheduling of agents and the world. The agents and world run completely asynchronously. In the case of a simulated world with multiple agents all running on the same machine, how the execution of the agents and world interleave depends on the vagaries of the process and thread scheduler. In the case of a

distributed world, in which multiple agents running on different machines connect to a shared world, perhaps a massively multiplayer world such as *Everquest* or *Star Wars: Galaxies*, agents and the world interleave according to the vagaries of the different speeds and loads on individual machines as well the details of message propagation in the communication infrastructure. And in the case of multiple physical agents (robots) executing in the real world, in which the top level loop in Figure 5-12 can only be established as a special case[10], agents and the world interleave according to the possibly varying execution hardware of the different agents and the dynamics of the agents and world. ABL is designed to be readily connectable to multiple agent bodies and worlds, including virtual, distributed agents and robots. The team negotiation protocol must make no assumptions about negotiation messages being processed in a fixed order across team members and must be robust to a variety of potential race conditions.

Race Conditions

This section describes how the negotiation protocol resolves potentially inconsistent team state arising from race conditions arising during negotiation.

Consider the first case of three team members, A, B and C, who are negotiating exit from a joint goal. At around the same time, A suspends the joint goal because a higher priority conflicting goal enters its ABT, B fails the goal, perhaps because a context condition on its local version of the joint behavior failed and no alternative joint behavior with the same team members is available for the joint goal, and C succeeds the goal, perhaps because its local version of the joint behavior successfully completed all of its steps. At this point each of the team members has a different intention with respect to the goal. Each team member broadcasts its conflicting intention to the rest of the team members. This situation is depicted in Figure 5-13, where each team member is shown with its current intention, its local copy of the commit set (the set of agents with the same intention), and the intentions being broadcast to the other agents. The three agent's intentions with the respect to the joint goal are maximally conflicting; each agent has a different intention, and each agent is receiving two intentions that differ both from the intention the agent already holds and from each other.

[10] For example, in the robotic-soccer architecture in [Veloso, Stone & Han 1998], perception is centralized via a single top-view camera looking down on the field, with individual robot control programs running on the same machine and sharing the global perceptual information.

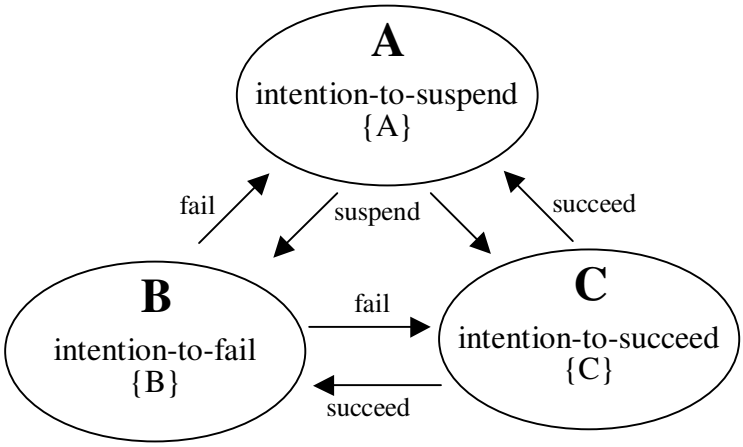

Figure 5-13. Team in a maximum negotiation conflict

ABL achieves a consistent negotiation outcome in such cases by maintaining a total precedence order over intentions. This precedence order is shown in Figure 5-14.

1.	Fail	highest precedence
2.	Succeed	
3.	Remove	
4.	Suspend	
5.	Unsuspend	lowest precedence

Figure 5-14. Precedence order on negotiation intentions

When an agent receives a higher precedence intention than the one it is currently holding, it switches to the higher precedence intention, resets its commit set to itself and the agent from whom it received a higher precedence intention, and broadcasts its new intention to team members. Agents ignore the receipt of intentions with lower precedence than the one it is currently holding.

An example sequence showing how intention precedence ordering results in the achievement of a consistent exiting state for a joint goal is shown in Figure 5-15. Assume that A receives C's intention to succeed before B's intention to fail. In Step 1, A has assumed an intention to succeed, placed itself and C in its commit set, and broadcast intentions to succeed. A's original intention to suspend messages are still in flight to B and C, B's intention to fail messages are still in flight, and one of C's intention to succeed messages is still in flight to B. Now suppose that B and C receives A's intentions to suspend. Since suspend is lower precedence than both succeed and fail, B and C both ignore the message. Further, B receives C's original intention to succeed; B ignores this message. Then C receives B's intention to fail. Since failure is higher precedence than success, C switches to intention to fail, resets its commit set to itself and B, and broadcasts intentions to fail. The current situation is depicted in Step 2. Finally, A's succeed messages are received and ignored by B and C, and B's fail message is received by A. A switches intentions for a second time to intention to fail, resets its commit set to

itself and B, and broadcasts intentions to fail. The situation at this point is depicted in Step 3. The only remaining messages in flight are intentions to fail. In a short time all three agents will receive the remaining messages, commit to failure, and fail the joint goal in each of their individual ABTs.

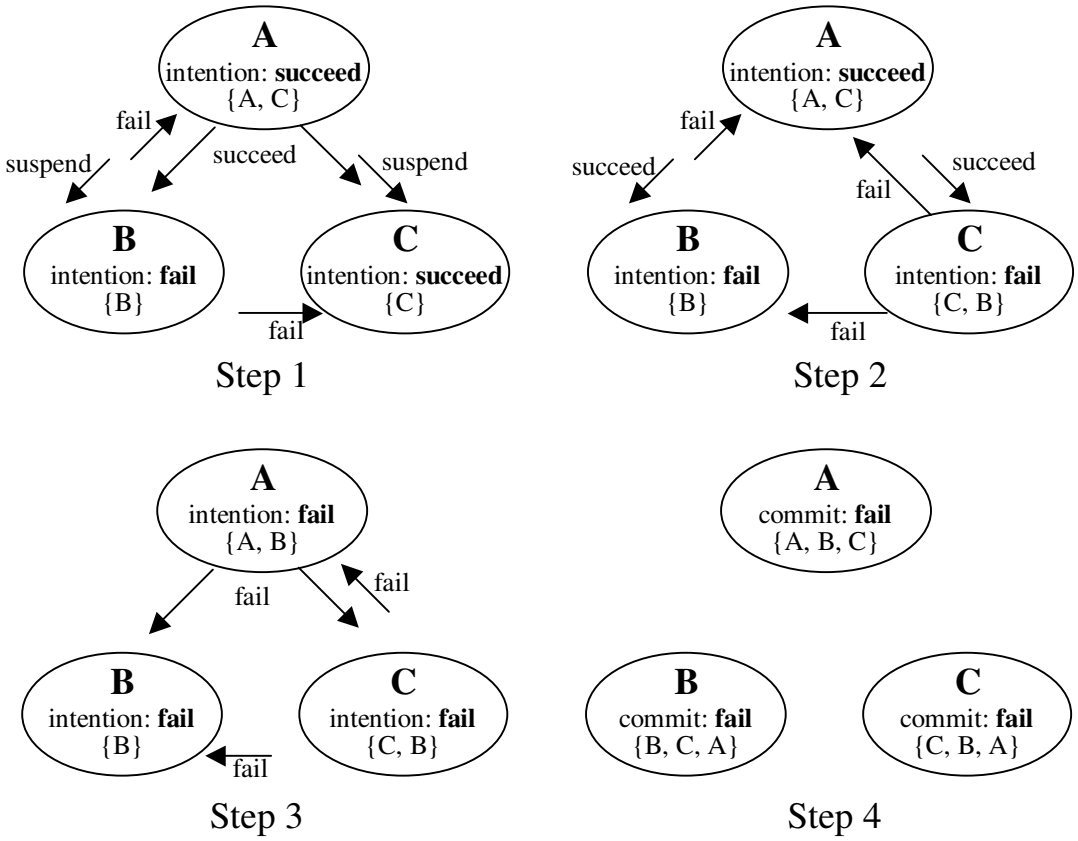

Figure 5-15. Behavior negotiation resolves conflicting intentions

Note that since each agent does have its own ABT, the effects of failing the goal may be different for every agent, resulting in the cascading failure of different parents (recursively), the removal of different executing subtrees, and the unsuspension of different goals. These cascading effects on the ABT may affect other joint goals, involving the same or different team members, starting new negotiation activity. Also note that during this negotiation, the agents continue to behave. Only the line of expansion containing the negotiating joint goal is blocked. All other lines of expansion are still available for execution. Any joint goals in these other lines of expansion remain free to engage in parallel negotiations.

As a second case, consider the case of A, B and C negotiating entry into a joint goal, perhaps initiated by A. B and C select appropriate joint behaviors and broadcast their intentions to enter to the rest of the team. Assume that A and B receive all messages and commit to entry, while C has not yet received all messages, perhaps because the intention to enter sent from B to C is strangely delayed, either by the communication infrastructure or because of the scheduling of C. This situation is depicted as Step 1 in Figure 5-16. Since A and B have committed to entry, they add the joint goal and their respective local versions of the joint behavior to their ABT's and begin executing the behavior. At this point, A begins an exit negotiation, perhaps an intention to fail propagating from a step

failure. A broadcasts an intention to fail, and, because of the communication infrastructure, C receives the failure intention before the entry intention. This situation is depicted in Step 2. What should C do? One possibility is that C internally ignores the failure intention (it has no effect on the ABT), but pretends to fail by broadcasting an intention to fail, thereby allowing the team to fail. When the intention to enter is eventually received, it would have to be ignored, perhaps using a timestamp mechanism to ignore any intention with respect to a particular goal that is older than the most recent intention received for that goal. But, though on one level it seems safe for C to preemptively "fail" this non-existent goal, the problem is that C will see a *different history* with respect to the joint goal than A or B. Where A and B saw the goal enter the ABT and then fail, C never sees the goal enter the ABT at all. And the appearance and failure of this goal may have additional effects on the ABT as failure propagates, as demons that may have been watching for the goal fire, and, if Em is being used, as emotion state is modified by the pursuit and failure of the goal. So the solution ABL adopts in this case is for C to enter a state where it waits for entry with an intention to fail as soon as the goal is entered (Step 3). While A and B wait for C's failure intention, C, upon eventual receipt of the intention to enter, selects a behavior, places the joint goal and behavior in the ABT, and immediately fails the goal. The goal enters an intention to fail state, adds the previously received failure intentions to the commit set, and broadcasts an intention to fail to the rest of the team (Step 4). Though this example used a fail intention, it works symmetrically for all exit intentions.

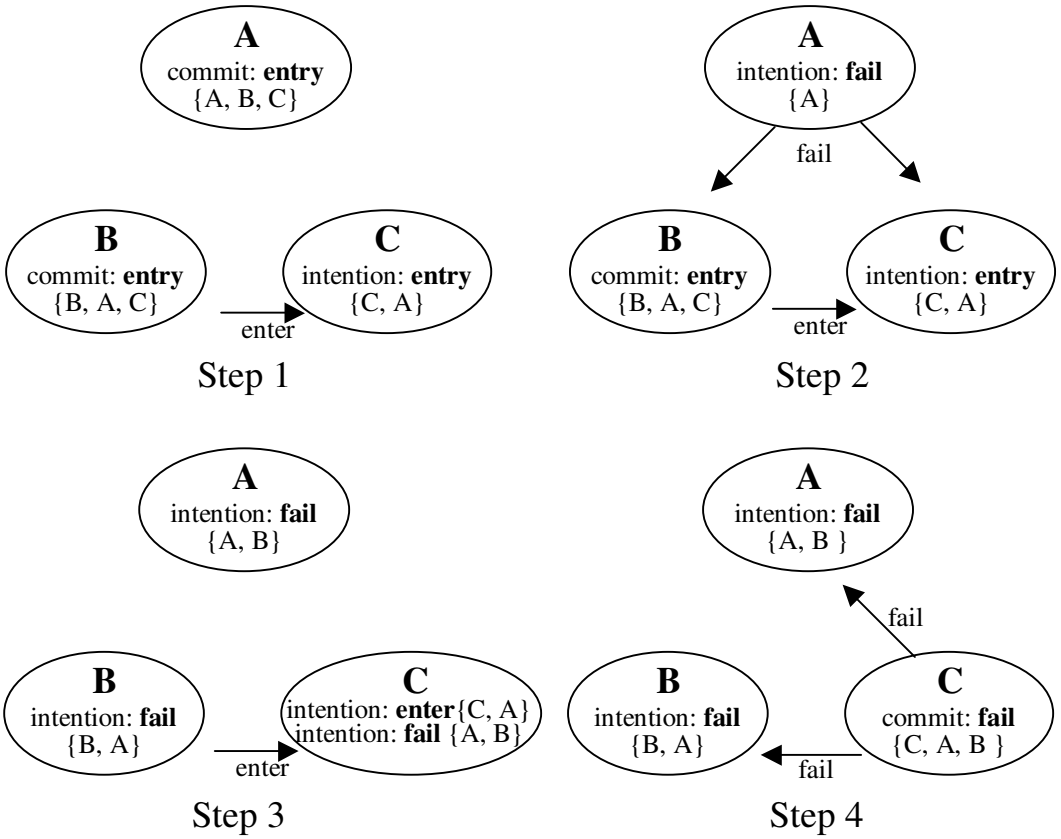

Figure 5-16. Behavior negotiation resolves out-of-order entry and exit

An agent can induce two versions of this same race condition, suspend before entry and remove before entry, purely through the action of its own ABT. A joint goal may be in the middle of entry negotiation when it is suspended by a goal conflict higher up in the ABT. Similarly, a negotiating joint goal may be removed by success or failure of a goal or behavior higher in the ABT. Both cases are handled by the same solution: wait for entry then immediately initiate exit negotiations.

Freezing Subtrees

The success, failure or suspension of any goal, whether individual or joint, raises another issue – what to do with joint goals within the subtree rooted at the exiting goal? This situation is depicted in Figure 5-17. The individual goal G_1 succeeds, perhaps because a success test annotation on G_1 becomes satisfied. When G_1 succeeds, all goals and behaviors within the subtree rooted at G_1 are removed (any executing leaf steps are aborted before they are removed). But in this case, the subtree contains a joint goal $G_{joint-3}$. How should the removal of the subtree rooted at G_1 be coordinated with the removal negotiation for $G_{joint-3}$? In order to preserve the joint goal guarantee that a joint goal only exits when the entire team has committed to the exit, $G_{joint-3}$ should negotiate removal before being removed from the tree. But while it is negotiating, what should happen to other lines of expansion rooted at G_1, such as the subtree at G_2? It seems strange for steps within the subtree at G_2 to continue to execute when the parent goal G_1 has succeeded, just because $G_{joint-3}$ is negotiating. Furthermore, continuing execution within the tree may cause a failure or suspension to occur, propagating this up to G_1, even though G_1 has supposedly already succeeded. The solution adopted in ABL is to *freeze* the subtree rooted at a succeeding or failing ABT node.

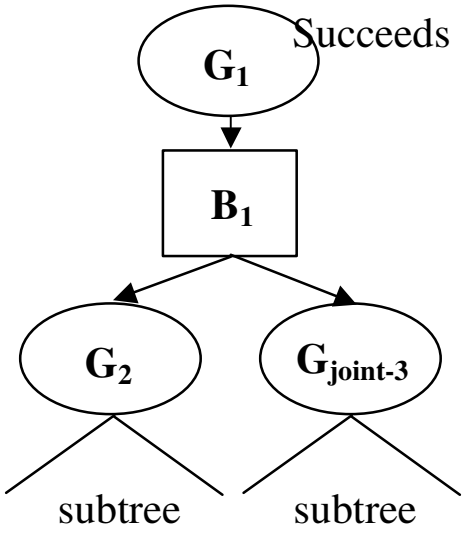

Figure 5-17. Succeeding subtree containing a joint goal

Figure 5-18 shows the algorithm for subtree freezing. The first two steps deactivate all success tests and context conditions in the subtree. This prevents subtree goals and behaviors from succeeding or failing because of success tests and context conditions succeeding and failing while negotiation occurs in the subtree. For example, imagine that context conditions are left activated and that B_1's context condition fails while $G_{joint-3}$

negotiates removal. The reason the subtree is being removed in the first place is that G_1 succeeded because of its success test succeeding. Now G_1 potentially fails (if it can't find another behavior) because of a context condition that failed *after* G_1's own success test already succeeded. The deactivation of success tests and context conditions in the subtree makes sure that the temporal ordering of ABT node success and failure is consistent with the temporal ordering of the failure of context conditions and the success of success tests. Steps 3 and 4 make sure that no execution in the subtree continues after the decision cycle in which the parent of the subtree succeeded or failed. Step 5 initiates removal negotiation for all joint goals in the subtree. Only *after* all joint goals have successfully negotiated removal is the subtree removed and the subtree parent succeeded or failed. If no joint goals are found during step 5, the subtree is immediately removed and the subtree parent immediately succeeds or fails (just like Hap). In our example, G_1 won't succeed until $G_{joint\text{-}3}$ negotiates removal. Of course, the line of expansion containing G_1 is blocked until $G_{joint\text{-}3}$ negotiates removal. Subtree freezing maintains an intuitive *within-agent* temporal ordering on ABT modification while preserving *cross-agent* team commitments to joint goals.

1. Recursively deactivate all context conditions in the subtree.
2. Recursively deactivate all success tests in the subtree.
3. Abort all executing physical acts in the subtree.
4. Remove all subtree leaf steps from the conflict set.
5. Recursively initiate remove negotiations for all joint goals in the subtree. Unlink negotiating goals from parents.

Figure 5-18. Algorithm for freezing an ABT subtree

Step 5 of the subtree freezing algorithm, besides initiating removal negotiation, also unlinks negotiating joint goals from their parent behaviors. This is done to avoid race conditions that can occur during subtree freezing. When a subtree is frozen, it may be the case that joint goals within the subtree are already negotiating goal exit. For example, consider the case that $G_{joint\text{-}3}$ in Figure 5-17 is already negotiating failure when the subtree is frozen. Since removal has a lower precedence than failure, the initiation of the remove negotiation is ignored. But, once $G_{joint\text{-}3}$ negotiates failure, failure should not propagate up the ABT. The subtree is being removed because G_1 succeeded; the propagation of failure from $G_{joint\text{-}3}$ would introduce an inconsistent state within the ABT. Decoupling the upward link from $G_{joint\text{-}3}$ to its parent prevents failure from propagating up the tree. $G_{joint\text{-}3}$ becomes a *failure island* within the ABT; it locally fails, perhaps having side effects if Em is in use or demons are waiting on $G_{joint\text{-}3}$ failure, but does not propagate failure. A similar condition arises if $G_{joint\text{-}3}$ is not negotiating before the subtree freeze and thus initiates remove negotiation, but a team member initiates success or failure during the remove negotiation. Since intentions to succeed or fail have higher precedence than intentions to remove, $G_{joint\text{-}3}$ will switch intentions and eventually succeed or fail. $G_{joint\text{-}3}$ again becomes a success or failure island. The idea of success and failure islands arises because the introduction of joint goals causes subtree removal to *take time*; if there are no joint goals in a subtree, removal is atomic with respect to the decision cycle.

Limits of Synchronization

ABL's ability to synchronize team behavior via joint goals is limited by communication delays and variable execution rates across the team. These limitations, similar to the clock synchronization issues discussed by Lamport [Lamport 1978], are fundamental and would thus be shared by any conceivable multi-agent synchronization protocol.

Consider the two cases in Figure 5-19. In case 1, messages sent from C to A propagate slowly (perhaps because of an intermittent network failure); messages sent between all other pairs propagate normally. During joint goal entry negotiation initiated by A, B and C quickly receive intentions to enter from the other team members while A is still waiting for C. Since B and C have received messages from all team members, they commit entry and begin executing their respective joint behaviors while A has not yet entered. To the extent that B and C need A to do something during the joint goal, their behaviors may block while further synchronizing with A, either through nested joint goal negotiations or less formally through sensing. In any event, B and C have entered the joint goal when A hasn't; the best the joint goal protocol can ensure is that B and C won't be allowed to exit until A enters (since A won't respond to intentions to exit until entry is achieved). If the joint goal is negotiating with team_needed_for_success, then B and C will have to wait as well for A to finish the joint behavior.

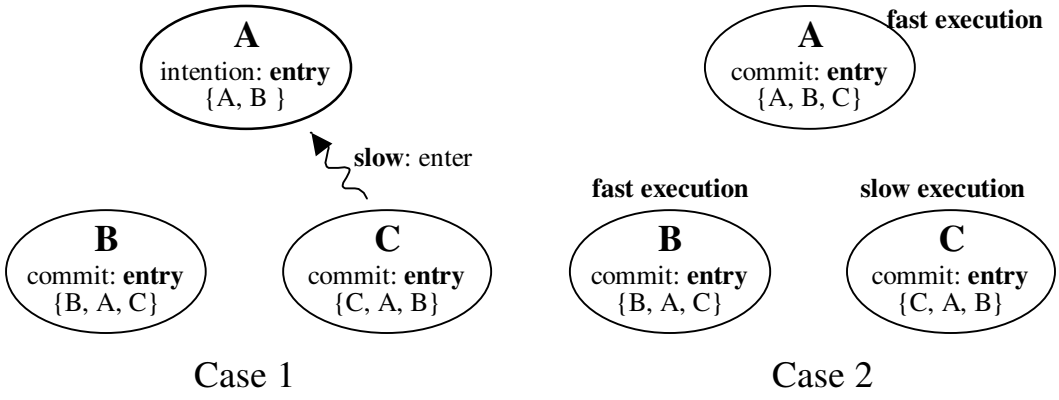

Figure 5-19. Cases demonstrating limits of synchronization

In case 2, C executes much more slowly than A and B. Again consider joint goal entry negotiation initiated by A. A and B quickly receive each other's intentions, then wait for C. When C eventually gets around to processing the entry intentions, it sends its intention to enter. At this point all three have committed; A and B immediately begin executing their joint behaviors, while C, though committed, has not yet placed the joint goal and behavior in its ABT. Again, the best the joint goal protocol can ensure is that A and B won't be allowed to exit until the slow C processes the exit intention and commits to the appropriate propagation of effects in its ABT.

Networked agents experiencing severe network delays, or physical agents separated by vast distances (e.g. a team of space probes separated by light minutes), would face these synchronization issues. For virtual agents executing on the same machine (with a fair scheduler), teams of physical agents with similar computational capabilities in relatively close proximity, or networked agents in a properly functioning network, these synchronization issues won't be of practical concern.

Decision Cycle With Goal Negotiation

ABL's decision cycle with joint goal negotiation is shown in Figure 5-20.

1. If there are pending negotiations, give them a chance to execute.
2. Else if needed, adjust the ABT for actions that have finished.
3. Else if needed adjust the ABT based on changes in the world.
4. Else if needed adjust suspended goals.
5. Else pick a leaf step to execute, and execute it.

Figure 5-20. ABL's decision cycle with joint goal negotiation

This decision cycle is the same as the Hap decision cycle in Figure 5-5, except that there is an additional step to execute pending goal negotiations. An alternative to executing negotiations as a step in the decision cycle would be to execute negotiations in response to ABT changes. This is how ABT effects propagate for non-joint steps. For example, the behavior step picked for execution in step 5 of the decision cycle might be the last step of a behavior. When the step succeeds, the behavior succeeds, causing the parent goal to succeed, etc, all during the execution of step 5. For the case of joint goals, the goal could negotiate success and propagate its effects during the decision cycle step. The problem with this is that the decision cycle *blocks* during the negotiation. If the negotiation takes some time (perhaps in a networked environment), or transition behaviors are associated with the goal (see page 101), blocking the decision cycle would compromise reactivity during the negotiation. Alternatively, one could spawn a negotiation thread at the goal; the thread would block while waiting to receive intentions from team members, allowing the decision cycle to continue. But the problem with this is that the negotiation thread, after committing to the intention, could *modify the ABT concurrently* with a step of the decision cycle; this could lead to all kinds of chaos. And ensuring mutual exclusion on the ABT (with locks or whatever) is not enough; we really want ABT modification due to negotiation to occur at a specific place within the decision cycle. The design adopted in ABL is to represent pending negotiations as continuations, each executing in its own thread. Negotiations are scheduled during step 1 of the decision cycle. When a negotiation is scheduled, it blocks the decision cycle thread and does a bit of work to further the negotiation, such as sending intention messages or checking the current negotiation state. If, in checking the negotiation state, it finds that the negotiation has committed, it performs the appropriate ABT modifications and exits, returning control to the decision cycle. If the negotiation is still waiting to commit (waiting for intention messages from team members), the negotiation blocks (a non-busy wait) and returns control to the decision cycle. When a negotiation is initiated, a negotiation thread containing the continuation for that particular intention is created and registered with the decision cycle[11].

The details of the decision cycle negotiation scheduling step are show in Figure 5-21. The decision cycle iterates over all pending negotiations. Any new negotiations, that is, negotiations registered with the decision cycle since the last time the cycle ran, are

[11] Negotiation continuations are implemented as anonymous instances of JointGoalNegotiationThread, a subclass of java.lang.Thread on which appropriate scheduling hooks for the decision cycle are defined.

started. Any completed negotiations, that is, negotiations that have already committed and finished their ABT modifications, are removed from the negotiation queue. In progress negotiations are given a chance to execute. The decision cycle blocks until the negotiation yields. In any one scheduling cycle, negotiations only run a short amount of time, so the decision cycle is not blocked for long. Generally, a negotiation is either waiting for commit, in which case it just has to run long enough to check the current negotiation state, or it is committing, in which case it just runs long enough to modify the ABT. The decision cycle continues to schedule negotiation threads until no thread completes. This is done because sometimes negotiation threads chain on each other, for example, when a joint goal negotiation is waiting for negotiations in its frozen subtree to complete.

```
repeat
        iterate over pending negotiations
                if a negotiation has not been started, start it
                else if the negotiation has already completed, remove it
                else give the negotiation a chance to execute
until no negotiation completes
```

Figure 5-21. Negotiation scheduling

This concludes the detailed discussion of the joint goal negotiation protocol.

Coupled ABL Agents Form A Multi-Mind

Joint goals introduce complex patterns of coupling between teams of ABL agents. When an ABL agent participates in a joint goal, the execution details of its ABT now depend on *both* its autonomous response to the environment as it pursues its individual goals and behaviors, *and* on the execution details of its team members' ABTs, but only to the degree those execution details impinge on the joint goal. This situation can be better understood by comparing it to two multi-agent extremes: one mind and many minds.

In the one-mind approach, a collection of agents are really the different effectors of a single entity. This single entity controls the detailed, moment-by-moment activity of all the "agents". One can certainly imagine writing such an entity in ABL; sensors and actions would be parameterized to refer to specific "agents". In an interactive drama context, this is similar to the story plans approach employed by Lebowitz [Lebowitz 1984; Lebowitz 1985], in which he generated non-interactive episodic soap operas (as text) by using story plans (as opposed to character plans) to coordinate multiple characters in specific story events. One-mind provides maximum coordination control, but also introduces maximum program complexity. Besides the usual data hiding and modularity arguments that such a program would be hard to write and understand, and that, consequently, unforeseen side effects would arise from cross-talk between "agents", there is the additional issue that much of the combinatorics of agent interaction would be thrust upon the author. All simultaneous agent activity, whether explicitly coordinating or not, has to be explicitly authored.

The many-minds approach is the intuitive model of strong autonomy. Agents individually pursue their own goals and behaviors. Any coordinated activity arises from sensed coordination between agents. The internal details of agents are hidden from each other, providing the data hiding and modularity that makes programs easier to write and

understand. Agent interaction is mediated by the world; much of the combinatorics of agent interaction arises through the world mediation without having to be explicitly authored. But, as argued on page 40, dramatic interactions in a story world require a degree of coordination difficult to achieve with sensing or ad hoc communication mechanisms.

Joint goals open up a middle ground in this apparent dichotomy between one and many minds. With joint goals, a collection of agents becomes a variably coupled multi-mind, neither a single master entity controlling a collection of puppets, nor a collection of completely autonomous agents, but rather a coupled system in which a collection of ABTs influence each other, not arbitrarily, but in a manner controlled by the semantics of joint goal commitment. At any point in time, an ABL agent may hold multiple simultaneous joint goals, potentially with different teams. These joint goals fully participate in the rich, cascading effects of normal ABT nodes; only now the web of communication established between specific nodes in multiple ABTs allows ABT execution effects to cascade *across* agents as well as *within* agents. As the number and frequency of joint goal commitments across a collection of agents increases, the collection of agents is moving towards a one-mind. As the number and frequency decreases, the collection of agents is moving towards many minds. There is still much work to be done in exploring idioms that harness the expressive power of joint goals. Chapter 6 describes one set of joint goal idioms we developed for *Façade*.

Additional ABL Extensions

Support for joint action is ABL's primary extension of Hap. There are, however, a number of more minor extensions. This section discusses ABL's support for connecting to asynchronous sensory-motor systems, meta-behavior mechanisms for clean, safe reflection, miscellaneous working memory extensions, and the additional annotations atomic, post and post-to.

Support for Asynchronous Sensory-Motor Systems

As discussed in the section *Asynchronous Agents and World* on page 80, both the text world and real-time implementations of Hap schedule the Hap agents and world in a top-level loop; many games employ a similar loop to schedule game characters and the world. ABL agents, on the other hand, are designed to run fully asynchronously with both the world and each other. This design allows ABL to more easily connect to different worlds. Besides solving the usual synchronization issues of multi-threaded, multi-process programming within the ABL infrastructure, the major ABL support for providing "plug and play" capability with different bodies and worlds is provided by the mechanisms for defining sensors and actions.

Sensors and Actions

Sensors and actions mediate between an ABL agent and the world. To connect an ABL agent to a new sensory-motor system (body within a world), an ABL author defines a new set of appropriate sensors and actions. Sensors and actions are implemented by extending abstract sensor and action classes provided by ABL. These classes define a collection of methods and contracts expected by ABL's runtime system. As long as the new sensors and actions define these methods appropriately, where appropriately means

the method satisfies the expected contract, the new sensors and actions work with the ABL runtime system.

To define a new sensor, the author must define the following methods on the sensor object.

- canBeParallel() – return true if the sensor can be run in parallel, false otherwise. Whether a sensor can be run in parallel depends on the details of the sensory-motor system, specifically whether the sensory system is reentrant.
- senseOneShot(Object[] args) – called when sensing is needed to test a precondition. The sensor should update appropriate WME(s) in working memory to reflect the new sensed value. If testing the precondition(s) requires multiple sensing, ABL optimizes sensing by spawning multiple threads to perform parallel sensing for those sensors whose canBeParallel() flag returns true.
- initializeContinuous(Object[] args) – called when a continuously monitored condition requiring sensing (i.e. a sensed context condition or success test) is first entered. The sensor should update appropriate WME(s) in working memory to reflect the current sensed value and perform any internal bookkeeping to prepare for continuous sensing. As an example of bookkeeping, a position sensor may want to store the sensed value, and, while continuously sensing, only report a new position when the new value differs from the stored value by some threshold.
- senseContinuous(Object[] args) – called while a continuously monitored condition requiring sensing is active. Like initializeContinuous, senseContinuous should update appropriate WME(s) and possibly perform internal bookkeeping.

Example sensors for *Façade* include position and rotation sensors that sense the position and rotation of characters and objects in the world, held object sensors that sense which objects are being held by a character (and in which hands), and a text sensor that senses the text typed by the player.

To define a new action, the author must define the following methods on the action object.

- execute(Object[] args) – called when an ABL act step executes, this method actually starts the action in the world. As the decision cycle blocks on calls to execute, it should return as quickly as possible. This does not mean that the action itself must complete immediately, only that the call to execute should happen as quickly as possible. The action, such as moving an arm, can continue in the world while the decision cycle selects other steps. The act step that resulted in the call to execute doesn't succeed or fail until the action in the world actually completes (with success or failure).
- abort(Object[] args) – called when an executing act is removed from the ABT. This method should instruct the sensory-motor system to abort the action (whatever that means given the particularities of the action and the motor system) and clean up any internal state being maintained by the action instance.
- completionCallback(boolean status) – for those actions that take time to execute in the motor system, the action may request that the sensory-motor system call the action back when it completes. The completion callback takes a boolean parameter: true if the action completed with success, false if it completed with failure. A default implementation is provided that appropriately sets the completion status of the action instance to success or failure.

For defining the execute method, concrete physical actions may employ one of a number of strategies:

- For actions that execute very quickly in the underlying sensory-motor system (e.g. setting a flag), execute can directly perform the appropriate action and set the completion status.
- Actions that take some time to execute may spawn a separate thread and immediately return. The spawned thread is responsible for setting the completion status.
- For actions that take some time to execute and for sensory-motor systems that accept callbacks, execute may initiate the action, register a completion callback, and return immediately. The sensory-motor system then calls the callback when the action finishes executing. The default definition of completionCallback appropriately updates the action's completion status.

Example actions for *Façade* include a walk action, which causes a character to take a step towards a destination, a gesture action, which performs a short animation script (e.g. point) with one of a character's arms, and a music action, which starts playing a specified .mp3 file.

Registration of Sensors and Actions

Once sensors and actions have been defined for a particular sensory-motor system, the sensors and actions must somehow be connected to ABL code. This is accomplished through ABL *registration* declarations.

Sensed WMEs are registered with sensors. The ABL sensor registration syntax is:
register <WME class> with <concrete sensor class>;
For example, in Grace and Trip, the sensor that senses player gestures (e.g. kiss, hug, comfort) is registered with the PlayerGestureWME as follows:
register PlayerGestureWME with PlayerGestureSensor;
The registration of a WME with a sensor tells ABL that the sensor "owns" the WME. ABL is responsible for making sure that in the normal course of execution, whenever the WME is referenced (appears in a test), it contains an up-to-date value. For example, if a demon is waiting on a player to make a certain gesture, the demon's success test just contains a test of PlayerGestureWME. ABL makes sure that initializeContinuous is called on the PlayerGestureSensor when the demon test first enters the ABT, that senseContinuous is called repeatedly while the test remains in the ABT, and that player gesture sensing stops when the test leaves the ABT. If a reference to PlayerGestureWME appears in a precondition, senseOneShot will be called on PlayerGestureSensor before the precondition is tested. As in Hap, ABL reference counts sensors participating in continuous conditions in order to share sensing across multiple conditions referencing the same sensed WME. User code should be careful of directly modifying registered WMEs, as the ABL runtime will change them without notice if it detects that sensing is required. User code that wants to save a previously sensed value should store a copy of the value in a WME class not registered on a sensor.

ABL act signatures are registered with concrete action classes. The ABL action registration syntax is:
register <act signature> with <concrete action class>;
For example, in Trip, the act to perform a gesture with the left arm is registered with the action class DoTripArmLGesture as follows:
register armLGesture(int) with DoTripArmLGesture;
Now when the act armLGesture (which takes as an argument an enumeration integer indicating the specific gesture) appears as a step in a behavior, ABL automatically type checks the argument(s) at compile time, and at runtime makes use of the methods defined

on DoTripArmLGesture to execute the action, wait for a callback, and abort the action if the act step is removed from the ABT before completion. Both the name of the action and the types of the arguments in an act step are used to determine which registered concrete action instance to use, so actions can be overloaded as long as they are distinguishable by type signature.

Decision Cycle Sensory-Motor Call

ABL provides a general hook for sensory-motor bookkeeping that needs to be performed regularly: the decision_cycle_sm_call. This ABL declaration tells the ABL runtime to call the specified Java function every decision cycle. In *Façade*, the decision cycle sensory-motor call is used to check whether any asynchronous callbacks from the animation engine have occurred (e.g. action completion) and perform the appropriate actions (e.g. call completionCallback on an action instance). In *Façade's* case, this is done because the connection between the animation engine and ABL is a state-full interface, specifically a Windows shared memory dll accessed from the ABL side via the Java native interface. If, instead, remote procedure calls had been used to communicate between ABL and the animation engine, perhaps no decision cycle call would have been needed. However, given the vagaries of the many kinds of bodies and worlds ABL might be connected to, this call gives the author another degree of freedom in designing the sensor and action substrate.

Separation of Sensor and Decision Cycle

As a compiler option, ABL allows the author to specify that the sense cycle should run independently (in a separate thread) from the decision cycle. This is useful for sensory-motor systems that block on returning a sense value until the next value is available. For example, *Façade's* animation engine, when it receives a sense request, waits until the next frame boundary to update the state of the world and service the request. If the necessary sensing for continuously monitored conditions was done within the decision cycle thread, such an arrangement would throttle the decision cycle down to the update rate of the world (best case assuming individual sensors are parallelizable). By turning on the asynchronous sense cycle, the active sensors for continuously monitored conditions are constantly run in a separate thread. As this thread continually blocks on sensing, the decision cycle thread can continue. When new sensed values are available from the sensed thread, the decision cycle tests the sensed continuously monitored conditions at the appropriate place within the next decision cycle. Of course, if the sensory-motor system ABL is connecting to does not block on sensing (the case, for example, of virtual worlds that return that most recent value rather than waiting for the new one), then this compiler option should not be used.

Meta-ABL

A programming language supports reflection if it provides some mechanism for writing code that accesses and potentially modifies the execution state of the program itself. In Hap and ABL, this means the ability for behaviors to look at and potentially modify the ABT. Both the text world and real-time implementations of Hap support reflection; in fact, Hap's emotion system, Em, is implemented in Hap using reflection. The *design* of Hap included first-class support for reflection from the beginning. However, the *implementations* of Hap did not provide clean, general support for reflection. Until Sengers' work on transition behaviors [Sengers 1998a; Sengers 1998b], reflection was

generally used for reading rather than modifying the ABT state. Em, the primary system built using Hap reflection prior to Sengers' work, generates emotions based on detecting changes in the ABT, but does not directly modify the ABT. In Hap, programming using reflection required some degree of going "under the hood", and was employed by Hap power users to build significant extensions on top of Hap. ABL provides clean, general support for reflection; it is intended that reflection, rather than being viewed as exotic, become a standard technique in an ABL author's toolbox. However, powerful features such as reflection, which edge towards "you can do arbitrary things", are paradoxically useless without idioms providing guidance on how to use the feature. Sengers' transition behaviors are one such idiom. Chapter 6 describes additional reflection idioms we developed in the course of *Façade*.

Reflection WMEs

Reflection WMEs provide reflective access to the ABT. As ABT nodes are created and destroyed, corresponding reflection WMEs appear and disappear in working memory. Behaviors are able to access reflection WMEs just like any other WME; through preconditions, context conditions and success tests. Reflection WMEs provide a safe interface for accessing and modifying ABT nodes; since behaviors don't have direct access to the ABT nodes themselves, they can only perform the operations provided by the reflection WMEs.

Reflection WMEs are organized in a type hierarchy. For example, GoalStepWME is a subtype of ExecutableStepWME (here, *executable steps* are steps that take time to execute i.e. subgoals and physical actions), which is itself a subtype of StepWME. The reflection interface defined on a GoalStepWME (including methods inherited from parent WME types) is shown in Figure 5-22. The interface includes methods for reading and writing goal step state and performing actions on goal steps. Behaviors, as well as other step types, also have reflection WMEs; GoalStepWME is discussed in detail as an example.

The first section of Figure 5-22 shows the get accessors for reading goal step state[12]. Most of the readable state corresponds directly to step annotations or the step's execution state. Two of the accessors, getChild and getParent, provide access to the reflection WMEs for the parent and child behaviors. This allows tests to match on ABT structures, not just individual nodes. The getIsValid accessor allows ABL code to determine whether the reflection WME still mirrors a valid ABT node. While references to reflection WMEs within a test (i.e. precondition, context condition or success test) are guaranteed to be valid, if an ABL programmer grabs a reference to a reflection WME within a test and holds on to it for awhile, the WME may become stale because the node it mirrors has been removed from the ABT.

The set accessors allow a meta-behavior to change properties of the goal step. The important thing to note is that there are far fewer set accessors than get accessors. Providing such asymmetric access is one way in which ABL helps make reflection safe.

---

[12] ABL implementation detail: all WME fields are accessed through get and set accessors. When getting and setting WME state within a mental act, the accessors must be explicitly called. But WME field references within preconditions, context conditions and success tests can appear directly as shown in Figure 5-2 through Figure 5-4. As long as the names of get accessors conform to the Java bean reflection standard, the ABL compiler turns such references into appropriate get accessor calls. The creation of get and set accessors is handled automatically for WMEs defined within an ABL program using the ABL WME declaration syntax; for WMEs defined directly in Java outside of an ABL program, the author is responsible for creating accessors as well as some other bits of WME machinery.

If an author were able to directly change the signature of an executing goal, or change the suspended status without actually suspending the goal, the ABT would be left in an inconsistent state. The current collection of set accessors is conservative. For example, it may be perfectly fine to allow meta-behaviors to dynamically change the declared goal conflicts. I just haven't thought through all the implications and potentially dangerous cases that would have to be handled by the ABL runtime for this set accessor.

The action methods provide an interface for meta-behaviors to fail, succeed, reset and suspend the ABT node mirrored by the reflection WME. The reflection methods honor all joint goal commitments. Any subtree freezing and negotiation that would take place if the step had failed, succeeded or suspended on its own, also take place if the goal exit is initiated through the reflection interface. Suspend() and unsuspend() interact with conflicts as follows: a suspended goal unsuspends if the number of calls to unsuspend() equals the number of previous calls to suspend() and no goal conflicts are causing suspension. As an example, imagine that goal G_1 is suspended by a meta-behavior via a call to suspend(). Later, a meta-behavior calls unsuspend(). But, in the meantime, a higher priority conflicting goal has also entered the ABT. Though unsuspend() has been called, G_1 remains suspended until the conflicting goal leaves the ABT (or is itself suspended). Unsuspend() can't arbitrarily unsuspend goals with active goal conflicts since this would violate the declared goal conflicts. Similarly, imagine that G_1 is suspended due to a goal conflict and two meta-behaviors each call suspend() on the already suspended goal. When the conflicting goal leaves the tree, G_1 remains suspended until two calls to unsuspend() occur.

GoalStepWME provides the two isParent() methods as a convenience. Recursive parent tests can be useful while matching on reflection WMEs.

Readable goal step state

- boolean **getPersistent()**– true if step is marked persistent
- boolean **getPersistentWhenSucceeds()** – true if step is marked persistent when succeeds
- boolean **getPersistentWhenFails()** – true if step is marked persistent when fails
- boolean **getIgnoreFailure()** – true if step is marked ignore failure
- boolean **getHasSuccessTest()** – true if the step has a success test
- boolean **getEffectOnly()** – true if step is marked effect only
- boolean **getIsAtomic()** – true if the step is part of an atomic behavior (atomic behaviors described on page 93).
- int **getPriority()** – the priority of the step
- boolean **getIsSuspended()**– true if the step is currently suspended
- BehaviorWME **getParent()** – the reflection WME for the step parent
- boolean **getIsValid()** – true if the reflection WME still refers to a valid ABT node
- String[] **getConflicts()** – an array of signatures of declared conflicting goals
- boolean **getIsExecuting()** – true if the goal is currently executing
- boolean **getIsSuspended()** – true if the goal is currently suspended
- boolean **getIsExpanded()** – true if a behavior has been chosen for the goal
- String **getSignature()** – the goal signature
- BehaviorWME **getChild()** – the reflection WME of the child behavior

Writeable goal step state

- **setPersistent(boolean b)** – change the persistent step annotation
- **setPersistentWhenSucceeds(boolean b)** – change the persistent when succeeds step annotation
- **setPersistentWhenFails(boolean b)** – change the persistent when fails step annotation.
- **setIgnoreFailure(boolean b)** – change the ignore failure step annotation
- **setEffectOnly(boolean b)** – change the effect only step annotation
- **setPriority(boolean b)** – change the priority of the step

Actions performable on goal steps

- **fail()** – fail the step
- **succeed()** – succeed the step
- **suspend()** – suspend the goal step
- **unsuspend()** – unsuspend the goal step
- **resetStep()** – reset the goal step

Miscellaneous methods

- boolean **isParent(GoalStepWME parentGoal)** – returns true if the parameter parentGoal is a recursive parent of the goal step
- boolean **isParent(BehaviorWME parentBehavior)** – returns true if the parameter parentBehavior is a recursive parent of the goal step

Figure 5-22. Reflection interface on GoalStepWME

User Properties

ABL provides mechanisms for associating arbitrary author-defined state with steps. This is accomplished with a user property annotation:
with (property <name> <value>) <step>.
User properties are fully accessible through reflection: WME tests can match user properties and meta-behaviors can read and write them. As an example, in the ABL idioms developed for *Façade*, we have the concept of *beat goals*. During the execution of a beat, certain meta-behaviors actively manage beat goals differently than other simultaneously active ABL goals. Thus beat goals need to be marked as distinctive so that the meta-behaviors can enforce a different policy regarding beat goals (exactly analogous to distinctively marking a goal with the persistent annotation so that it is treated differently by the ABL runtime). User properties allow us to create a new goal annotation for beat goals, e.g.:
with (property isBeatGoal true) subgoal AffinityGame().

User properties must be declared before being used. Property declarations allow the ABL compiler to perform appropriate type checking of properties. For the isBeatGoal property, this declaration looks like: property boolean isBeatGoal.

The introduction of user properties is a step towards achieving parity between the ABL runtime and authored meta-behaviors. The ABT consists of decorated nodes; some of these decorations are defined by ABL and some defined by the author. Both authored meta-behaviors and the ABL runtime are influenced by these various decorations and are able to read and write them.

Error Conditions

ABL guards against errors introduced by reflection. Consider the situation in Figure 5-23. In its precondition, behavior Bad grabs references to two goal step children of behavior Foo. Bad then first fails one of the steps and succeeds the other. What happens? When goal1 is failed, all failure processing, including failure propagation, completes before the call to fail() returns[13]. Now we are in trouble when we succeed goal2; goal2 isn't even in the ABT anymore. When the reflection interface is employed on invalid reflection WMEs, the ABL runtime throws an exception. Note that the user code could have guarded against this case by testing getIsValid() before succeeding goal2.

[13] If any joint goals initiate negotiation during failure propagation, the call to fail returns *before* the negotiations complete (and thus before failure propagation completes) as we never want to block on joint goal negotiations.

```
joint sequential behavior Bad() {
   precondition {
     FooWME = (ParallelBehaviorWME signature == "Foo()")
     goal1 = (GoalStepWME parent == FooWME)
     goal2 = (GoalStepWME parent == FooWME)
     (goal1 != goal2)
   }

   mental_step {
     goal1.fail();
     goal2.succeed(); // This will cause an exception!
   }
}
```

Figure 5-23. Example behavior that causes a reflection exception

Working Memory Extensions

Working memory support is provided as a stand-alone facility. ABL agents create an instance of WorkingMemory for their internal working memory, but other components of the *Façade* architecture also create and maintain working memories. For example, the drama manager uses an instance of WorkingMemory as a story memory for storing global story state. Besides performing the usual hashing to provide efficient query access to stored WME's, WorkingMemory provides named access to a global space of working memories, as well as simple support for episodic queries.

Multiple Named Memories

When a memory is created, global access to the memory can be granted by naming and registering the memory. Any WME match test can simultaneously reference multiple memories; if a specific memory name is not specified, the default memory is the ABL agent's working memory. In *Façade*, agents make use of named memories to access the global story memory. If an author wants to allow agents direct access to each other's working memories (mind reading), the agents' working memories can be globally registered. If the author wants the agents' memories to be private, then the working memories aren't registered.

Episodic Memory

An episodic memory supports temporal queries over temporal sequences. WorkingMemory provides simple support for episodic queries. TimestampedWME is one of the abstract WME superclasses that can serve as a parent of concrete WME types. Instances of TimestampedWME provide accessors for a time stamp. If the time stamp hasn't been previously set, working memory sets the time stamp when a TimestampedWME is added to memory. A number of working memory methods support simple temporal queries over instances of subtypes of TimestampedWME, including findNext, findPrev, findFirst, findLast, findAll (in a temporal range), countWMEBefore, and countWMEBetween. The WMEs created by the post and post-to annotations (see below) are examples of TimestampedWMEs. Currently, TimestampedWME only supports point events, not events with duration, so queries for all 13 of Allen's temporal relations are not yet supported [Allen 1983]. WME test syntax does not yet directly support episodic tests,

except insofar as arbitrary Java boolean expressions can be embedded in match tests, thus allowing calls to the temporal query methods to embedded.

Step posting

Two step annotations, post and post-to, add a WME to working memory when the step completes (with either success or failure). Post adds the WME to the agent's working memory, while post-to adds the WME to a specified named working memory. The step completion WMEs are subtypes of TimestampedWME. As an example, consider using post to annotate a follower's subgoal in the Woggle game of follow the leader (see page 76): with (post) subgoal CatchUp(). When the step completes, with either success or failure, a CompletedGoalWME is added to working memory. A CompletedGoalWME, the definition of which is provided by the ABL runtime, contains the name of the goal, its completion state (success or failure), the name of the agent who performed the goal, any goal arguments, and, as a subtype of TimestampedWME, a timestamp. The `post` annotation creates the WME, automatically fills in the fields with appropriate values, and adds it to memory.

This facility, inspired by the sign management system in Senger's extension of Hap [Sengers 1998a; Sengers 1998b], can be used to provide an agent with a selective episodic memory. The future behavior of an agent can now conditionally depend on past episodic sequences. Since the ABT no longer has state for *already completed* subgoals and actions, an ABL agent's reflective access to its own ABT doesn't provide access to past episodic sequences. In the Woggle example, a follower could use the episodic information about CatchUp to stop playing follow the leader if it has had to catch up a certain number of times within a short interval (say, 3 times in the last 30 seconds). In a team situation, completed step WMEs posted to a shared memory might be used to provide lightweight coordination within a joint behavior if the more heavyweight commitments provided by further subgoaling joint goals are not needed.

The post annotations, combined with ABL's support for reflection, provide support for Sengers' transition behaviors. The posting of signs is accomplished by appropriately annotating low-level subgoals. In Sengers' system, the higher level signifiers are posted by behaviors that look for patterns in the lower level signs (compound sensors that internally sense signs). The same approach works in ABL by writing demons (compound sensors), appropriately annotated with post, that fire when a given pattern of completed step WMEs appears.

Atomic

The behavior annotation atomic prevents other active behaviors from mixing in during the execution of the atomic behavior. Atomic behaviors are useful for atomically updating state (e.g. updating multiple WMEs atomically). Behavior atomicity is inherited by behavior steps (though steps can't directly be annotated as atomic, only behaviors). If there are any atomic steps in the conflict set, the decision cycle ignores continuously monitored conditions and only selects atomic steps. Continuously monitored conditions are ignored because ABT changes resulting from the success of success tests or the failure of context conditions could result in the atomic behavior being removed from the tree in mid-execution, thus breaking the semantics of atomicity. Atomicity is inherited during subgoaling, so a subgoal executed within an atomic behavior will cause the selected behavior to execute atomically, even if the selected behavior is not itself annotated with atomic.

If there are multiple atomic steps available in the conflict set, one is selected using standard step arbitration. As long as atomic behaviors consist (recursively) only of mental acts and subgoals, multiple simultaneous atomic behaviors will be serialized by the prefer-current-line-of-expansion heuristic. But, if any of the simultaneous atomic behaviors block, perhaps by executing a physical action, then *another atomic behavior will mix in*. In practice, blocking steps tend to never be included in atomic behaviors. Rather, atomic behaviors are used sparingly to accomplish short sequences of purely computational activity. Time-consuming atomic behaviors are dangerous, as they impair reactivity.

It may seem that atomic demons aren't possible. Since success tests aren't tested when the conflict set contains atomic steps, it appears that a wait step with a success test within an atomic behavior would block forever and prevent all other behaviors from executing. Wait steps, however, are never actually added to the conflict set. Wait steps are not executable, though the success of a success test on a wait step still propagates normally in the ABT. So in the case of an atomic demon, the wait step is not in the conflict set, so no atomic step is in the conflict set, so the decision cycle precedes normally. However, when the success test succeeds, the resulting ABT modification will cause an atomic step to enter the conflict set; the rest of the atomic demon will therefore execute atomically.

Related Work

As mentioned previously, ABL builds on the Oz Project work on believable agents [Bates, Loyall & Reilly 1992a; Neal Reilly 1996; Loyall 1997; Sengers 1998a], both technically, in that ABL is a re-implementation of Hap adding additional features and language constructs, and philosophically, in that ABL is informed by the Oz stance on believability.

The Media Lab's Synthetic Character group explores architectures that are based on natural, animal systems, particularly motivated by the ethological study of animal behavior [Blumberg 1996]. Their recent architecture, C4 [Burke et.al. 2001], builds on their previous architectures, and includes a focus on learning, particularly reinforcement learning for action selection (see [Yoon, Blumberg, & Schneider 2000] for a discussion of animal training techniques applied to believable characters). Their work is grounded in the premise that modeling realistic, animal-like, sensory and decision making processes is necessary to achieve believability, particularly the appearance of self-motivation and the illusion of life (see page 8 for a list of the requirements for believability).

The Virtual Theater Project at Stanford has explored the use of explicitly represented character models in synthetic actors. For example, in the *Master and Servant* scenario, the agents make explicit use of the notion of *status*, borrowed from improvisational acting, to condition their detailed performance of a dramatic scenario [Hayes-Roth, van Gent & Huber 1997]. In the *Cybercafe*, the agents make use of explicit personality traits (e.g. confidence, friendliness), borrowed from trait theories in psychology, to condition the selection and performance of behaviors [Rousseau & Hayes-Roth 1998]. More recent work has focused on building annotated environments in which a character dynamically gains new competencies and behaviors from objects in the environment [Doyle 2002; Doyle & Hayes-Roth 1998]. In this approach, a character's core, invariable features are factored out from the character's environment-specific capabilities and knowledge, with the later being represented in the environment rather than in the character.

A number of groups have explored the use of believable agents in educational simulations. Such work requires that the agent simultaneously communicate its personality while achieving pedagogical goals. The IntelliMedia Project at North Carolina State University has used animated pedagogical agents to provide advice to students in constructivist learning environments [Lester et. al. 1999; Lester & Stone 1997]. The group has also performed studies to determine whether using agents to deliver advice in such environments actually improves student learning vs. providing the same advice in a non-agent-based form [Moreno, Mayer & Lester 2000]. The Institute for Creative Technologies at USC is building story-based military training environments inhabited by believable agents. The agent architecture makes use of a cognitive appraisal model of emotion (similar to Em) [Gratch & Marsella 2001] built on top of the STEVE agent architecture [Rickel & Johnson 1998].

Over the last several years, game designers have built a number of innovative believable agent architectures for use in commercial games. The virtual pets products *Petz* and *Babyz* [Stern, Frank & Resner 1998; Stern 1999] employ a multi-layer architecture in which state machines perform low-level action sequencing while a goal-processing layer maintains higher-level goal state, activating and deactivating state machines as needed. *Creatures* employs an artificial life architecture in which a neural net selects actions, an artificial hormone system modulates the activity of the neural net, and a genome encodes parameters of both the neural net and hormonal system, allowing traits to be inherited by progeny [Grand 2001]. Finally, the creatures in *Black & White* make use of decision tree learning to learn to perform actions on the player's behalf, while the simulated people in the *Sims* hill-climb on a desire satisfaction landscape defined by both the internal drives of the agent (defines which drives the agent currently needs to satisfy) and the current objects in the environment (objects provide actions for satisfying drives).

Future Work

Transition Behaviors

Sengers, in her analysis of the *Luxo Jr.* short by Pixar, identifies behavior transitions as a major means by which narrative flow is communicated [Sengers 1998a]. Animators actively communicate changes in the behavior state of their characters (e.g. the change from playing to resting) through short transitional behaviors that communicate why the behavior change is happening. Sengers' architectural extensions to Hap provide support for authoring individual transition behaviors [Sengers 1998a]. In her approach, the behavioral decomposition of the agent is based on what the viewer should perceive, not on the internal logic of the activity. Each behavior communicates something to a viewer. Transition behaviors connect changes from one behavior to another. The transitions are responsible for communicating *why* the agent is switching between the behaviors. Her architecture facilitates the authoring of transition behaviors by providing a structured mechanism for behaviors to reflect on the behavioral state of the agent. Behaviors can be annotated with labels representing what the execution of the behavior communicates to a viewer. The hierarchical structure of the behaviors is mirrored by the hierarchical structure of these labels. As behaviors execute, the labels are posted to working memory. Transition behaviors use these posted labels to figure out *where* the transition is switching from and where it is switching to, and thus *what* needs to be communicated to the viewer to explain *why* the switch is happening. As described above, ABL could

similarly support transition behaviors by using the post and post-to annotations to post WMEs representing step completion to working memory, demon behaviors that trigger on combinations of step WMEs to recognize higher level patterns of activity, and meta-behaviors that use reflection to communicate the transition.

However, Sengers also notes that animators make use of coordinated multi-character transitions to communicate changes in multi-character behavioral state, but does not provide architectural support for this in her system. By exposing the negotiation protocol to the agent programmer, ABL could support the authoring of behaviors that *communicate transitions in multi-agent behavior state*. For example, consider the case of one Woggle inviting another to play Follow the Leader. The leader pursues the joint goal, selects a behavior, and sends an intention-to-enter message to the follower. If the follower were suddenly to "snap" to the joint goal and immediately start playing, this abrupt transition could be confusing to the audience. In another case, the follower may reject the intention-to-enter, perhaps because the follower currently isn't in the mood to play (i.e. the precondition on the joint Follow the Leader behavior test current emotional state). But if the follower just ignored the request (sending an intention-to-refuse-entry behind the scenes), the audience may be confused, perhaps thinking that the follower didn't even see the request. Both these cases indicate the desirability of associating transition behaviors with steps of the negotiation protocol. Figure 5-24 illustrates what a joint behavior for the follower might look like if ABL were extended to associate behaviors with negotiation steps in the joint protocol.

```
joint sequential behavior PlayFollowTheLeaderAsFollower() {
    decide {subgoal DecideFollowTheLeader();}
    accept {subgoal AcceptFollowTheLeader();}
    reject {subgoal RejectFollowTheLeader();}
    decide-to-terminate {subgoal StopFollowing();}
    activity-terminated {subgoal FollowTheLeaderTerminated();}

    // Steps for following here
}
```

Figure 5-24. Joint behavior with negotiation protocol transitions

Synchronization

As currently implemented, the joint goal mechanism introduces an asymmetry between joint goal initiators and responders. When one member of a team initiates the joint goal, the other members of the team, on successful entry into the joint goal, spawn the goal at the root of their active behavior tree (ABT). Only the joint goal initiator has the goal deeper within the ABT. If other members of the team initiate joint subgoals in the service of the original joint goal, these joint subgoals will appear at the original initiator's ABT root.

The ABT is not just a bookkeeping mechanism controlling execution, but a *representation* of the agent's activity. The ABT captures the structure of an agent's thoughts, its mind. Reflective processes, such as the emotion system Em or the handlers described in Chapter 6, treat the ABT directly as a representation. Additionally, the mechanisms for success and failure propagation, the many annotations that modify success and failure propagation, and continuously monitored conditions, all work together to embed a goal deep in the ABT within complex patterns of activity. Thus, we would like the joint goals of all team members, both responders and initiators, to

potentially be rooted deeper in the tree. One possible mechanism for supporting this is a new synchronize annotation.

Consider Grace and Trip's HaveAFight() behaviors in Figure 5-25. In Trip's behavior, he and Grace first coordinate on a joint subgoal and yell at each other for awhile. When they are done yelling, Trip independently stomps over to the bar and makes a drink for himself, banging bottles and muttering while he makes the drink. After he's done making the drink, he coordinates on another joint subgoal and continues the fight. Grace similarly fights for awhile, then stomps over to the cabinet where she fiddles absentmindedly with her trinket collection while muttering angrily. At some point she is done fiddling with her collection and continues the fight.

```
// Trip's behavior
joint sequential behavior HaveAFight() {
   with (synchronize) joint subgoal YellForAwhile();
   subgoal StompOverToBarAndMakeDrink();
   with (synchronize) joint subgoal FightSomeMore();
}

// Grace's behavior
joint sequential behavior HaveAFight() {
   with (synchronize) joint subgoal YellForAwhile();
   subgoal StompOverToCabinetAndFiddleWithCollection();
   with (synchronize) joint subgoal FightSomeMore();
}
```

Figure 5-25. Abstract HaveAFight() behaviors illustrating synchronize

To understand what the proposed synchronize annotation does, first imagine the case where it is absent. Imagine that Trip initiates the HaveAFight() goal, and that both Grace and Trip enter their respective versions of that behavior. One of them will be the first to execute the YellForAwhile() subgoal. If Grace were the first to execute this subgoal, she would broadcast an intention-to-enter to Trip, causing him to spawn this goal at the root of his ABT. As they jointly pursue the new YellForAwhile() line of expansion, Trip will continue to pursue the HaveAFight() line of expansion, eventually initiating YellForAWhile() on his side, causing Grace to spawn the goal at her root and enter another copy of the behavior. At this point each is pursuing two copies of the joint behavior YellForAwhile(), one copy rooted at the subgoal within HaveAFight(), and the other rooted at the root of the ABT. This is not what the behavior author intended; rather it is intended that when the characters synchronize on the joint subgoal HaveAFight(), they would each begin pursing their local version of YellForAwhile() rooted at the respective subgoals within their local versions of HaveAFight().The synchronize annotation provides this functionality, allowing the behavior author to specify that a joint behavior should be rooted at a specific subgoal, rather than at the ABT root. Since the YellForAwhile() subgoal in both Grace and Trip's local versions of the HaveAFight() behavior are both annotated with synchronize, regardless of who initiates this joint goal, they will both have the goal rooted within the HaveAFight() behavior. Eventually they will negotiate exit from YellForAwhile() and independently pursue their sulking goals; these independent goals, however, are still pursued in the service of the higher-level HaveAFight() joint goal. Eventually, one of them will stop sulking and begin pursuing the last subgoal. Imagine that Grace stops sulking first and pursues the joint subgoal FightSomeMore(). Trip is still sulking behind the bar, either fixing his drink or angrily sipping at it. When Grace signals the intention-to-enter FightSomeMore(), since this goal

is annotated with synchronize, it forces Trip to complete StompOverToBarAndMakeDrink(), root the requested FightSomeMore() at the appropriate step within HaveAFight(), and begin pursuing this goal. In general, within sequential joint behaviors, synchronization on a synchronize subgoal forces the success of all steps between the current step and the synchronize subgoal, and moves the step counter up to the synchronize subgoal.

Besides allowing joint goals to be rooted more deeply within the ABT for both joint goal initiators and responders, synchronize also supports decentralized decision making. In the HaveAFight() example, Grace and Trip independently decide when to stop sulking. Whichever one stops sulking first causes the team to advance to the next step of the HaveAFight() behavior.

The synchronize annotation is just one possible language extension for supporting richer patterns of commitment across a team of ABL agents. The general goal of this class of language extensions is to provide a more symmetric relationship between joint goal initiators and responders.

CHAPTER 6
ABL IDIOMS IN *FAÇADE*

Introduction

This chapter describes the various ways we use ABL, the *authorial idioms*, for authoring the goals and behaviors for Grace and Trip in *Façade*. The expressive power of an architecture or language lies in the authorial idioms it supports. The idioms are the meeting place where the negotiation between authorial intention and the computational structures offered by the architecture occur. As authors invent new ways of using an architecture, this both reveals the power of the architecture, and the places where the architecture falls short, where it needs to be modified to more directly support the author's intentions. A true understanding of the strengths and weaknesses of an architecture can only be achieved by studying idioms[14].

Body Resources

One of the strengths of a reactive planning language such as ABL is the possibility of rich, dynamic combinations of behavior. While a behavior author defines distinct behaviors for specific situations, much of the richness of an ABL character comes from the interactions *between* multiple behaviors as the character pursues multiple goals in response to changes in the environment. For example, within some particular beat, Trip may have a dialog behavior, TripDialog(), in which he glances quickly at Grace, then looks at the player and says "Somehow I *knew* Grace would say that", delivering the line with an irritated expression on his face while gesturing with both hands as he says the word "knew". While this behavior is fine on its own, the real richness in performance comes from how this behavior can combine with other behaviors that Trip might be simultaneously pursuing. What if Trip happens to be engaged in the behavior FixDrinks(), making a drink behind the bar while saying this line? One or both of Trip's hands are busy at various times while making drinks, so how should the gesture mix in? If during FixDrinks() Trip normally looks down at the blender while making blended drinks, how does this combine with the glance towards Grace and the player if he just happens to be operating the blender when the glance is supposed to occur? What if Trip happens to be walking across the room when he delivers the line?

In any given interaction with the characters, the specific performance the player experiences depends on how the player's interactions and the timing of these interactions results in different simultaneous behaviors combining within the believable agents. It is this kind of combinatorial richness in behavior that helps make a character believable. But this combinatorial richness does not come for free; there must be some mechanism for specifying how various behaviors can combine. In *Façade*, this mechanism is body resource management, a collection of ABL utility behaviors for keeping track of which behaviors currently "own" which parts of the body.

[14] A more detailed (and theoretical) description of the relationship between the authorial affordances and idioms of an architecture can be found on page 142.

Before proceeding into the details of the body resource manager, it is useful to examine why such a mechanism is needed at all. On first blush it may seem that the conflict mechanism that ABL inherits from Hap would be enough. Recall that the conflict mechanism allows an author to specify conflicts between physical acts and goals. When conflicting goals or physical acts appear in the active behavior tree (ABT), the lower priority goal or physical act suspends, with same-priority goals and acts treated on a first come, first served basis. How would we use this mechanism to support mixing between FixDrinks() and TripDialog()? Since both behaviors use the arms and the eyes, they obviously can't just be allowed to mix willy-nilly – this would result in strange behavior such as Trip snapping his arms erratically between the movements for making a drink and the dialog gestures. Specifying a goal conflict between FixDrinks() and TripDialog() would certainly prevent the erratic gesture mixing, but at the price of disallowing *any* mixing, preventing the combinatorial richness we're trying to achieve. Using the conflict mechanism, the only other option is to specify act conflicts between the physical acts used to move the arms in FixDrinks() and TripDialog(). This seems closer to the right answer, in that it constrains mixing only at the point where the behaviors are trying to make the body do two different things. But suspending the conflicting acts isn't what we want either, at least not in all cases. For example, the arm gesture in TripDialog() is happening in parallel with a physical act that moves the lips to actually speak the line. If the arm gesture suspended because of a conflict, the line would continue being spoken – then at some future point the conflict would be over, the gesture act would unsuspend and the now out-of-context gesture would occur. Of course you can imagine fixing any one case using other ABL constructs, perhaps in this case annotating the arm gesture as effect_only so that, if it did suspend, the step doesn't have to happen for TripDialog() to succeed (but what if we really want the dialog to be spoken with the gesture?). What we'd really like is some means for behaviors to reserve "parts" of the body they are going to use, to test whether a "part" is available, and, in the event of resource conflicts, to be able to either skip over the conflicting step (if it's not always necessary), fail the enclosing behavior, or wait for the resource to become available.

The body resource manager controls access to a set of body resources; a body resource is a body "part" or muscle that a behavior can control. The set of body resources is dependent on the details of the particular body an ABL program is controlling. The body resources used for *Façade* appear in Figure 6-1.

The current state of each body resource is stored in working memory in a BodyResourceWME. Each WME stores the current resource owner (if any), the priority at which the resource is owned, and an optional timeout indicating that the resource should automatically be released after a specified length of time. The owner of a body resource is a behavior. Within a BodyResourceWME, the owning behavior is represented by a reference to the reflection WME (a BehaviorWME) for the owning behavior. The reflection WME uniquely identifies the current owner; even if multiple instances of the same behavior appear in the ABT, each behavior instance has its own unique reflection WME.

legsBody – grants control over gross body movement, such as walking, sitting, turning to face a direction.
armL – grants control over the left arm.
armR – grants control over the right arm.
gaze – grants control over the gaze direction of the eyes.
gazeTorso – grants control over the twist of the torso for larger gaze direction changes.
voice – grants control over the lips.
faceExpressionBase – grants control over one of the three dimensions of facial control (they are summed to yield the current dynamic face expression).
faceExpressionMod – grants control over one of the three dimensions of facial control (they are summed to yield the current dynamic face expression).
facialMod – grants control over one of the three dimensions of facial control (they are summed to yield the current dynamic face expression).

Figure 6-1. Body resources in *Façade*

The primary body resource management behaviors are listed in Figure 6-2. A description of the details of RequestBodyResource, and what it means to *own* a resource, should make the rest of the body resource management behaviors clear.

RequestBodyResource(int resourceID, int resourcePriority, int timeout, int actionOnFailure) – requests a body resource at a specified priority.
ReleaseBodyResource(int resourceID) – releases a body resource.
BodyResources_IsOwner(int resouceID) – succeeds if the subgoaling behavior owns the specified resource, fails otherwise.

BodyResources_CleanupDemon(int resourceID) – a demon that releases the specified resource if the currently owning behavior disappears from the ABT without releasing the resource. This is an internal behavior of the body resource manager.
BodyResources_TimeoutDemon(int resourceID, int timeout, BehaviorWME owner) – a demon that releases a resource after the specified timeout period has elapsed. This is an internal behavior of the body resource manager.

Figure 6-2. Body resource management behaviors

RequestBodyResource requests a body resource at a specified priority. There are four cases to consider when a resource is requested. In case 1, the requesting behavior already owns the resource. A requesting behavior owns a resource if it is a recursive child of the current resource owner. This means that resource ownership is inherited down the ABT; any behavior within the subtree rooted at the behavior that originally grabs the resource, owns the resource. The RequestBodyResource behavior uses meta-ABL features to accomplish this recursive ownership test. Additionally, a current resource owner can grant ownership status to another behavior with the user property resourceOwner (user properties are described on page 97). The value of a resourceOwner property is a BehaviorWME; the subtree rooted at a goal marked with a resourceOwner property owns the resource if the value of resourceOwner is the current resource owner. Granting ownership in this manner is useful when the currently owning

behavior wants to spawn a subgoal elsewhere in the tree. Since the spawned goal is not a child of the current resource owner, there needs to be some way of granting it ownership. In case 1, where the requesting behavior already owns the resource, RequestBodyResource changes the timeout and resource priority if they are different from the current timeout and priority values and succeeds.

In case 2 of RequestBodyResource, there is no current resource owner. The appropriate BodyResourceWME is modified to make the requesting behavior the current resource owner with the specified priority and timeout.

In case 3, another behavior already owns the requested body resource but at a lower body resource priority than the requesting behavior. In this case the requesting behavior takes ownership of the body resource away from the current owner. RequestBodyResource fails the current owner as well as any goals to which ownership has been passed via the resourceOwner property, releases all resources owned by the failed behavior, cleans up any waiting clean-up demons and timeout demons, and modifies the appropriate BodyResourceWME to make the requesting behavior the current resource owner with the specified priority and timeout. RequestBodyResource of course makes use of meta-ABL functionality to accomplish all this.

In case 4, another behavior already owns the requested body resource at a higher priority than the requesting behavior. What happens to the requesting behavior depends on the value of the actionOnFailure parameter. There are two possible values for this parameter, fail and block. In the case of fail, the RequestBodyResource goal fails, while in the case of block, the current line of expansion blocks until the resource is free or owned at a lower priority than the priority of the request.

It is informative to note the differences between the body resource manager and the use of act conflicts. Consider the case of a higher priority behavior taking a resource away from a lower priority behavior. Instead of the higher priority physical act suspending the lower priority act, the lower priority resource owner is failed. It would be easy to add an additional parameter to RequestBodyResources that specifies whether the behavior losing the resource should suspend or fail. However, we've found in practice, at least with *Façade*, that suspending the behavior losing the body resource is rarely (if ever) the right thing to do. When a behavior is interrupted by losing a body resource, the interpretive context within which the behavior is running has been disturbed. If the interrupted behavior just starts up from where it left off, the resulting performance makes little sense. Instead, the context for the interrupted behavior needs to be reestablished. Sengers makes similar arguments in [Sengers 1998a; Sengers 1998b] when arguing for the need for transition behaviors. While with meta-ABL one can certainly author transition behaviors, within *Façade* we often instead just restart a behavior from the beginning, authoring "retry" behavior variants when appropriate. The "retry" variants of a behavior use episodic state to determine that the behavior was previously interrupted, and thus, when the goal is reattempted, that a behavior variant reestablishing the context while avoiding strict repetition (which, being robotic, would not be believable, unless the character is robot) is needed.

Another difference between the body resource manager and just specifying act conflicts is that the resolution of resource conflicts affects *the entire owing behavior* rather than just the conflicting step. This difference is particularly notable in the case of parallel behaviors in which, with act conflicts, all steps except for the conflicting act continue; with the body resource manager, the entire behavior is stopped by a resource conflict. Again, it would be a simple matter to extend the body resource manager to support associating resources with individual steps as well as with behaviors (one could

just use the already defined resourceOwner property) for cases in which this finer grained control is wanted. For *Façade* we just haven't felt the need for this capability, while we definitely have needed the ability for resource conflicts to result in entire behaviors failing.

The body resource manager also supports easy querying of the current body resource state. This allows the easy authoring of behavior variants that accomplish the same goal with different body resources. While in principle one could author similar query behaviors to test conflict declarations using meta-ABL [15], the point is that no matter how it's provided, the ability to reason about *future* resource conflicts is necessary to enable rich behavior blending with complex bodies; the Hap conflict mechanism on its own doesn't provide this.

An interesting area for future work is to push body resource management into ABL as a primitive language feature. One can imagine an extended form of the conflict specification sub-language plus built-in step annotations that together support a generalized resource mechanism (body resources plus more abstract goal resources). It is interesting to note that with the user property feature of meta-ABL, additional functionality provided as a package written in ABL looks an awful lot like primitive support for the functionality. This is similar to the way Em is written in Hap and yet looks like a primitive feature of Hap. Perhaps the best way to add general behavior management support to ABL is to provide it as a robust package written in ABL. In any event, regardless of how body resource management is implemented, future work on general support for body resource management requires more experience with controlling a variety of sensory-motor systems with ABL.

Beat Goals And Handlers

As discussed in Chapter 3, beats are the story-level building blocks for *Façade*. Beat-specific ABL behaviors provide the procedural knowledge necessary to carry out the beat. This section describes the organization of these beat specific behaviors.

Beat behaviors are organized around a collection of *beat goals*, the dramatic content the beat wants to communicate to the player through the beat performance, and a collection of beat-specific *handlers* that modify the performance of the beat in response to player interaction. Consider first the beat goals.

Beat Goals

The high-level beat goal specification for a beat appears in Figure 6-3. In this beat, Grace is hinting at the fact that there is something weird, perhaps a bit sick, about her obsessive decorating of their apartment. In the backstory, Grace studied art and design in college. She really wanted to be an artist, but a combination of a lack of self-confidence plus pressure from both Trip and her family to "do something practical" caused her to become

[15] It wouldn't quite be possible with current meta-ABL support. Meta-ABL provides access to the current state of the ABT as well as to conflict specifications; this would be enough to determine, in the case of a goal or act step, the set of signatures the goal or act conflicts with. But for sequential behaviors, meta-ABL does not currently provide access to the behavior continuation; this makes it impossible to compare the set of conflicting signatures with the future steps of the sequential behavior. However, since adding access to sequential behavior continuations is simple in principle, one can think of meta-ABL plus the conflict mechanism as *in principle* providing access to future act conflicts.

a designer in an advertising firm instead. Grace's obsessive decorating of their apartment is a frustrated outlet, but it really doesn't make her happy and has become a symbol in her mind for their sour marriage. In this beat Grace is trying to get the player to agree that something is "off" about her current decorating scheme, while Trip is trying to get the player to agree that it looks great and everything is fine. This beat is an instance of what we call an "affinity game" (see page 156 for more on the *Façade* story design).

Transition In (the no-subtopic version)

G: So, Player, I'm hoping you can help me understand where I went wrong with my new decorating (bitter laugh).

T: (pacifying tone) Oh, Grace, let's not do that.

Address subtopic, part 1 (armoire subtopic)

G: (sigh) You know, when I saw this armoire on the showroom floor, I thought it had such a clean, simple look to it...

T: It's a nice choice.

Address subtopic, part 2 (armoire subtopic)

G: ...but looking at it here in the apartment, it just looks like... (laugh half-lighthearted, half-bitter) a monstrosity!

T: (under-breath impatient sigh) uhh...

Wait Timeout (default version)

(G looks impatiently between the object and the player, T fidgets)

Transition Out (lean towards Grace affinity)

G: Ah, yes, I've been waiting for someone to say that!

Figure 6-3. Partial beat goal specification for the decorating beat

Beats generally have a transition-in beat goal responsible for establishing the beat context and relating the beat to action that happened prior to the beat, a transition-out beat goal communicating the dramatic action (change in values) that occurred in the beat as a function of player interaction within the beat, and a small number of beat goals between the transition-in and transition-out that reveal information and set up the little interaction game within the beat.

In the decorating beat, the transition-in introduces the ostensible topic of this beat: Grace thinks something's wrong with her decorating[16]. The particular transition-in shown in Figure 6-3 is the most "generic" transition, the one to use if the interaction priori to this beat didn't have anything to do with the room or decorating. Other transition-ins are available in this beat for cases in which the player has somehow referred to decorating just prior to this beat being sequenced, for example, by referring to an object associated with decorating such as the couch or the armoire[17].

The body of this beat, the two "address subtopic" beat goals, establish that Grace is unhappy with some aspect of her decorating, that Trip thinks it looks fine, and that

[16] The real topic is deeper than this but is communicated through the surface topic.

[17] "References" to an object can happen both verbally (e.g. the player types "I like the couch") and physically, by standing near an object and looking at it.

something is a bit weird about the whole situation. Additionally, these beat goals implicitly invite a comment from the player, agreement or disagreement with Trip or Grace. For this beat there are a number of different "address subtopic" beat goals for different decorating subtopics corresponding to different objects in the room. If the player hasn't referred to a specific object, one is chosen at random, otherwise the object referenced by the player (perhaps in an interaction just prior to this beat) is used.

During the "wait timeout" beat goal, Trip and Grace wait for a response from the player, in this case fidgeting nervously and glancing at each other. If the player doesn't respond within a certain timeout period, the lack of response is taken as a response. For this beat, a lack of response is taken as implicit agreement that the place looks fine and that there is nothing wrong with the decorating.

The transition-out communicates how the player's interaction within the beat has changed the story situation. One of the primary story values in *Façade* is the player's affinity with the characters, whether the player is allied with Trip, with Grace, or is neutral. Affinity in *Façade* is zero-sum – if the player moves towards positive affinity with Trip, the affinity with Grace becomes negative. For this beat there are three transition-outs, one each for the cases of the affinity changing towards Trip, one for towards Grace, and one for staying neutral. The transition-out in Figure 6-3 is the one for an affinity change towards Grace. In this case the player has agreed with Grace (e.g. "Yes, the armoire is ugly."), and, perhaps surprisingly, Grace is happy to hear someone say that.

The ABL code for the sequencing logic of these beat goals is shown in Figure 6-4.

```
parallel behavior BeatGoals() {

   with (priority 4) subgoal bAAt1N_ChooseTxnIn();

   with (priority 3, post,
      persistent when_fails,
      property isBeatGoal true)
   subgoal AddressSubtopic();

   with (priority 2, post,
      persistent when_fails,
      property isBeatGoal true)
   subgoal AddressSubtopic_part2();

   with (priority 1, post,
      persistent when_fails,
      property isBeatGoal true)
   subgoal WaitTimeout();

}
```

Figure 6-4. ABL code for the decorating beat goals.

This beat goal sequence (the priorities make the parallel behavior act like a sequential behavior[18]) represents the default logic for the beat, that is, the sequence of activity that would happen in the absence of player interaction. Of course, the whole point of an interactive drama is that the player can interact – thus there needs to be some mechanism

[18] Beat goals are declared to conflict with each other. This, plus the priorities, makes the parallel behavior act like a sequential behavior. Without the conflicts the beat goals would start executing in priority order, but mix as beat goals blocked on the execution of physical acts.

for incorporating interaction into the default beat logic. This is the job of *handlers*. The various annotations on the beat goals, plus the use of a parallel instead of sequential behavior, are there to facilitate the handlers' manipulation of the beat goals in response to player interaction.

Handlers

Handlers are demons responsible for handling player interaction. Each handler waits for some particular type of player interaction and "handles" it accordingly. Every beat specifies some beat-specific handlers; additionally, there are global handlers for handling interactions for which there is no beat specific response in the current beat. Handlers are meta-behaviors; they make use of reflection to modify the default logic of the beat. Handlers fall into two broad classes, mix-in handlers, which primarily mix a response into the current beat, and sequence handlers that primarily modify the sequencing logic of the current beat. Before diving into detailed examples of these two categories of handler, it is necessary to describe the general concept of *animation cues*, and four specific animation cues used with dialog.

Animation cues are tokens that can be associated with frames of animation data within the animation engine. When the associated animation data is played[19] in response to a primitive act in ABL, the animation engine sends any cues associated with animation frames back to ABL as the frames are played. Within ABL, an AnimationCueSensor has been defined; this supports the authoring of demon behaviors that wait on specific cues. Animation cues thus provide a mechanism by which ABL agents in the *Façade* world can "sense" when an agent has reached a specific point in accomplishing a physical action. The heaviest use of cues is for dialog animation. A dialog animation script contains the data for lip movements for a specific line of dialog, as well as information about the text associated with the lip movements and the sound file for the vocal performance of the dialog. Cues can be associated with specific words in a line of dialog[20]. For the purposes of handlers, the most important cues are BeatGoalGist, BeatGist, LetMeFinishOn, and LetMeFinishOff. A BeatGoalGist cue indicates that the meaning, or gist of a beat goal has been communicated. For example, a BeatGoalGist might be associated with the end of Grace's line in the transition-in beat goal in Figure 6-3. Trip's line, though an important part of the performance, is not absolutely necessary to communicate the "meaning" of this beat goal. A BeatGist cue indicates the point at which the meaning of the beat has been communicated. The "meaning" of a beat is the point at which the conflict within the beat has been firmly established, thus providing an interaction context for the player. The BeatGist in Figure 6-3 appears at the end of the "address subtopic part 2" beat goal. Trip's final line in this beat goal is not necessary to establish the conflict – that Grace is upset about the decorating and Trip is trying to placate her. The LetMeFinishOn and LetMeFinishOff cues are used to turn uninterruptible segments on and off. During an uninterruptible segment, the reaction to any player interaction is deferred until the end of

[19] To say that animation data is "played" means that the sequence of animation data frames is placed in the rendering pipeline, where it is dynamically combined with other sequences of animation data (multiple layers of animation data can play simultaneously), combined with procedurally produced animation, possibly procedurally tweaked, and finally rendered.

[20] Our dialog animation tool takes a line of dialog and turns it into a timed sequence of visemes. The animation engine morphs between visemes at playback. A line of dialog can be annotated to associate cues with the start or end of specific words; the dialog tool appropriately places the cues within the generated dialog script.

the segment. Uninterruptible segments may span beat goals or just span part of a beat goal. When uninterruptibility is off, the characters immediately interrupt themselves to respond to player interaction, in mid-word if they happen to be speaking dialog. Demons listen for the various dialog cues and update WMEs that store the gist status for each beat goal and for the beat, as well as whether the beat is currently interruptible or not.

With this basic understanding of dialog animation cues, we are now ready to give examples of interaction handlers. For the simplest example, consider first the handlers that choose the appropriate transition-out. Notice that in Figure 6-4 no transition-out appears in the default beat goal sequence. The transition-out only happens in response to interaction, so rather than including it in the default sequence, it is added to the default sequence by a handler. Even if the player says nothing during the beat, an implicit interaction happens during the WaitTimeout(). When the timeout period is exceeded, a timeout discourse act is produced (it's as if the player explicitly "said" nothing) and a handler responds to this discourse act by choosing the neutral transition-out. A transition-out handler for the case of affinity leaning towards Grace is shown in Figure 6-5.

```
sequential behavior ReactionHandler() {
   precondition {
     (ReactionWME rxnType == eDARxnType_TxnOut
         param1 == eTxnOut_LeanToGPA)
     beatGoals = (ParallelBehaviorWME signature == "BeatGoals()")
   }

   subgoal AbortBeatGoalAndSuspendBeatGoals();
   with (persistent when_fails, post, property isBeatGoal true,
      property isTxnOutBeatGoal true, priority 10)
     spawngoal TxnOut_LeanToGPA() at beatGoals;
   subgoal SucceedGoal("WaitTimeout");
   subgoal UnsuspendBeatGoals();
}
```

Figure 6-5. Transition-out interaction handler for the redecorating beat

A demon with a success test listening for interaction subgoals ReactionHandler() when an interaction occurs. The various preconditions of the many ReactionHandler() behaviors differentiate the handlers. Notice that the handler's precondition isn't directly testing for a discourse act, but rather is testing for a ReactionWME; the reasoning to choose a reaction (in this case a transition-out) given a discourse act (e.g. an Agree discourse act resulting from the player typing "Yes, the armoire is ugly") has already happened. The phase II pipeline of the natural language processing system performs this reasoning; this is described in more detail in Chapter 10. The handler's job, once a reaction has been chosen, is to manipulate the current beat goal logic so as to carry out the reaction. The first step of the handler is to abort the current beat goal and suspend all the beat goals. The meta-behavior AbortBeatGoalAndSuspendBeatGoals() takes care of this. Now the purpose for the various annotations on subgoal steps in Figure 6-4 can be made clear. The user property isBeatGoal uniquely identifies the currently executing beat goal within the ABT. By convention (part of the definition of beat goal), there is only one beat goal currently executing in the ABT at a time. The currently executing beat goal is aborted by failing it. The persistent when_fails annotation on beat goals means the goal will remain in BeatGoals() – after the interruption initiated by the handler, the aborted beat goal will be retried. The various alternative behaviors for a given beat goal make use of episodic state to appropriately choose a version of the behavior appropriate for whether this is the

first time the beat goal has been attempted or if the beat goal is being re-attempted after an interruption (and thus should do something different to reestablish context). In addition to aborting the current beat goal, BeatGoals() is suspended so that none of its steps will try to execute while the handler is modifying it. The handler then spawns the appropriate transition-out goal at BeatGoals() and
succeeds away the WaitTimeout(). Since the job of handlers is to modify the beat goal logic in response to interaction, they will often want to add new beat goals to the collection of beat goals, or succeed away a beat goal that has not happened yet. This is why BeatGoals() is a parallel behavior rather than a sequential behavior. Currently meta-ABL does not support modifying the continuation associated with a sequential behavior. "Faking" a sequential behavior with a parallel behavior gives handlers access to all of the beat goal logic, both the current goal and future goals. Because beat goals are annotated with persistent when_fails, succeeding a goal is a way to permanently remove it from the current collection of beat goals. Finally the handler unsuspends BeatGoals(). The new beat goal logic, which has been modified in response to player interaction (in this case a player interaction resulting in a transition-out), continues executing.

Now consider the case of a mix-in handler. Imagine that the player agrees with Grace before the beat gist. For example, imagine that after Grace's line in the transition-in of the redecorating beat ("So, Player, I'm hoping you can help me understand where I went wrong with my new decorating (bitter laugh)."), the player agrees with Grace ("Yeah, it looks bad"). Since the beat gist hasn't occurred, meaning that the beat conflict hasn't been established, it wouldn't make sense to choose the transition-out. In this case, a beat-specific mix-in occurs, during which Trip says "Well hold on, hold on, take a closer look, give it a chance to soak in a little." The determination that the Agree discourse act results in a mix-in rather than a transition-out is handled by phase II of the NLP. The decision hinges on whether the Agree occurs before or after the beat gist. The logic for the mix-in handler is similar to the transition-out handler in Figure 6-5. First the handler aborts the current beat goal (by failing it) and suspends BeatGoals() (the root of all the beat goals). Then the handler spawns a high-priority mix-in beat goal that will perform the reaction. Finally the handler unsuspends BeatGoals() and the beat goals continue executing, starting with the high priority mix-in. Unlike the handler in Figure 6-5, no future beat goals are succeeded away. The rest of the beat proceeds normally after the mix-in completes.

Global mix-in handlers respond to interactions that aren't handled within the current beat context. For example, imagine that during the decorating beat the player moves up to the bar and looks at it for a moment. This results in a (ReferTo Bar) discourse act being generated (player physical actions can also generate discourse). There is no beat-specific meaning to referring to the bar within the decorating beat, so a global handler responds to this discourse act with a global mix-in. The collection of global mix-ins is described in Chapter 10.

Uninterruptible segments, determined by the LetMeFinishOn and LetMeFinishOff dialog animation cues, are used to control interaction during a beat. First consider the case of *no* uninterruptible segments. In this case every beat goal can be interrupted at any point during its performance. This will be inappropriate for beat goals in which the characters deliver dramatically intense, important lines. Allowing them to be arbitrarily interrupted would destroy the dramatic intensity of the beat goal. The dramatic intensity of the line can be preserved by making it uninterruptible – any interactions occurring during the line are cached and processed after the line is over. Another reason to control interruptibility is to control authoring costs. Whenever a beat goal is aborted by a handler,

it will eventually be retried after all the higher priority goals added by the handler have completed. In order to maintain believability, we must prevent the characters from appearing robotic. It is particularly easy for language-using agents to appear robotic when they *exactly repeat* a phrase. For example, consider the following interaction.

> G: You know, when I saw this armoire on the showroom floor, I though it…
> P: I don't like it.
> T: Well hold on, hold on, take a closer look, give it a chance to soak in a little.
> G: You know, when I saw this armoire on the showroom floor, I thought it had such a clean, simple look to it.

The exact repetition of the first part of Grace's line makes her appear robotic. The obvious solution is to author alternate versions of behaviors for a given beat goal, using episodic state to select a "first-time" version of the behavior vs. "retry" versions. But since *Façade* doesn't use natural language generation or speech synthesis, all of this variation must be authored by hand. By making a line uninterruptible up to the beat goal gist, a response to an interaction is deferred until after the meaning of the beat goal has been communicated – thus, after a mix-in, the interrupted beat goal doesn't have to repeat. But uninterruptible segments may compromise a sense of responsiveness and agency. For *Façade* we've struck the following compromise between interruptibility and authoring costs.

- Most lines (beat goals) are uninterruptible to the beat gist.
- Long lines are broken into multiple shorter lines (multiple beat goals). Each of the shorter lines is uninterruptible (to the gist), but interruptions can still happen between lines. A short establishing phrase (e.g. "Anyway,…") may be used if an interruption occurs between the beat goals of a longer line. An example of breaking a longer line into multiple beat goals are the part 1 and 2 address subtopic beat goals in Figure 6-3.
- Mix-in goals sequenced by a handler are uninterruptible to the gist point (though a mix-in, like any beat goal, is in principle interruptible).

Finally, consider the case of sequence handlers, handlers that primarily modify the current sequence of beat goals. Within the decorating beat, a sequence handler handles references to objects related to the decorating beat. There are a number of variants of the address subtopic beat goals, each of which makes use of a different object to establish the beat conflict. The version of the address subtopic goals in Figure 6-3 uses the armoire – other versions use the rug, the couch, the paintings, the wedding picture, the balcony, the general arrangement, and Grace's trinket collection. If during the beat the player makes a reference, either physically or verbally, to any of these objects, the beat switches subtopics, rewinding to address the new subtopic. An example interaction illustrating a rewind appears in Figure 6-6.

In this interaction the player sits down on the couch near the end of the armoire version of AddressSubtopic(). Since the couch can also be used as a focus object by the decorating beat, the beat switches subtopics – the handler modifies BeatGoals() to include the new AddressSubtopic goals for the couch. This effectively rewinds the beat back to the beginning of AddressSubtopic() with a new subtopic object. The gray text indicates behavioral variation because of this being the second subtopic in the beat. Trip's lines are different in the case of a second subtopic to reflect that Grace is now complaining a second time about objects in the room, thus causing Trip to become more wheedling in his instance that everything is all right. Grace's line "Ugh, this couch…" is used as glue to connect the first subtopic to the second subtopic.

G:	(sigh) You know, when I saw this armoire on the showroom floor, I thought it had such a clean, simple look to it...
T:	It's a nice choice.
P:	**(sits down on the couch)**
G:	**Ugh, this couch…**
G:	(sigh) You know, this corner felt like it needed something... big... and bold...
T:	**There's nothing wrong with it.**
G:	(a bit bitter) ...but now I can see how I should have chosen a nice, comfortable little... loveseat.
T:	**(trying hard to be upbeat) Grace, come on, it looks fine...**

Figure 6-6. Example decorating beat interaction illustrating a subtopic rewind

The handler that switches subtopics appears in Figure 6-7. Again, the decision of what reaction to perform has already been made by the NLP system. The decision is communicated to the handler through a ReactionWME, which in this case indicates that we are switching subtopics and the subtopic we're switching to. The test for a BeatFlagWME in story memory (an example of a WME test against a memory other than the character's private working memory) determines whether we have already done a subtopic switch. For the decorating beat, subtopic switches can only happen once – if one has already happened, then this handler's precondition isn't valid and a different handler will do something else. Next, after modifying various WMEs to indicate that a subtopic switch has occurred, the handler *succeeds all the beat goals* away[21]. This is different than the other handlers we've looked at so far, in which only the current beat goal is aborted. This handler is erasing the current beat goal logic and writing brand new logic. In this case it adds back the two AddressSubtopic beat goals as well as the WaitTimeout. When the beat goals unsuspends, address subtopic will start at the beginning with a new subtopic. Beat goal behavior variants use the WME state written by the handler to determine that a subtopic switch has occurred and appropriately perform any glue and dialog variants.

[21] The behavior which succeeds all the beat goals first adds a dummy beat goal to BeatGoals() so that it doesn't immediately succeed.

```
sequential behavior ReactionHandler() {
    precondition {
      (ReactionWME rxnType == eDARxnType_SwitchSubtopic
          param1 :: newSubtopic)
      { StoryMemory !(BeatFlagWME sVal == "SwitchSubtopic") }
      beatGoals = (ParallelBehaviorWME signature == "BeatGoals()"
    }

    <Modify WME state to indicate the subtopic switch>

    subgoal SucceedAllBeatGoalsAndSuspendBeatGoals();

    with (priority 3, post, persistent when_fails,
       property isBeatGoal true)
    spawngoal AddressSubtopic() at beatGoals;

    with (priority 2, post, persistent when_fails,
       property isBeatGoal true)
    spawngoal AddressSubtopic_part2() at beatGoals;

    with (priority 1, post, persistent when_fails,
       property isBeatGoal true)
    spawngoal WaitTimeout() at beatGoals;

    subgoal UnsuspendBeatGoals();
}
```

Figure 6-7. Rewind-subtopic interaction handler for the decorating beat.

Interaction = Beat Goals + Handlers

Our authoring strategy for handling within-beat discourse act interaction is specify the non-interactive beat goal logic (what we want the beat to accomplish) and a collection of handler behaviors that modify this default logic in response to interaction. In order to modify the default logic, handlers make use of meta-ABL functionality in order to modify the ABT state. While handlers can in principle arbitrarily modify the ABT state, most handlers fall into one of two general classes – mix-in handlers that add an additional beat goal in the middle of a beat while keeping the rest of the sequencing the same, and sequencing handlers that more radically reorganize the beat goal sequence. The ability to factor behavior into non-interactive sequences organizing the longer-term temporal structure of a character's behavior and meta-behaviors that modify this longer-term temporal structure is a powerful idiom enabled by ABL's support for reflection.

Performance Behaviors

Beat goals and handlers deal with the high-level logic of a beat and the effect of discourse act-level interaction within the beat logic. But there is a more detailed level of description, the level at which the precise performance of beat goals is specified and sub-discourse act interaction (such as moving around a bit) is handled. This is the level of performance behaviors. Consider the detailed performance spec in Figure 6-8 for a single line of the redecorating beat.

Grace: ...but looking at it here **in** the apartment, it just looks like... (laugh half-lighthearted, half-bitter) **a** monstrosity!

Grace physical performance:

- Throughout the line Grace physically adjusts to face the player.
- On the bold words she slightly nods her head and barely raises both arms.
- At the beginning of the line she looks at the armoire with a frown, changing to a serious expression a short way into the dialog.
- At the beginning of the line her mood becomes anxious.

Trip physical performance:

- Throughout the line Trip physically adjusts to face the player.
- At the beginning of Grace's line he turns (just his head and eyes) to look at Grace.
- A short way into Grace's line he gets an impatient expression on his face.
- A short way into Grace's line he reacts with an anxious reaction.

Figure 6-8. Detailed performance specification for one line of the decorating beat

Grace and Trip are tightly coordinated during this line – both of them physically react during the line as well as respond to sub-discourse act player interaction (continuing to face the player as the player moves around). In order to coordinate on performing this line they synchronize on a joint goal. By entering into a joint goal together, they establish a shared behavioral context during which they perform this line. A simplified version of Grace and Trip's joint behaviors for this line appears in Figure 6-9.

Consider first the details of Grace's joint behavior. The first subgoal, TryToKeepFacingSprite, is a performance utility behavior that periodically adjusts Grace in order to keep facing a specific sprite, in this case the player. The first argument, 0, is the priority with which to request the required body resources (legsBody) to perform the physical acts to adjust Grace's direction. The relatively low priority of 0 means that these turning adjustments should yield to most other behaviors that might request the legs.

The second subgoal, SetPerformanceInfo, sets up the emphasis arm and head movements on the words "in" and "a", and her initial frowning expression and the change to a serious expression partway into the line. The initial parameter, 40, is the priority at which this behavior should request body resources. One way of setting up these actions, such as the emphasis arm and head movements, would be to associate dialog animation cues with the appropriate words and spawn demons that wait on these cues; when triggered, the demons would perform the appropriate actions. However, given the vagaries of the animation system and ABL existing in separate processes, and the communication latency to communicate across these processes, ABL may take as long as a couple of frames (at 30 frames/second) to respond to an animation cue. Sometimes we want the gesture associated with a specific word to be frame-level accurate, for the gesture to start exactly as the word starts being spoken. So down in the guts of the SetPerformanceInfo behavior it actually makes use of a peculiar physical act, the setPerformanceInfo act, to tell the animation engine to perform specific primitive acts on specific animation cues. In a sense the setPerformanceInfo act allows ABL to give the animation engine a little plan, telling it what future actions to take on cues. The animation engine performs these actions with frame-level accuracy. ABL is still responsible for deciding what actions to perform on what cues. And the setPerformanceInfo act is still fully abortable – if the enclosing behavior succeeds or fails out of the tree, the act aborts,

resulting in the animation engine "plan" being cleared. The cues that the setPerformanceInfo act sets up actions for are still communicated to ABL, so ABL demons can act on them as well. Animation cue demons (as we will see in Trip's behavior) are used for performing actions that do not require frame-level accuracy or for complex actions consisting of a number of physical acts. The SetPerformanceInfo behavior actually uses a mixture of performing actions in ABL and passing future cue/action pairs to the animation engine.

Grace's behavior

```
joint parallel behavior AddressSubtopic_part2_Line1() {
    teammembers Trip, Grace;

    with (persistent, team_effect_only)
      subgoal TryToKeepFacingSprite(0, player);
    with (priority_modifier 1, ignore_failure, team_effect_only)
      subgoal SetPerformanceInfo(40, <performance args>);
    with (priority_modifier 2)
      subgoal SetMood(eMood_anxious, eMoodStrength_low, 1, 0, 8.0f);
    subgoal DoDialog(70, <dialog args>);
}
```

Trip's behavior

```
joint parallel behavior AddressSubtopic_part2_Line1() {
    teammembers Trip, Grace;

    with (persistent, team_effect_only)
      subgoal TryToKeepFacingSprite(0, player);
    with (priority_modifier 1, ignore_failure, team_effect_only)
      subgoal SetPerformanceInfo(40, <performance args>);
    with (priority_modifier 1, ignore_failure, team_effect_only)
      subgoal PerformLittleActionAtDialogCue(10, cue_DialogUnderway,
      spouse, eActionType_react, eActionTone_anxious,
      eActionStrength_low);
}
```

Figure 6-9. Joint performance behaviors for one line of the decorating beat

The third subgoal in Grace's behavior, SetMood, sets up an anxious mood. Setting a mood results in a mood performance being mixed in with the rest of the character's performance for the duration of the mood. In this case the anxious mood has been given a timeout period of 8 seconds – at the end of this time the anxious mood will disappear. Every combination of mood and mood intensity has its own mood performance. For the low intensity anxious mood, a character will have an anxious look on their face, will periodically avert their eyes from whatever they are looking at, and will periodically cross their arms if they are not holding anything and their arms are not busy. Like all the performance utility behaviors in the system, mood performance makes use of the body resource manager to facilitate behavior mixing. For example, if the mood system decides it wants to perform the current low-intensity anxious mood just as the character happens to be performing an arm gesture, an appropriate variant mood performance behavior can be selected by querying the body resource manager. In this case that the arms aren't free, the anxious mood will be performed with just the facial expression and averting the eyes. Mood is used to establish a background performance that lasts for the duration of a performance behavior, and potentially crosses performance behaviors or beat goals.

The final subgoal in Grace's behavior, DoDialog, actually performs a line of dialog. The relatively high body resource priority of 70 means that DoDialog will tend to trump other behaviors that may request a conflicting resource, in this case the voice resource.

Note that Grace and Trip's performance behaviors are both parallel behaviors. All the steps are performed simultaneously, with priorities used to start certain steps first. For example, in Grace's behavior, SetMood is started first, then SetPerformanceInfo, then the rest of the steps. SetPerformanceInfo has to be started before DoDialog in order to ensure that any cue/action pairs have been established in the animation engine before the cue-producing behavior begins generating cues.

In Grace's behavior two different behaviors are controlling the facial expression. SetPerformacneInfo establishes a change from a frown to a serious expression, while mood performance wants to perform an anxious expression. The animation engine provides multiple layers or channels for controlling facial expression. The three facial expression channels appear as the last three body resources in Figure 6-1. Expressions in these three channels are combined to yield a total composite expression. Thus, in this case, Grace will start the line with an anxious frown on her face, followed by an anxious serious expression. If her mood was angry rather than anxious, then the frown-followed-by-serious expression established by SetPerformanceInfo would combine to become an angry frown followed by an angry serious expression. It is important to note that expressions are not static – expressions are actually procedural facial animations. So when Grace performs an anxious frown followed by an anxious serious expression, this doesn't mean that her face morphs from one wooden expression to another; rather, her face moves from one pattern of small facial movements (frowning movements) to another (serious movements). Additionally, the mood performance behaviors make use of expression alternatives as they periodically perform the mood – in this case, over time, different anxious facial expressions will layer on top of the frown or serious base.

Trip's performance behavior is similar to Grace's, except that, since he's not speaking, his doesn't contain a DoDialog act. Additionally, the cue/action pairs he establishes with SetPerformanceInfo listen not to his own dialog cues (since he's not speaking), but rather to Grace's cues. Thus his performance is a reaction to what he's "hearing" Grace say. His third subgoal, PerformLittleActionAtDialogCue, introduces a new performance utility behavior. *Little actions* are short term performances that temporarily change the facial expression, as well as performing actions such as rapid blinking, averting the eyes, or tilting the head. High intensity little actions may actually step back, gesture, or temporarily turn away from the object or character currently being looked at. Facial expression modifications make use of the third facial expression channel, facialMod – facial expression modifications initiated by a little action thus combine with the mood and base facial expressions. Like setting a mood, little actions can either be performed on a dialog cue (a demon waits for the cue), or directly subgoaled. Also, like moods, little actions are more complex than a single atomic physical act. Little action and mood performance behaviors select and perform over time a number of physical actions, making use of the body resource manager to appropriately mix with other behaviors.

For *Façade*, the default joint goal negotiation uses team_needed_for_success (set using the ABL compiler pragma joint_goal_success_negotiation). In order for the joint performance goal to succeed, both goals (and thus both behaviors) must succeed. Note that all the steps of Trip's behavior are annotated with team_effect_only. Like effect_only, this means that it is not necessary for the step to succeed (or even to be tried) in order for the enclosing behavior to succeed. Since all the steps in Trip's behavior are

annotated in this manner, then the behavior could succeed immediately without trying any of the steps. Since the default negotiation protocol requires that the goals of all participating team members must succeed before the goal succeeds, if the steps of Trip's behaviors were marked effect_only, the behavior would immediately freeze its subtree (thus removing all the effect_only steps), initiate a success negotiation, and wait for Grace's behavior to finish before succeeding. But this behavior isn't what we want – we want Trip to continue his performance until Grace is done with her performance, but succeed immediately when Grace succeeds. Team_effect_only provides this functionality. As described on page 80, the success of steps marked team_effect_only is not necessary for the local version of a joint behavior to succeed, but the steps continue to execute until the entire team negotiates success. So Trip will immediately initiate a success negotiation, but continue executing the steps of his performance behavior until Grace signals that her performance behavior has succeeded; Grace's performance behavior succeeds once her mood has been set and dialog completes. This is the general idiom employed for all line-level joint performance behaviors. The non-speaking character executes all team_effect_only steps, thus continuing their performance within the tight context of that particular line until the speaking character finishes.

Staging and Long-term Behaviors

The body resource section of this chapter argued that combinatorial behavior mixing is a source of performance richness for believable agents. Within the performance behaviors above, some mixing is happening as little actions, moods, dialog, responses to sub-discourse act interaction (e.g. tracking the player) and cue/action pairs initiated by setPerformanceInfo all mix. But another class of beat behaviors provide longer term, larger scale mixing – *cross-beat behaviors*. These are behaviors that cross beat goal, and potentially beat, boundaries. An important class of cross-beat behaviors are *staging behaviors*, which characters use to move to dramatically significant positions (e.g. close or far conversation position with the player or another character, into position to pickup or manipulate another object, etc.). Staging requests are generally initiated by beat goals. For example, a request to move to close conversation position with the player might be initiated by the first beat goal in a beat. Rather than subgoaling the staging goal directly from the beat goal, it is spawned to another part of the ABT so that it can continue beyond the duration of the beat goal. After the first beat goal completes its behavior, the character can pursue other beat goals as the agent continues to walk towards the requested staging point.

Long-term behaviors make use of multiple staging behaviors to accomplish some long-term task. For example, the FixADrink behavior stages to the bar, engages in various actions behind the bar (e.g. picking up bottles, running the blender), stages back to the player, and offers the player the newly made drink. The body resource manager is used to coordinate the mixing of both staging and long-term behaviors with beat goal performance behaviors. At any time during a cross-beat behavior, beat goals and handlers can use reflection to find out what cross-beat behaviors are currently being pursued, and succeed or fail them if the cross-beat behaviors are inappropriate for the current beat goal's or handler's situation.

Behavioral mixing at the level of staging and long-term behaviors provides another dimension to the performance. The same beat goal may now be performed close to the player, while walking across the room, while standing behind the bar, etc. Gestures may

be elided or added depending on how arm activity is mixing. This is happening on top of the mixing of various performance subsystems (e.g. mood, little actions, cue/action pairs) within a performance behavior, within the context of the longer-term beat goal logic, which is itself dynamically modified by discourse act reaction handlers, all within the context of the longest term story structure maintained by the drama manager.

Beat Logic and Joint Goals

The section Autonomy and Story-Based Believable Agents on page 40 made the argument that the model of strongly autonomous agents infrequently coordinated by a drama manager is inappropriate for story-based believable agents. Such agents frequently and tightly coordinate to carry out dramatic action. The argument concluded that what one really wants is a way to directly express coordinated character action at a level above the individual characters. Within *Façade*, the beat goals and handlers for a given beat are precisely this shared logic for multi-character, dramatic coordination. The beat goals and handlers execute in one of the characters – as the coordinator executes the joint performance behaviors within beat goals, joint goal entry is negotiated in the non-coordinator, the goal appears in the non-coordinator's ABT (at the ABT root), and both characters independently execute their joint performance behaviors, re-synchronizing on goal exit. The particular joint goal idioms used in *Façade* may lead to some misunderstandings about the nature of ABL's joint goal mechanism. I would like to address these potential misunderstandings here.

One possible misunderstanding is that joint goals are about "one character controlling the other(s)". In the particular case of *Façade*, there may be the sense that the coordinator is somehow special, or more important than the other agent. In fact, the choice of which character is the coordinator is completely arbitrary. The whole "specialness" of the coordinator can be diffused by creating a third, bodiless character (a disembodied mind) to execute the beat goals and handlers. This "character" would initiate the joint goals; Trip and Grace would perform their independent versions of the joint behaviors while the "disembodied mind" would execute it's own empty versions of the joint behaviors, waiting for Trip and Grace to finish (team_needed_for_success would take care of this).This arrangement makes clear that there is nothing special about any one of the characters, but rather that there is some shared logic at a level above the individual characters.

If one agrees that there is no "special" character controlling the rest, another potential misunderstanding is that joint goals are about "building a single mind that controls all the characters". The concern of course is that this "one-mind" approach obviates all the good things about the agent metaphor, requiring the author to take care of all the moment-by-moment behavioral details of a bunch of characters in a single, monolithic program. But, as argued on page 89, a team of ABL agents coordinated by joint goals is neither a one-mind nor many minds, but a variably coupled multi-mind; joint goals are a way to strike a middle-ground between the two extreme polls. An entered joint goal provides a coupling point between multiple ABTs, through which cascading within-ABT effects can additionally cascade *across* ABTs. But there will still be large chunks of individual ABTs that, through the vagaries of the ABTs' momentary topology and conflict specifications, are isolated from the cascading success, behavior, suspension or unsuspension effects entering through the coupling point of a joint goal. In the particular case of *Façade*, each character autonomously executes their own performance utility behaviors, staging

behaviors and long-term behaviors (as well as lower level utilities behaviors such as the body resource manager). As joint performance behaviors come and go, they will both influence and be influenced by these autonomously executing behaviors.

As a multi-mind enters joint goals more and more frequently, it does approach a one-mind extreme. In *Façade*, Trip and Grace do enter joint goals fairly frequently, typically a couple of times per beat goal. While still not a one-mind, they are probably on the one-mind side of the middle point in this spectrum. However, there is nothing about the generic structure of ABL requiring behaviors to be organized according to the idioms described in this chapter. There is much interesting future work to develop idioms for larger teams, for simultaneous multi-team commitments, and for more loosely coupled teams.

CHAPTER 7
EXPRESSIVE AI: AFFORDANCES AND SEMIOTICS

Affordances

As described in Chapter 4, while art practice focuses on the negotiation of meaning as mediated by the art object, AI research focuses on internal system structure and the interaction between system and environment. Expressive AI simultaneously focuses on the negotiation of meaning *and* the internal structure of the AI system. These two apparently disparate views are unified through the concept of affordance: negotiation of meaning is conditioned by interpretive affordances while the internal structure of the AI system is conditioned by authorial affordances.

In Chapter 2, affordance is introduced in the context of a neo-Aristotelian model for interactive drama; the model made the prescriptive claim that agency is maximized when material and formal affordances are balanced. The notion of affordance was first suggested by Gibson [Gibson 1977; Gibson 1979] in his theory of perception and was later re-articulated by Norman [Norman 1988] in the field of interface design. For Gibson, affordances are *objective*, actionable properties of objects in the world. For an animal to make use of the affordance, it must of course perceive it in some way, but for Gibson, the affordance is there whether the animal perceives it or not; an unperceived affordance is waiting to be discovered. For Norman, affordances become *perceived and culturally dependent*. That is, rather than viewing the relationship between sensory object and action as an independent property of the object+animal system, this relationship is contingent, dependent on the experiences of the perceiver within some cultural framework. For example, for a person who has spent the last 10 years using the web, blue underlined text now affords an action, clicking with a pointing device, with the expectation that this clicking will "follow a link" to another information node. If blue underlined text is used in a different interface merely as a way to emphasize text, this is likely to generate confusion because the hypothetical interface is violating an affordance. It is this second notion of contingent affordance that I use here. But note that though affordances are contingent, they are not arbitrary – affordances are conditioned by the details of human physiology (what we can sense, how our bodies move), by cultural memory, and by the perceivable physical properties of objects. While new affordances can come into existence, as illustrated by the link-following affordance of blue underlined text, these innovations are conditioned by earlier affordances (e.g. the physical affordances of computer mice) and take active cultural work to establish.

Interpretive Affordance

Interpretive affordances support the interpretations an audience makes about the operations of an AI system. In the conversation model of negotiated meaning (page 63), it is the interpretive affordances that condition the meanings negotiated between artist and audience. Interpretive affordances provide resources both for narrating the operation of the system, and additionally, in the case of an *interactive* system, for supporting intentions for action.

For AI-based art, narrative affordances support the audience in creating a story about the operation of the piece and how this operation relates to the artist's intention. For example, imagine having *Office Plant #1* (Appendix B) on your desk. The name, plus the physical form, prepares one to view the sculpture as a plant – it has identifiable parts that metaphorically relate to the stem, flower, and leaves of biological plants. The wooden box of the base, hammered finish of the flower, and whimsical piano-wire fronds topped with crinkled, copper-foil-wrapped spheres, give the plant a non-designerly, hand-built look that communicates that it is neither a consumer electronic toy nor serves any functional purpose. Yet it is clearly a machine – it hums quietly while operating, moves very slowly (the motion is visible only if you watch patiently), and, when returning to the desk after an absence, is sometimes in a different configuration than it was left in. The plant starts moving when email is received; over time one can notice a correlation between the plant's physical poses and the email received. All of the perceived features of the plant, the materials used and the details of fabrication, the physical form, the temporal behavior, the relationship between this behavior and email, constitute the narrative affordances, the "hooks" that the plant's owner uses to make sense of the plant, to understand the plant in relationship to themselves and their daily activity.

For interactive art, intentional affordances support the goals an audience can form with respect to the artwork. The audience should be able to take an action and understand how the artwork is responding to this action. This doesn't mean that the artwork must provide simple one-to-one responses to the audience's actions. Such simple one-to-one responses would be uninteresting; rather, the poetics of the piece will most likely avoid commonly used tropes while exploring ambiguities, surprise, and mystery. But the audience should be able to understand that the system is responding to them, even if the response is unexpected or ambiguous. The audience should be able to tell some kind of unfolding story about their interaction with the work. Both the extremes of simple stereotyped responses to audience interaction making use of well-known tropes, and opaque incoherence with no determinable relationship between interaction and the response of the art work, should be avoided. *Subjective Avatars* (Appendix A) dynamically manipulate the presentation of the world in such a way as to change the player's disposition towards different courses of action. This can be understood as an explicit, system-level manipulation of intentional affordances. The neo-Aristotelian model of Chapter 2 can be understood as providing guidance for organizing intentional affordances in the context of interactive drama, further dividing intentional affordances into the two categories of material and formal affordance and arguing that they should be balanced.

A concern with interpretive affordances is often alien to AI research practice. Though the role of interpretation is sometimes discussed (e.g. the Turing test is fundamentally about interpretation [Turing 1950], Newell's knowledge level is an attribution made from *outside* an AI system [Newell 1982]), most often AI systems are discussed in terms of *intrinsic* properties. But for artists, a concern with interpretive affordance is quite familiar; negotiating meaning between artist and audience is central to artistic practice. Expressive AI adopts this concern within the context of AI-based art. But Expressive AI also adopts a concern for the internal functioning of the artifact from AI research practice.

Authorial Affordance

The authorial affordances of an AI architecture are the "hooks" that an architecture provides for an artist to inscribe their authorial intention in the machine. Different AI

architectures provide different relationships between authorial control and the combinatorial possibilities offered by computation. Expressive AI engages in a sustained inquiry into these authorial affordances, crafting specific architectures that afford appropriate authorial control for specific artworks.

This concern with the machine itself will be familiar to AI research practitioners. However, AI research practice often downplays the role of human authorship, focusing on the properties of the architecture itself independent of any "content" authored *within* the architecture. Multiple architectures are most often compared in a content-free manner, comparing them along dimensions and constraints established by theories of mind, or theories of brain function (not necessarily at the lowest, neuron level), or comparing their performance on established benchmark problems. For Expressive AI, the concern is with how the internal structure of the machine mediates between authorship and the runtime performance.

A focus on the internals of the machine itself is often alien to current electronic media practice. In keeping with the conversation metaphor, the internal structure of the machine is generally marginalized. The machine itself is considered a hack, an accidental byproduct of the artist's engagement with the concept of the piece.

> One might generalize in this way (with apologies to both groups): artists will kluge together any kind of mess of technology behind the scenes because the coherence of the experience of the user is their first priority. Scientists wish for formal elegance at an abstract level and do not emphasize, or do not have the training to be conscious of inconsistencies in, the representational schemes of the interface. [Penny 2000]

In discussions of electronic media work, the internal structure of the machine is almost systematically effaced. When the structure is discussed, it is usually described at only the highest-level, using hype-ridden terminology and wishful component naming (e.g. "meaning generator", "emotion detector"). At its best, such discursive practice is a spoof of similar practice within AI research, and may also provide part of the context within which the artist wishes her work to be interpreted. At its worst, such practice is a form of obfuscation, perhaps masking a gap between intention and accomplishment, the fact that the machine does not actually do what is indicated in the concept of the piece.

Yet it is nonetheless the case that an artist's concern with the coherence of the audience experience, with the crafting of interpretive affordances, is entirely appropriate – creating an audience experience is one of the primary reasons the artwork is being made in the first place.[22] So why should an artist concern herself with authorial affordances, with the structural properties of the machine itself? Because such a concern allows an artist to explore expressive possibilities that can only be opened by a simultaneous inquiry into interpretive affordance and the structural possibilities of the machine. Interpretive and authorial affordances are coupled – a concern with the machine enables audience experiences that aren't achievable otherwise.

[22] Of course, the motivations for art making are complex. An artist is often working out ideas or concerns; creating the artwork becomes a way of thinking by doing, just as important to the artist as the creation of an audience experience. In the extreme, the artist may be entirely concerned with the process of making the piece, or with the private working out of a theory or symbology, with little or no concern with how the work will be received by others. But even in the extreme case of a private art, the artist may be viewed as her own audience, with a rapid fluctuation or vibration occurring between interpretation and authorship.

Combining Interpretive and Architectural Concerns

The splitting of AI-based art practice into interpretive and authorial concerns is for heuristic purposes only, as a way to understand how Expressive AI adopts concerns from both art practice and AI research practice. Expressive AI practice combines these two concerns into a dialectically related whole; the concerns mutually inform each other. The "interface" is not separated from the "architecture". In a process of total design, a tight relationship is maintained between the sensory experience of the audience and the architecture of the system. The architecture is crafted in such a way as to enable just those authorial affordances that allow the artist to manipulate the interpretive affordances dictated by the concept of the piece. At the same time, the architectural explorations suggest new ways to manipulate the interpretive affordances, thus suggesting new conceptual opportunities. Thus both the artist's engagement with the inner workings of the architecture and the audience's experience with the finished artwork are central, interrelated concerns for Expressive AI.

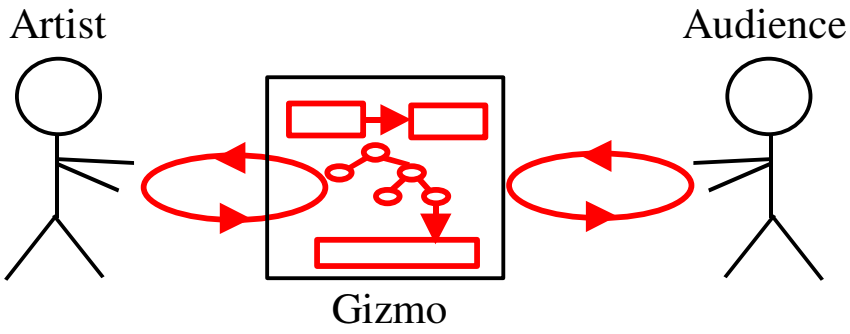

Figure 7-1. Architecture and audience experience both central for Expressive AI

The AI-based artist should avoid architectural elaborations that are not visible to the audience. However, this admonition should not be read too narrowly. The architecture itself may be part of the concept of the piece, part of the larger interpretive context of people theorizing about the piece. For example, one can imagine building a machine like *Terminal Time* in which some small collection of historical narratives have been prewritten. The narrative played is determined by a hard-coded selection mechanism keyed off the audience polls. For any one audience, the sensory experience of this piece would be indistinguishable from *Terminal Time*. However, at a conceptual level, this piece would be much weaker than *Terminal Time*. A *Terminal Time* audience is manipulating a *procedural process* that is a caricature of ideological bias and of institutionalized documentary filmmaking. The operationalization of ideology is critical to the concept of the piece, both for audiences and for artists and critics who wish to theorize the piece.

An Introduction to Semiology

Structuralist semiotics, or semiology, provides an analytic framework for further cashing out the notion of interpretive and authorial affordance, and their relationship to both art practice and AI research. Here I provide a *brief* (and hence inevitably simplified) explication of semiology. By semiology, I mean the semiotic tradition following

Saussure's General Linguistics [Saussure 1974], and explicated by thinkers such as [Hjelmslev 1961; Barthes 1967]. The treatment in this chapter most closely follows Barthes [Barthes 1967; Barthes 1972]. Readers already familiar with semiology should skip ahead to page 131.

Semiology is concerned with the phenomenon of meaning, that is, how it is that something (e.g. a mark on a page, an article of clothing, a dish in a meal), can have meaning for somebody. The fundamental unit of meaning is the *sign*, consisting of two parts, the *signifier* and the *signified*. The signifier is the uninterpreted object or sensory impression that, by convention, means something. The signified is the meaning, which is always a mental representation. For example, the following literal marks on the page, "**cat**", are a signifier, and the mental image that these words summon to mind, are the signified. Or, in the language of highway codes, the color red is a signifier, and the mental image of stopping a vehicle, is the signified. But these two faces of the sign never appear separately. When a reader fluent in English is confronted with the physical markings "**cat**", the visual experience of the markings and the mental image of a cat occur simultaneously, and only with special effort can be pried apart. And it is impossible to express the mental image or concept of a cat without signing, that is, without producing an appropriate signifier, such as the markings "**cat**", or a drawing of a cat, or imitating the sound of a cat, or producing a live cat. Thus a sign, though composed of two parts, is a unity.

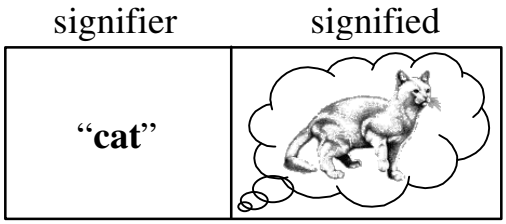

Figure 7-2. A sign for cat

Note that the signified is not a physical object, but a mental image or concept. The markings "**cat**" does not signify the physical, tiger-striped, brown-patched, purring animal lying on my couch, but rather a mental image. Objects may sign, that is, be the signifier of a sign, but are not themselves directly signified. Signs do not provide some kind of direct access to reality, but participate only in endless chains of signs.

Signs join together two planes, the *plane of expression* and the *plane of content*. Signifiers are drawn from the plane of expression, signs from the plane of content. Each plane is divided into two strata, *form* and *substance*. Substance is the undifferentiated mass that is shaped by form. Signs simultaneously make distinctions in the substance of the plane of expression and the substance of the plane of content. The substance of the plane of expression is the extra-semiotic material out of which signifiers are formed. The form of expression consists of the purely relational rules that organize the substance. For example, for spoken speech, the substance of expression is pure, non-functional, phonetic sound, while the form of expression is the rules governing speech. For the vestimentary code, which governs the wearing of clothes, the substance of expression is the material out of which clothes are made (individual articles of clothing are already signs in the vestimentary system, and thus have already formed the two planes). The substance of the plane of content is a sea of possible meanings. For example, for the vetimentary code, this sea of possible meanings includes all the meanings that clothes can signify. The form

of the plane of content is the rules and structures organizing these meanings, as imposed by the sign system. For both the planes of expression and content, it is difficult to say much about substance, since to even be able to identify it means that it has already been structured by form.

Semiology takes as its starting point linguistics – the prototype of all sign systems is spoken and written language. The semiological project is to extend notions of linguistic meaning to *all* meaningful phenomena, that is, to view the entire world, and all human action within it, as participating in generalized languages. Thus one may speak of the food system, or the car system, or cinema, or the pet system, as generalized languages, and seek to analyze the rules governing sign production in these languages. This generalized semiological sign has the interesting property that, unlike linguistic signs, its substance of expression has a non-signifying, utilitarian, technical function. For example, in the car system, a physical automobile serves the technical function of transporting people from place to place, in addition to serving as a sign. Such signs, which both signify and have a technical use, are *sign-functions*. Sign-functions are "the very unit where the relations of the technical and the significant are woven together." [Barthes 1967: 42]

Signs don't just occur individually, but participate in sign systems. Signs can appear in larger spatial and temporal configurations, as, for example, linguistic signs can appear in sentences and larger texts. Such configurations of signs are *syntagms*. Syntagmatic variation is the realm of freedom within sign systems – while users of a sign system cannot change the related elements (signifier and signified) of individual signs, they are free to assemble signs into syntagms. But this freedom isn't absolute. Combinational rules place constraints on how signs may participate in a syntagm, as, for example, syntactic rules govern the possible combinations of linguistic signs within a sentence, or the rules of fashion govern the combination of articles of clothing (sign-functions) in an outfit. Syntagms themselves become complex signs signifying a complex meaning. A movie, for example, is a temporal syntagm, obeying the combination constraints of the language of cinema (or some genre-specific sub-language), which as a whole conveys a meaning.

Signs, as well as participating in the actual structure of realized syntagms, participate in a potential structure of associative fields, the *paradigm*. Each associative field, or category, contains signs that can be substituted in the same role within syntagms, but with different meanings. For example, in the food system, syntagms are specific meals, while the paradigm consists of foodstuffs grouped into categories such as entrees, desserts, salads, where each category corresponds to a role within a specific meal. To construct a specific meal, one chooses sign-functions (foodstuffs) from the different categories, in view of its meaning relative to the other members of the category. This is the sense in which the paradigm is a potential space – different choices from among the associative fields of the paradigm result in syntagms with different meaning. With the positive appearance of a specific sign comes a virtualized cloud of its associative field, that is, all the signs that could have been chosen instead. For example, in the language of classical architecture, the appearance of a Doric column in a syntagm (building), immediately brings to mind the Ionic and Corinthian columns that could have been chosen instead. The meaning of the positive appearance of a sign is structured by the associated signs that *did not* appear; the meaning of a Doric column is structured by the fact that it is *not* Ionic or Corinthian. Thus the form of the content plane is structured by relational opposition among the signs of associative fields.

Linguistic sign systems, such as one's native language, are given. The entire weight of one's culture and its history enforces the system. While there is room for innovation at the level of the production of complex syntagms (i.e. conversations or texts), innovation that radically re-articulates the planes of content and expression is not possible. Attempts at such radical innovation quickly walks off the edge of language and culture, appearing as nonsense or insanity. Sign-function systems, on the other hand, often originate from privileged groups who design and enforce the system. For example, the fashion system of haute couture originates from a small body of designers, critics and publishers who design and disseminate the syntagmatic and paradigmatic rules for this season's fashions. The originating body has much more freedom to radically innovate in the production of the sign system than do consumers, who are generally limited to speaking the language.

For the purposes of this chapter, the last two semiological notions we need are those that relate two semiotic systems, namely connotation and meta-language. These phenomena occur when a semiotic system becomes either the expression plane or the content plane for a second semiotic system.

Connotation occurs when one semiotic system becomes the expression plane of another, as appears in Figure 7-3.

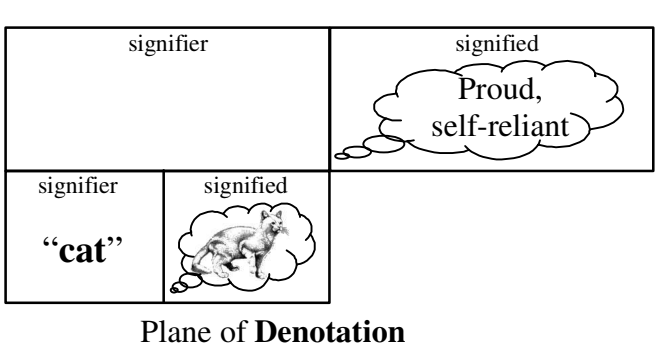

Figure 7-3. Connotation

In this situation, the first system is called the plane of denotation, and the second system the plane of connotation. Signifiers in the plane of connotation are entire signs, including complex signs (syntagms), from the plane of denotation. Connotation introduces hidden, oblique meanings, to the plane of denotation – the plane of connotation is sometimes referred to as myth or ideology. For example, imagine a television commercial for toothpaste. The scene cuts between a handsome man and a beautiful women brushing their teeth in the bathroom and smiling their bright smiles into the mirror, while a voice-over describes how the "triple-action" of the toothpaste simultaneously whitens teeth, freshens breath, and removes plaque. The commercial ends with the man and women meeting each other on the street, smiling broad, pearly-white smiles and laughing. On the plane of denotation, the commercial is saying that that this toothpaste effectively cleans your teeth. On the plane of connotation, the commercial is saying that this toothpaste will make you beautiful, successful, and alleviate loneliness. Semiologists are interested in unpacking connotative sign systems as a way of understanding the cultural myths or ideologies operating behind the scenes of a society.

Meta-language occurs when one semiotic system becomes the content plane of another, as in Figure 7-4. In this situation, the first system is called the object language, and the second system is called the meta-language. Signifieds in the meta-language are

entire signs, including complex signs, in the object language – the meta-language is talking about the object language. A meta-language is referred to as an *operation*, as it provides support for manipulating the object language. Semiology itself is a meta-language, a sign system used to talk about other sign systems. For example, the signifier "**sign for cat**" (the literal marks), signifies a sign that in turn has its own signifier and signified.

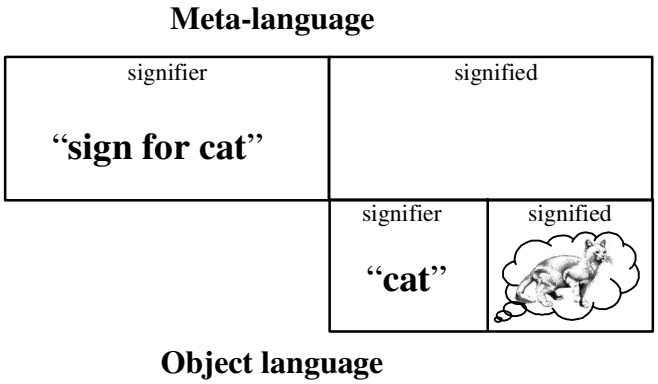

Figure 7-4. Meta-language

Structuralist semiotics provides a framework for understanding how both the technical architecture and the audience experience of an AI-based artwork function as interrelated sign systems.

The Code Machine and the Rhetorical Machine

AI (and its sister discipline Artificial Life), consists of both technical strategies for the design and implementation of computational systems, and a pared, inseparable, tightly entangled collection of rhetorical and narrative strategies for talking about and thus understanding these computational systems as intelligent, and/or alive.

These rhetorical strategies enable researchers to use language such as "goal", "plan", "decision", "knowledge", to simultaneously refer to specific computational entities (pieces of program text, data items, algorithms) and make use of the systems of meaning these words have when applied to human beings. This double use of language embeds technological systems in broader systems of meaning.

There is an uncomfortable relationship between a purely relational (and thus literally meaningless) technical manipulation of computational material, and the interpretation of this computational material by a human observer. Simon and Newell posited the symbolic symbol system hypothesis as a fundamental assumption of AI [Newell & Simon 1976]. This hypothesis states that a physical system consisting of a material base that can take on various configurations (call these configurations "symbols") and a material process that manipulates these physical constellations to yield new constellations, is sufficient for the production of intelligent behavior. This formulation immediately produces an interpretation problem in which an external observer is necessary in order to view the material constellations as signs in such a manner that intelligence can be observed in the material production of sign from sign. Interpretation, with all of its productive open-endedness, is thus crucial to the definition of intelligent system, but is usually pushed to the background of AI practice.

Figure 7-5. Total system = code machine + rhetorical machine

The necessity of rhetorical strategies of interpretation is not avoided by "subsymbolic" techniques such as neural networks or genetic algorithms utilizing numeric genomes (i.e. not the tree-shaped, symbolic genomes of genetic programming), nor by machine learning methods based on generalization from training data, nor by behaviorist robotic techniques that link sensors to effectors through stateless combinational circuitry or finite state machines. These approaches still require the interpretation of an observer in order to make sense of the input/output relationships exhibited by the system, to select the primitive categories (features) with which the inputs are structured, and to tell stories about the processes producing the input/output relationships. These stories are essential for thinking through which technical constructions to try next, that is, for simultaneously defining a notion of progress and a collection of incremental technical constructions that make progress according to this notion.

The rhetorical strategies used to narrate the operation of an AI system varies depending on the technical approach, precisely because these interpretative strategies are inextricably part of the approach. Every system is doubled, consisting of both a computational and rhetorical machine (Figure 7-5). Doubled machines can be understood as the interaction of (at least) two sign systems, the sign system of the code, and a sign system used to interpret and talk about the code.

The central problem of AI is often cast as the "knowledge representation" problem. This is precisely the problem of defining structures and processes that are *simultaneously* amenable to the uninterpreted manipulations of computational systems *and* to serving as signs for human subjects. This quest has driven AI to be the most promiscuous field of computer science, engaging in unexpected and ingenious couplings with numerous fields including psychology, anthropology, linguistics, physics, biology (both molecular and macro), ethnography, ethology, mathematics, logic, etc. This rich history of simultaneous computational and interpretive practice serves as a conceptual resource for the AI-based artist.

The Code System

The program code, considered as a sign system, relates two planes: a plane of expression containing the space of all possible pieces of program text (the marks on a screen or

page), and a plane of content containing the space of all potential executions. That is, a piece of program code is a signifier signifying (the mental concept of) the effect of executing this code. For example, the signified of the simple sign (code fragment) x = 1 is, for programmers used to working in imperative languages, probably something like placing a 1 within a box labeled x[23].

The relationship between the mental concept of an execution and the physical effect of executing a piece of code on a concrete computer (e.g. for contemporary digital computers, changing voltage levels in pieces of silicon) falls outside of the purview of structuralist semiotics. A code fragment is a sign-function, having both a utilitarian, technical use (the physical effect of executing the code on a concrete machine), while serving as a sign for its potential execution. Obviously there are constraints imposed on sign value by use value; for example, the physicality of a rubber ball, and the technical functions (e.g. bouncing) that the physicality of a rubber ball supports, prevents (or at least makes quite difficult) the rubber ball from taking on the sign value of a tasty snack. Similarly, the possible sign values of a code fragment are constrained by the use value, the physical effect of its execution on concrete machinery. Though a structuralist semiotic analysis has its limits, such as difficulty in offering a detailed analysis of the relationships between sign and use value, it remains the case that much of human activity is structured by language-like interactions, from which a semiotic analysis gains its traction. In the specific case of the activity of programming, programmers think about potential executions and read and write texts to express those potential executions; this language-like activity suggests that the semiotic view of program code as a sign system, while not explaining *everything* about the human activity of programming, is likely to yield dividends.

To further unpack the idea of code as a semiotic system, consider the example of rhetorical goals in *Terminal Time* (see Appendix C). The textual representation, the code, for a specific rhetorical goal appears in Figure 7-6.

```
(def-rhetgoal :name :give-positive-example-of-big-science
   :app-test (%and
               ($isa ?event %SciTechInnovationEvent)
               ($performedBy ?event ?bigscience)
               ($isa ?bigscience $LegalGovernmentOrganization)
               ($isa ?bigscience $ResearchOrganization))
   :rhet-plans (:describe-event)
   :emotional-tone :happy)
```

Figure 7-6. The code representation of a rhetorical goal

This complex sign is itself a syntagm, composed of a constellation of signs. But considering the complex sign as a unity, the rhetorical goal signifies potential executions in which the system will tend to include a certain class of historical events in the constructed documentary, in this case, events in which governmental research organizations engage in scientific or technical research, in such a way as to make a certain point, in this case, that it is beneficial when science and government come together. It is interesting, perhaps surprising, that this relatively small textual signifier signifies potential executions that relate so directly to *Terminal Time's* output; watching a

[23] For programmers used to working in functional or declarative languages, for whom this code fragment is a binding, not an assignment, the signified is likely different, something like tying x and 1 together with a piece of string.

generated documentary (in which this goal is active) with this code sign in hand, it is possible to relate the appearance of specific historical events in the documentary (such as a breathless, glowing description of the moon landing or the invention of the atomic bomb) to this code sign, that is, to the effect on execution of this textual signifier. It is certainly not a given that a system of code signs would necessarily provide form to the plane of textual representations (expression) and the plane of potential executions (content) in this way. It takes work to articulate the planes in this particular way – this work is in fact the *creation of a custom code system*.

Standard languages, such as C++, lisp, or Java, define code systems, specific ways of chopping up the spaces of textual representations and potential executions. Like many sign-function systems, the more radical innovation of the creation of the sign system lies with special individuals or organizations who define the language, with consumers of the language limited to working with the signs, the associations between text and execution, established by the language. But it is standard practice in computer science, enabled by Turing equivalence, to use a pre-given code system (language) to implement new code systems that provide *different associations between text and execution*. This practice allows individuals to engage in the more radical innovation of creating new code systems particularly suited for a specific task. Mainstream languages, such as the three mentioned above, tend to be strongly procedural; the control structure, which determines the temporal relationship between bits of execution, is explicitly captured in the textual representation. However, this is not the only kind of code system. One can define purely declarative code systems, such as the rhetorical goal above. In declarative systems, the textual representation does not explicitly capture temporal relations in execution. Rather, the code signs indicate execution propensities. The system as a whole will tend to behave in certain ways if the declarative sign is part of the system, though the precise execution path (temporal sequence of sign execution) is unknown. Or the custom language may be a hybrid, such as ABL, which combines the declarative features of production systems with the procedural features of more mainstream languages.

The *architecture* is the conglomeration of code that implements a custom language, that is, establishes the relationship between bits of textual representation and potential executions. *Terminal Time's* architecture appears on page 248. A rhetorical goal becomes a sign by virtue of its role within the entire architecture. The rhetorical goal has relationships with or participates in many parts of the architecture, including the knowledge base, the story board, natural language generation, the selection of music, and (indirectly, through the goal's effect on the natural language generator) the sequencing of video clips. This little bit of text gains its meaning through its effect on a broad array of processes throughout the architecture.

At this point it is possible to provide a semiotic account of the code system properties that yield interpretive and authorial affordances.

Affordance in the Code System

An AI-based artwork is a semiotic system productive of a (potentially large) number of syntagms. AI-based artworks are thus *generative*; computational processes provide the combinatoric machinery necessary to select terms out of the fields of potential terms (associative fields) provided by the system. The system produces variable syntagms in different situations. For example, *Office Plant #1's* behavior over time depends on the email received by its owner, the content of documentaries generated by *Terminal Time* depends on audience answers to the psycho-graphic polling questions, and Trip and Grace's moment-by-moment behavior in *Façade*, as well as the more global story

structure, depend on the player's real-time interaction and patterns of interaction over time.

The internal structure of the machine, the program code, wires, circuits and motors out of which a work might be constructed, is itself a syntagm of the semiotic system defined by the architecture (Figure 7-7). The architecture consists of the custom code systems, processes, modules, and relationships between modules, which together define the implementation language, the sign system within which the work will be constructed. Building an AI-based artwork thus means constructing a semiotic system of implementation (an architecture, $system_1$) such that it supports the construction of a syntagm (the specific work built within the architecture, $syntagm_1$), which, when executed, becomes a semiotic system ($system_2$) autonomously productive of its own syntagms ($syntagm_2$) in different situations. $System_1$ (the architecture) has appropriate *authorial affordances* when there is a "natural" relationship between changes to the $syntagm_1$ and changes in the syntagmatic productivity of $system_2$. By "natural" is meant that it is easy to explore the space of syntagmatic productivity consistent with the artistic intention of the piece.

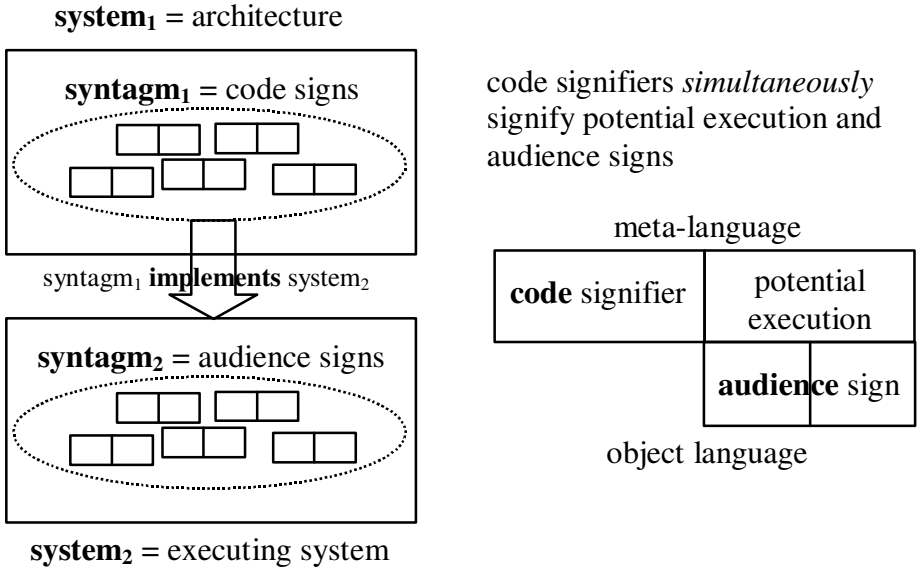

Figure 7-7. Relationships in the code system

For example, in *Terminal Time*, the architecture on page 248 is $system_1$. $Syntagm_1$ is the collection of historical events (collections of higher-order predicate calculus statements), rhetorical goals, rhetorical devices, natural language generation rules, rhetorical plans, and annotated video and audio clips, which collectively make up the specific artwork that is *Terminal Time*[24]. Individual signs within $syntagm_1$, as well as $syntagm_1$ as a whole, *are* signs (have meaning) by virtue of their participation within $system_1$. The execution of $syntagm_1$ results in $system_2$, in a runtime instance of *Terminal Time*. And, as the audience interacts with $system_2$, it produces $syntagm_2$, a particular documentary out of the space of all possible documentaries expressible within (producible by) $system_2$. While the structure of $syntagm_2$ is quite literally determined by

[24] Since signs may be added or changed over time, such as the modification or addition of rhetorical devices or historical events, *Terminal Time* as a specific piece changes over time.

system$_2$, for the audience, the meanings expressed by syntagm$_2$ are determined by a meshwork of different sign systems, including the system of documentary imagery, the system of cinematic music, the linguistic system for English (the voiceover), and a folk psychology of the execution of system$_2$ (e.g. "we voted that religion is a problem in the world, and now it's trying to make the point that religion is bad"). Thus syntagm$_2$ is multi-articulated; its meaning is determined not just by system$_2$, but also by a number of sign systems *outside* the technical system$_2$.

System$_1$ is a meta-language for talking about system$_2$; utterances in system$_1$ (syntagm$_1$ or fragments) talk about potential utterances of system$_2$ (syntagm$_2$ or fragments) (see Figure 7-7). For *Terminal Time*, system$_1$ utterances, such as the rhetorical goal in Figure 7-6, are a way of talking about potential system$_2$ utterances, such as a breathless, glowing description of the invention of the atomic bomb. System$_1$ offers *effective authorial affordances when one and the same syntagm$_1$ simultaneously talks about desired syntagms$_2$ (or fragments), and, when executed, implements the appropriate system$_2$ that indeed produces the desired syntagms$_2$*. This property is not trivial – there are a number of ways in which it can fail to hold.

It can be the case that system$_1$ fails to provide appropriate signs for talking about desired properties of syntagm$_2$. For example, an early version of *Terminal Time's* architecture represented historical events directly at the natural language generation and video clip sequencing level. There was a fairly direct connection between answers to the audience polls and the generation of specific text about specific events. Given this system$_1$, it was impossible to express *general* relationships between poll answers and categories of events. For example, if the winning answer to the question "What is the biggest problem in the world today" is "It's getting harder to earn a living and support a family", the desired syntagm$_2$ should include events demonstrating the evils of capitalism. Given a relatively direct connection between poll answers and natural language generation, there just was no way of expressing this more general desired property of syntagm$_2$, and thus certainly no way of implementing the appropriate system$_2$ with syntagm$_1$.

It can be the case that syntagm$_1$ utterances *purport* to talk about desired syntagms$_2$, but in fact, when executed, don't implement a system$_2$ that produces the desired syntagm$_2$. For example, in *Subjective Avatars*, the Oz emotion system Em [Neal Reilly 1996] is used to maintain autonomous emotional state within an avatar. When the author annotates an avatar goal so as to produce an emotional state, the purported effect on systagm$_2$ involves manipulating the presentation in such a way as to make the player experience this emotional state (and thus condition the responses of the player). But to make the goal annotation actually result in a system$_2$ that generates the desired syntagm$_2$, there must actually be mechanisms that perform the appropriate presentation manipulation as a function of emotional state. For the text world of *Subjective Avatars*, this means providing appropriate natural language generation rules that make use of emotional state when generating text from the formal sensory description language of the Oz text system. If these rules are poorly written or missing, then the emotion annotation fails to provide appropriate authorial affordances.

As a final example of the failure of authorial affordance, it can be that case that syntagm$_1$ is successful in simultaneously describing a desired syntagm$_2$ and implementing an appropriate system$_2$, but that, when the audience (who may in fact be the same as the author) actually experiences the produced syntagm$_2$, its interpretation is different than expected. This situation arises precisely because syntagm$_2$ doesn't participate in just the technical system$_2$, but in a meshwork of sign systems *outside* of the

technical system. That is, part (perhaps a large part) of the meaning of $syntagm_2$ is opaque to the technical system, but rather comes along for the ride as the technical system manipulates and produces signs. For example, in *Façade*, a beat, and the associated beat behaviors, may purport to serve the dramatic function of communicating that when Trip asked Grace to marry him she wasn't really ready, while simultaneously communicating that they are both getting more upset and that Grace currently feels disaffiliated with the player. The associated beat code may simultaneously describe the author's vision of the desired run-time experience, and, when executed, implement the author's vision of the desired runtime experience. But when the author, or another player, plays the experience, Trip and Grace actually seem less upset than in the preceding beat, even though they are supposed to be more upset. What happened here is that the details of the writing, and how the details of their physical performance actually read, are *extra-architectural*; they lie outside the literal code of the system. Even though the beat is "performing to spec", other sign systems are subverting its interpretation. Every AI system is doubled. A description of the code system is not enough – we need to examine the rhetorical system.

The Rhetorical System

The signs of both $system_1$ and $system_2$ are multi-articulated; their meaning arises both from syntagmatic and paradigmatic constraints established by the respective code systems, but also from a collection of sign systems *outside* of the code systems. This collection of external code systems is the rhetorical system. Both authors and audiences make use of the rhetorical system in narrating the operation of the system and forming intentions with respect to the system. The code and rhetorical systems are tightly entangled; both play a role in understanding interpretive and authorial affordances.

(Audience) Interpretive Surplus

$Syntagm_1$ never completely describes all the properties of $syntagm_2$; though $system_2$ literally prescribes the possible elements (paradigm) and spatial and temporal relationships between elements (syntagm) of $syntagm_2$, a portion (perhaps a large portion) of the signification is determined by external sign systems. This interpretive surplus occurs because $system_2$ operationalizes a meta-language ($syntagm_1$) for describing the audience experience ($syntagm_2$). The signifieds of this meta-language are themselves signs, participating in external sign systems, which are handled by the meta-language.

The crafting of these external, handled signs, becomes an irreducible problem in design and aesthetics. These handled signs must be crafted to marshal the signifying resources of these external sign systems in such a way as to match the purported meanings of the code system. For example, in *Façade*, we as authors have to write dialog that consistently communicates the character of Grace and Trip, while communicating meanings appropriate for a specific beat goal within a specific beat, while also being re-sequenceable to various degrees (both within the beat in response to handlers and at the beat level, in response to drama management). Specific lines of dialog must meet multiple constraints established by how the code machine will make use of the line. Additional meaning is carried by how a voice actor performs the line. The nuances of emotional tone, irony, sarcasm, desperation, etc., communicated by the voice performance, must also be consistent with these constraints. In authoring *Façade*, there is a reciprocal process between authoring these handled signs (e.g. dialog, snippets of

animation data) and code-level authoring within the architecture. Consistency between handled signs and manipulation by the code machine is established by moving back and forth in the authoring of these two domains. But consistency is not the same as identity; there are always aspects of audience interpretation that escape the code machine.

Another avenue for interpretive surplus is connotation; the handled signs may become the plane of denotation for a connotative system. For example, in *Terminal Time*, the ideological arguments made by the system are often (purposely) undermined through irony. The details of imagery, music, and the narrative track connote irony, while at the level of denotation an earnest argument is being made. For example, if the anti-religious rationalist ideologue has been activated, a 20th century event it may make use of is the Chinese invasion of Tibet. Within the knowledge base, the two actors of this event are Tibetan Buddhists (which the system infers are a kind of Religious Group), and Maoists (which the system infers are a kind of Rationalist through their connection to Marxism). Furthermore, the event is a War, instigated by the Maoists (Rationalists) against the Buddhists (Religious Group), in which the Maoists are successful. This is enough for the Anti-Religious Rationalist to decide it can use this event as a Positive Example of Rationalist Progress. Assuming that this event spin (the ideologically-slanted representation of the "objective" representation in the knowledge base) makes it into the final generated documentary, the system will earnestly argue that this is a positive example of Rationalists mopping up the remaining dregs of irrational religion (e.g. "There were reports that Buddhists monks and nuns were tortured, maimed and executed. Unfortunately such actions can be necessary when battling the forces of religious intolerance.") over a montage of Tibetan Buddhist imagery and Chinese soldiers holding monks at gunpoint, while playing the happy, "optimistic" music loop. The system does not "know" that it is undermining its argument through irony; irony is not a property described within the code machine. We as system authors marshaled the handled signs (language, video clips, music) to connote irony on top of the structure explicitly provided by the code machine.

Given that the audience interpretation of $syntagm_2$ always escapes full specification by the code machine, it may be tempting to conclude that computer-based art practice should primarily make use of the signifying resources of external sign systems via handled signs. Crafting the handled signs, animation snippets, imagery, video clips, music loops, and so forth, falls comfortably in the realm of more traditional art practice. Such an approach would move back towards the "code as a hack" model, throwing together the minimum code machine necessary to coarsely manipulate handled signs. But this approach would severely limit the maximum attainable agency. As the interpretive surplus becomes larger and larger, with more of the interpretive affordance pushed onto the handled signs, the imbalance between material and formal affordances grows. The rich handled signs suggest many avenues of action to the audience. But with no corresponding richness in the code machine, there is no way for the work to respond to these actions; the rich, coarsely handled signs suggest a richness of response that the work can't satisfy (see ***Agency*** on page 27). But the reason for designing a rich and expressive architecture goes beyond the "utilitarian" goal of supporting audience agency. The architecture ($system_1$), and systems designed within it ($syntagm_1$), are themselves embedded in a meshwork of external sign systems, providing the AI-based artist with a rich architectural surplus.

Architectural Surplus

Agre [Agre 1997a] describes how AI technical practice provides narrative affordances that support AI researchers in creating stories describing the system's operation.

> … the practical reality with which AI people struggle in their work is not just "the world", considered as something objective and external to the research. It is much more complicated than this, a hybrid of physical reality and discursive construction. … Technical tradition consists largely of intuitions, slogans, and lore about these hybrids, which AI people call "techniques", "methods", and "approaches"; and technical progress consists largely in the growth and transformation of this body of esoteric tradition. [Agre 1997a: 15]

Different practices (e.g. classical AI, interactionist AI) provide different affordances for narrating system behavior. For the classical AI researcher, the discursive construction consists of ways of talking about "goals", "plans", and "knowledge", while for the interactionist AI researcher, the discursive construction consists of ways of talking about "embodiment", "action", and "improvisation". These discursive constructions are a necessary part of the functioning of the system.

> To understand what is implied in a claim that a given computer model "works", one must distinguish between two senses of "working". The first, narrow sense, again is "conforms to spec" – that is, it works if its behavior conforms to a pregiven formal-mathematical specification. … the second, broad sense of "working" … depends on specific words of natural language. As I mentioned at the very beginning, an AI system is only truly regarded as "working" when its operation can be narrated in intentional vocabulary, using words whose meanings go beyond mathematical structures. When an AI system "works" in this broader sense, it is clearly a discursive construction, not just a mathematical fact, and the discursive construction succeeds only if the community assents. [Agre 1997a:14]

In typical AI research practice, these affordances are often not consciously acknowledged or manipulated. Rather, they serve as part of the unconscious background, co-evolving with the technical practice as a silent but necessary partner in the research. Systems are spoken of as having "goals" or engaging in "embodied action", as if these were primitive, readily detectable properties, like being blue, or being cold, rather than the hard-won results of rhetorical construction and debate. But in Expressive AI practice, the narrative affordances are an explicitly manipulated resource, an architectural surplus that makes the architecture not just a bunch of code, but a way of thinking about the world.

Within the semiotic framework of this chapter, the architectural surplus (an interpretive surplus on the author side), can be understood as one or more meta-languages, in which the signs in $system_1$ ($syntagm_1$) form the content plane, and as one or more connotative systems, in which signs in the meta-language form the plane of denotation.

For example, consider joint goals in ABL. The code sign for a joint goal appears in Figure 7-8. The sign signifies that a team of ABL agents will attempt to achieve `Goal1()`. A meta-language allows us to talk about and thus operate on these code signs. This meta-language consists of ordinary language that has been co-opted into talking about code signs. This meta-language in turn serves as the plane of denotation for a connotative sign system – this connotative sign system contains the "spillover" of the co-opted ordinary language, connotative meanings that escape the strict meaning of the code signs. In this

case, the meta-language sign for a joint goal connotes the idea of a team of people working together, with all the non-formalized richness of this notion. The connotation lifts the code sign out of the circumscribed meaning provided by the architecture, and into the more open-ended sign system used to talk about coordinated human activity in the everyday world. Once lifted into this connotative system, the author can use the connotative sign system to think about the human realm of teamwork. But new signs reached by thinking in the connotative plane can in turn have signifiers in the meta-language whose signifieds lie back in the code system. Thus ordinary language, in this case the ordinary language of human teamwork, becomes a meta-language for talking about and manipulating a technical system, in this case the code system for joint goals in ABL. This movement, from code system, into ordinary language, and back into code system, creates a circulation of signs that suggests both new ways of using the architecture and new architectural elaborations, in this case news ways of using joint goals and new architectural elaborations for joint goals.

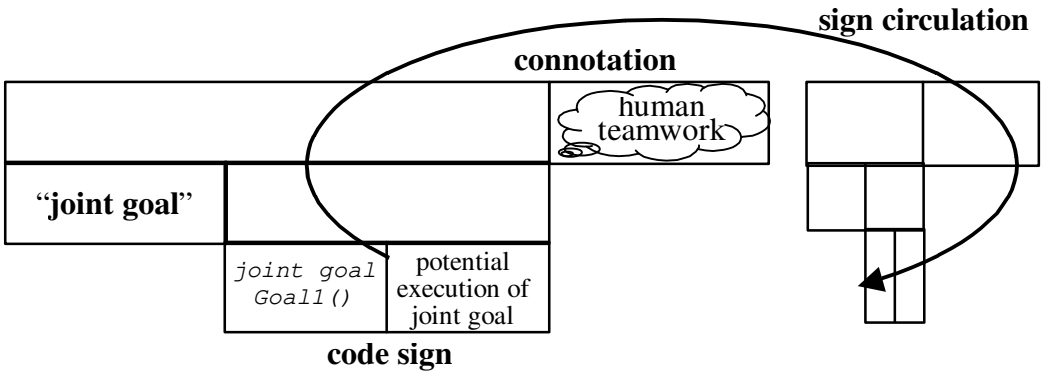

Figure 7-8. Code signs, meta-language, and connotation

Consider first how the ordinary language system of human teamwork suggests new ways of using joint goals. In the everyday human world, we think of people coordinating to achieve goals they want to achieve; that is, we imagine people having a positive emotional valence towards a goal. Two people might team up to hang a picture, or change a tire, but we don't picture people teaming up to have a big fight, or teaming up to accomplish a mugging, with one team member the victim and one team member the mugger. An author may thus never think of using joint goals to coordinate a big fight among two agents. But now imagine that in the connotative plane we start thinking about teams of movie actors or stage actors. In acting within a play or movie, human actors often tightly coordinate in the carrying out of dramatic activity in which the characters strongly oppose each other, as in, for example, a play in which a marriage falls apart as years of buried frustrations and misunderstandings are revealed. Now this gives us the leverage (meta-language) to imagine using joint goals in *Façade* to tightly coordinate conflicts between characters. Ordinary language, used as both the plane of connotation and as meta-language, is a necessary part of the *total* system – it provides the code system with broader meaning and consequently suggests new ways of manipulating the code system. Note that this example involves consciously manipulating and exploring the plane of connotation in order to reveal a new possibility within the code system. If we were uncritically wedded to the ordinary language system of "rationality", in which

people only pursue goals for things they emotionally desire, then the code system idea of jointly accomplishing conflict may never arise.

The plane of connotation and meta-language not only suggests ways of using the code system ($syntagm_1$), but modifications and elaborations of the architecture itself ($system_1$). Continuing with the joint goal example, consider the control of activity within a joint goal. In ordinary language, when we imagine team members accomplishing a task together, we often imagine the decision of what step to do next being distributed among the team members. Certainly there are hierarchical situations in which a team leader is responsible for managing the teams, but many teamwork situations are more collaborative and decentralized. Now consider the initiation of joint goals in the code system. When one member of a team initiates the joint goal, the other members of the team, on successful entry into the joint goal, spawn the goal at the root of their active behavior tree (ABT). Only the joint goal initiator has the goal deeper within the ABT. If other members of the team initiate joint subgoals in the service of the original joint goal, these joint subgoals will appear at the original initiator's ABT root. This is a bit counter-intuitive, given that within the ABT subgoals are normally children of the goal (via a behavior) they are in service to. But strictly at the code level there is nothing wrong with this arrangement. However, consider how the ABT is connotatively read or interpreted. The ABT captures the structure of an agent's thoughts, its mind. It is not just a bookkeeping mechanism controlling execution, but a *representation* of the agent's activity. Reflective processes, such as the emotion system Em or the handlers described in Chapter 6, treat the ABT directly as a representation. But even without reflection, the mechanisms for success and failure propagation, the many annotations that modify success and failure propagation, and continuously monitored conditions, all work together to support the reading of the ABT as a representation. When a goal appears deep in the ABT, it is enmeshed in more complex patterns of activity than a goal shallower in the ABT – ABT depth becomes a proxy measure for the complexity of the agent. With this reading of the ABT, combined with the ordinary language model of teamwork, the default joint goal initiation mechanism is seen as lacking. Initiated joint goals, since they are always at the root of the ABT, aren't able to fully participate in complex patterns of activity. This is particularly problematic for "flat" teams, in which all team members equally participate in the control logic for the team, and thus both initiate and respond to requests to enter joint goals. This circulation between readings of the ABT, code signs for joint goals, and readings of these code signs, suggests an architectural modification, specifically the synchronize keyword described on page 102, which supports the initiation of joint goals anywhere in the ABT.

Authorial affordance consists not just of the code system relationship that $syntagm_1$ simultaneously implements $system_2$ and describes $syntagm_2$, but also of the rhetorical relationship that $syntagm_1$ is readable and handleable by interpretive systems and meta-languages. An architecture is a machine to think with. The complex circulation between code signs and the interpretive framework provides authors with both resistance (some things will appear hard or impossible) and opportunity (new ideas arise). Thinking with the architecture suggests new audience experiences, creating a feedback loop between authorial intention and the details of the *total* system (code + rhetoric). But establishing this interpretive framework, the plane of connotation and meta-language, takes real work. It is the outcome of a practice that simultaneously tries to articulate the code machine *and* the ways of reading it and talking about it. In contrast, a practice that views the system as a hack, as a means to an end, will likely construct systems with poor authorial

affordances, lacking both the code system relationships and rich rhetorical frameworks necessary to enable new audience experiences.

Idioms

Idioms are ways of using an architecture, conventional structures for the authoring of $syntagm_1$. For example, ABL idioms are described in Chapter 6 and beat manager idioms are described in Chapter 8 starting on page 163. Idioms arise through the interplay of the architecture and its interpretive frameworks. In a sense, the idioms actually cash out the interpretive framework, being the place where interpretation and code meet. This is why idioms are so important for truly understanding an architectural system. An abstract description of a code system will make use of all kinds of ordinary language words, such as "plan", or "embodied activity", or "learning", but understanding the particular entanglement of rhetoric and code that is the total system requires examining the detailed circulation between these language signs and code signs. Idioms are the place where this detailed circulation occurs.

As idioms become larger and more diffuse, they begin restricting the circulation between code and rhetoric. The code signs become large and diffuse, making the connotative lifting and meta-language handling difficult. For example, in *Façade*, if ABL lacked the reflective structure to implement beat behaviors as beat goals plus handlers, the behavioral representation of the dramatic activity the characters wish to carry out in the beat would be entwined with the logic that handles player interaction. The beat behavior idiom would perforce combine the two concepts (dramatic action, interaction) into some kind of blob. This blob would make it difficult to even think of a beat as being composed of a logic of dramatic action and a logic of interaction, because the more diffuse nature of these code signs would make it difficult to connotatively lift and handle the code signs using language signs like "dramatic action" and "interaction".

Idioms can thus reveal breakdowns in the total system, conceptual domains in which the circulation between rhetoric and code are restricted. The breakdowns suggest architectural opportunities, modifications of the architecture that enable new idioms and simultaneously re-articulate the interpretive sign systems, providing new ways of talking and thinking about the code system. Systems built without an explicit concern for authorial affordances are likely to be *all* idiom, and thus severely restrict the circulation between rhetoric and code. This would be the case, for example, if *Façade* was written as a giant program in a standard programming language such as C. The only code signs at our disposal would be the rather low-level signs provided by C. Everything else would be idiom, with large chunks of C code having only a diffuse relationship to signs of the audience experience ($syntagm_2$) and to connotative and meta-languages. This extreme case of the code system being nothing but idiom, code piled on code, provides poor authorial affordances, making it difficult to think about, discover, and express, new conceptual frameworks and new audience experiences.

Generality of the Double Machine

The use of a structural semiotic terminology in this chapter, with the focus on "sign systems", "languages", "connotation" and so forth, may lead a reader to conclude that the analysis of affordances in terms of doubled machines of rhetoric and code is only useful for classical AI systems, with their explicit focus on symbolic knowledge. The analysis applies much more broadly than this, however, to any AI or ALife practice. All such practices make use of a rich entanglement between technical systems and ways of talking

and thinking about the technical system. Consider a robot built along the lines of subsumption architecture [Brooks 1986], in which finite state machines mediate rather directly between sensory input and motor actuation. The finite state machines may in fact be implemented entirely in hardware, rather than as code in a general purpose micro-controller. Yet there is a still a "code machine" that participates in complex discursive constructions. Wires bearing voltages are in no less need of interpretation than fragments of textual code, and participate in the same sign system relationships that support interpretive and authorial affordances.

The focus in this chapter on authorship may similarly lead a reader to conclude that this analysis is not relevant to machine learning. But again, the methods of machine learning consist of a technical/rhetorical system, one organized around the "learning" or "discovering" of "patterns" in "raw data". But, of course, human authors select the primitive features, define the representations of hypotheses or distributions, define the search methods employed to tune parameters, and design how particular machine learning methods are embedded in larger architectures. For example, Thom [Thom 2001] makes use of unsupervised learning techniques to build a believable musical companion, a system that improvises with you in your own style. Thom's system, *Band out of a Box*, learns distributions of musical features she authorially selected, as well as temporal patterns over these features, in an invertible representation that allows her to then generate music with the same feature distributions. Unsupervised learning methods provide a technical/rhetorical system with authorial affordances supporting the creation of an experience with desired interpretive affordances ("Hey, it's trading licks with me in my own style."). *Office Plant #1* similarly makes use of the technical/rhetorical system of text learning as part of an architecture supporting the creation of a non-human companion responding to email activity.

Conclusion

This chapter develops authorial and interpretive affordances as central terms in the hybrid practice of Expressive AI. The relationship between these two affordances shows how Expressive AI is simultaneously concerned with art's creation of meaningful experience (and the consequent focus on interpretation of the art object), and AI's construction of machines that can be understood as behaving intelligently (and the consequent focus on the structures, properties and processes of these machines). Structuralist semiotics, through its concern with signs systems and the relationships between systems, provides a common ground in which both the artwork as experienced by the audience and the construction of machines as experienced by the author can be seen as instances of sign systems – this provides the framework for a more detailed analysis of the relationship between these affordances.

As an analytical framework, structuralist semiotics has its limits. Arising from the tradition of Sassure, its view of the world as a meshwork of language systems whose rules can be analyzed has trouble accounting for the actual processes involved in the use and production of signs. Some work in the analysis of computational media has fruitfully made use of Peirceian semiotics, whose sign concept includes a notion of meaning more amenable to process (e.g. [Anderson, Holmqvist & Jensen 1993; Frasca 2001:chapter 4]). Further analysis of the negotiation of meaning in technical systems could fruitfully make use of ethnographic and phenomenological frameworks. However, the structuralist analysis here, with its focus on the relationships between sign systems, goes a long way

towards understanding both how and why Expressive AI is simultaneously concerned with the code system and audience interpretation.

CHAPTER 8
THE *FAÇADE* DRAMA MANAGER

Drama Management

A drama manager responds to player interaction within an interactive story in such a way as to create a pleasing story structure. This chapter describes the *Façade* drama manager, which sequences beats in response to player interaction. But before diving into the details of the beat sequencer, let's first consider the general goals and requirements of a drama manager.

Global vs. Local Agency

As described in Chapter 2, the goal of neo-Aristotelian interactive drama is to create a sense of agency within a dramatic story. Player agency can be further classified into local agency and global agency. Local agency means that the player is able to see clear, immediate reactions to her interaction. Global agency means that the sequence of events experienced by the player (in our case, the sequence of beats) is strongly determined by player interaction. Looking at a completed story, we should be able to look at the sequence of beats and create an explanation for why that sequence happened. Looking at a story as it's happening, global agency means that what the player does now has a strong influence on what will happen in the future, say, three beats from now.

Local agency is present in the detailed interactions with characters; the characters immediately react to player actions and utterances. In *Façade*, the smarts to handle local interaction, and thus create a sense of local agency, resides within beats plus global mix-ins. The logic of beat goals plus handlers is the framework for local interaction. The smarts to incorporate interaction into larger scale story structures, to incorporate not just the previous interaction, but in some sense the entire history of interaction into the future direction of the story, and thus to provide global agency, resides in the drama manager. In *Façade* this is the beat sequencer, which selects the next beat in the story based on the previous interaction history.

Simulation is sometimes suggested as a fruitful approach to interactive story [Smith 2000]. A simulation does seem to provide both local agency, as characters and objects respond immediately to player interaction, and global agency, as future action follows causal chains. For example, [Cavazza, Charles & Mead 2002] describes a system based on the world of the TV show *Friends* in which the characters pursue various goals in the face of interference created by player interaction. But what the simulation approach lacks is *tight story structure*. It is informative to examine the use of simulation in the history of AI-based non-interactive story generators. The first story generation system, *Tale-Spin* [Meehan 1976], generated Aesop-fable-like stories using simulation to generate action from an initial condition. For example, Crow might be given the goal to find the Worm, the Worm is placed somewhere in the world, various other characters (e.g. Fox) know about the location of the Worm, have their own goals, etc. The action of the story unfolds as the characters pursue their individual goals. Within the story generation community, the big lesson learned from *Tale-Spin* is that *simulation on its own rarely produces good stories*. Most "stories" generated by *Tale-Spin* consist of long-sequences of causally-

related activity that meander along until characters finally accomplish their goals. Only by luck does any interesting story structure appear in these sequences of actions. This is not to say that simulation is completely useless in story generation; more recent story generators, such as *Universe* [Lebowitz 1984; Lebowitz 1985] or *Minstrel* [Turner 1991; Turner 1994], have a simulation-like component (e.g. characters attempting to accomplish goals in a changing world), but add additional structure in the form of story knowledge (and processes) that actively shapes a sequence of activity into a sequence of story action. The drama management challenge is to provide this story-level structuring of a sequence of activity, while remaining responsive to player interaction and creating a sense of global agency.

The Combinatorics of Global Agency

Another window onto the drama management problem is provided by a combinatoric analysis of the authorial requirements necessary to create a sense of global agency. The first analysis considers fixed branching structures of the type found in many hypertext narratives. Imagine a branching tree-shaped narrative composed of story pieces (nodes) offering two choices per node. Each path to a leaf of this tree is a unique total story history. These story paths offer strong global agency; in fact, branching trees potentially offer[25] the maximum global agency in that the specific node you are on in any given path (story history) depends completely on *every previous decision* made along the path. But this maximum agency comes at a cost. A binary branching structure with 256 possible paths (leaves) requires the authoring of 511 story pieces.

Now consider a fully permutable collection of story pieces (nodes). Any node can precede or follow any other node. Further imagine that within any node there are two choices (like the binary branching hypertext) symbolically labeled *blue* and *green*. Half of our fully permutable nodes are blue nodes, half are green nodes – if blue is chosen within a node, the next node sequenced is blue (randomly chosen) – similarly for green. While there is potentially local agency in that the next node chosen depends on the player's decision, there is no possibility of global agency. Choices made now have no effect on the future beyond the next immediate node. And given a current node, the explanation for how you got there depends only on the previous local decision (there is a very weak history dependence in that you can't previously have "used up" the node). Someone might complain that we're cheating by, given a player choice, randomly selecting from among the appropriate set (blue or green). This seems to only offer the potential for weak local agency; perhaps if we improved the local agency, we would improve the global agency. So, rather than a binary choice, imagine that at every node there are as many choices as the remaining unused nodes. A player can directly select any desired node to follow the current node. This offers the potential for maximum local agency (at the node level), but does nothing to improve global agency. There are still no history-dependent effects nor large scale structures in the resulting node sequences. At the level of global structure, in an experience lacking global agency there is no difference between maximum local choice and random choice. But this scheme offers great authorial combinatorics. Where the binary tree structure required the author to create 511

[25] Since the notion of agency developed in Chapter 2 depends on the balance between material and formal affordances, that is, a balance between the communication of potential action and the means for action, whether any particular interactive narrative creates a sense of agency depends on more than the purely structural notions considered here. Thus a branching tree structure *potentially provides* maximum global agency; the material and formal affordances must still be balanced.

nodes in order to have 256 possible story histories, the flat permutable set only requires 6 nodes to create 720 unique possible histories (6 factorial).

There is a fundamental tension between the number of story pieces (and thus authorial work) and creating a sense of global agency. But increasing global agency doesn't mean that one has to jump immediately to the maximum global structure tree. In fact, providing more global agency without hand-authoring all possibilities is what a drama manager is all about.

Story Policies

Drama managers attempt to incorporate player interaction into the global structure *without* requiring the hand-authoring of all possibilities. Drama managers pull off this magic trick by softening the global structure requirements. The drama manager has some concept of what makes a good story (whether this is encoded as an explicit scoring function as in [Weyhrauch 1997], or implicitly as purely local heuristics, or in a mixed global-local form, like story-value arcs) and, based on player interaction, tries to select the next story piece so as to make a good story happen. "Good" stories tend to have lots of global structure (non-local connections between the actions of the player and story action). Of course, the drama manager doesn't have the perfect piece available for every situation; if it did, we would be back in the world of hand-authoring all possibilities. This is dealt with by making the notion of good story soft; when the perfect piece isn't available, the drama manager does the best it can based on some scoring function. In this way you can make a system that offers local agency, while creating something that, more-or-less, has interesting global structure and thus global agency, without having to hand-author something for every situation.

So a drama manager implements a story policy. Given a story situation, which includes the previous story history, the drama manager selects the next story piece to make happen (or try to make happen). The policy can be imagined as a table with an entry containing a story piece for every possible story situation – given a story situation, one looks up the situation in the table and determines which piece to sequence next. If there is a unique story piece for every possible story situation, we are back in the maximum-authoring/maximum-global-structure situation. The softness of the story policy comes from generalization – multiple story situations share the same entry. One can improve replayability of the story by, rather than having a single story piece per entry, instead having multiple story pieces with some probability distribution over them per entry. A further generalization of the story policy is to introduce hierarchy, where the entries of a table contain not story pieces, but story moves, where story moves are themselves sub-policies for selecting pieces in order to carry out the move. The difference between a story piece and story move is fuzzy. In *Façade*, the story pieces are beats, which certainly feel "piece-like" – but beats are of course themselves little machines composed of beat goals and handlers that perform the beat while incorporating and responding to within-beat interaction. However, there is no need to get hung-up on the difference between a story piece and story move, but only to recognize that story policies can themselves form a (heterogenous) hierarchy.

Given that a story situation includes not just the immediate story situation but the past history of interaction, the drama management policy table will be huge. Directly specifying this policy *as a table* is, from a human authorship perspective, intractable. Thus knowledge representation is part of the drama management problem – what kind of knowledge must the system have in order to dynamically compute this table? One way of representing story "goodness" knowledge is as an evaluation function, which, given a

total story history, returns a number indicating the "goodness" of the story. Such an evaluation function can be coupled with adversary search to project future story histories, evaluate them, back-up the total history evaluations to rate the "goodness" of the next immediate story moves, and select the best move [Weyhrauch 1997]. In the case of Weyhrauch's work, he discovered certain symmetries in his evaluation function that allowed him to memoize his search, thus explicitly constructing the policy table off-line and just doing table lookup when performing on-line story guidance. Even in the absence of such symmetries, reinforcement learning techniques can be used to learn a policy from adversary search, as in the case of Tesauro's world championship backgammon program, which learned a policy for playing backgammon from playing thousands of games against itself [Tesauro 1995]. Other drama managers may implicitly specify the table by representing local knowledge on the individual story pieces, and somehow use this knowledge distributed over the story pieces to select the next story piece. This is the case with the *Façade* drama manager.

The Beat Manager

In *Façade* the story pieces are beats. The drama manager is responsible for sequencing beats. This beat manager consists of two parts: a beat language in which local knowledge about beats is specified, and the beat manager itself, which uses this local knowledge, plus some global knowledge about desired story arcs, to select a beat.

The Beat Language

Figure 8-1 is an example schematic beat demonstrating the various constructs in the beat definition language. The beat compiler compiles beat definitions into java; the beat manager uses the compiled representation of beats to sequence beats. Every beat has a unique name and beat ID (an integer) – it is often convenient to use an enumeration constant for the beat ID.

Selection Knowledge

There are six types of selection knowledge that influence the selection of a beat. A brief description of each of these knowledge types is given here; the section describing the beat sequencing algorithm (page 150) provides a complete description of how these six knowledge types combine to yield a beat sequencing decision.

precondition {<wme test>} defines a working memory element (WME – first described on page 66) test over the current contents of story memory. The precondition must be satisfied in order for the beat to be a candidate for selection.

weight <float> defines a static weight modifying the probability of this beat being selected. For example, if the weight is 2.0, the beat is twice as likely to be selected (assuming its precondition is satisfied) as otherwise. If no weight is specified, the default weight is 1.0.

weight_test <float> {<wme test>} defines an associated weight/WME test pair. If a WME test is true (generally tested over the preconditions), the probability that the beat will be selected is multiplied by the weight. If a beat defines both a static weight and a weight test, the weight associated with a satisfied weight test overrides the static weight. If multiple weight tests are true, the test with the largest associated weighting factor is used.

```
defbeat ExampleBeat {
    // Local beat variables
    BeatStartWME exampleVariable = null;

    beatID eBeatID_ArtistAdv_GPA_T1;

    precondition {<WME test>} // Can have >1 precondition

    effects (tension 1); // one or more story variable effects

    priority 5; // static priority
    priority_test 10 {<WME test>} // can have >1 priority test

    weight 1.0; // Static weight - 1.0 is the default weight
    weight_test 2.0 {WME test>} // Can have >1 weight tests

    init_action {<code to execute at the beginning of the selection process>}
    select_action {<code to execute when a beat is selected>}
    succeed_action {<code to execute if the beat succeeds>}
    abort_action {<code to execute if the beat aborts>}
}
```

Figure 8-1. An example schematic beat expressed in the beat definition language

priority <float> defines the static beat priority. Beats are selected for sequencing by a weighted random draw from the beats with satisfied preconditions in the highest priority tier. If no priority is specified, the default priority is 0.

priority_test <int> {<wme test>} defines an associated priority/WME test pair. If the test is true, the beat is given the specified priority. If a beat defines both a static priority and a priority test, the priority associated with a satisfied priority test overrides the static priority. If multiple priority tests are true, the test with the largest associated priority is used.

effects <story value changes> defines the changes the beat makes to story values if the beat completes successfully. Story values are represented as a named floating point value stored in a WME in story memory. Story value changes are described using name/value pairs. If a beat causes changes to multiple story values, then the effect is specified with multiple pairs. Changes to story values are either absolute, in which case the completion of the beat changes the story value to a specific value, or relative, in which case the completion of the beat increments or decrements the story value. To specify a relative story value change, the story value is preceeded by inc (for increment) or dec (for decrement). Absolute changes are specified by leaving out the inc or dec. For example, the beat in Figure 8-1 changes the story value named "tension" to an absolute value of 1. As described below, a beat's effects contribute to the probability of selecting a beat depending on how well the effects match a desired story arc.

Actions

Beats can define actions that are performed at various stages in the beat selection process. The init_action is performed at the beginning of the beat selection process. Init actions are typically used to initialize any beat state (perhaps stored in beat scope variables) that will be needed during the selection process. The select_action is executed if a beat is selected for sequencing. Select actions are typically used to activate the beat behaviors (beat goals, handlers and performance behaviors) and set any WME state that the beat behaviors will need. The succeed_action is executed if a beat succeeds, while the

abort_action is executed if the beat aborts. Both succeed and abort actions are typically used to update state in story memory (the updating of story value changes as specified by effects takes place automatically). Action code is written in java.

Beat-Scope Variables

Beats can define beat variables that are accessible by all tests and actions within a beat. The beat environment is persistent – beat variables maintain their values across beat selection cycles. If the author desires that some variables should be initialized at the beginning of a beat selection cycle, they can be initialized in an init_action. Beat variables can appear in WME tests – if a test binds a beat variable to a WME or WME field value, the binding is visible to all other tests and actions. If a name *not* declared as a beat variable is bound within a WME test, an appropriately typed local variable is implicitly declared within the test, supporting the private chaining of bindings within a test.

Beat scope variables may be both bound and accessed within all tests and actions; it is therefore important to understand the order in which variables are bound. The first variable binding occurs when variable initialization statements are executed during beat construction. A beat is constructed only once when the drama manager starts up. Initializers can be used to ensure that variables have a meaningful value from the very beginning, and to establish variable values that will be used across all beat selection cycles. In Figure 8-1, the variable exampleVariable is bound to null in a variable initialization statement.

During a beat selection cycle, further bindings occur in this order:

1. init_action
2. preconditions
3. priority tests
4. weight tests
5. select_action
6. succeed_action or abort_action

For preconditions, priority tests and weight tests, beat variable bindings are only visible outside of the test if the test succeeds. A way to think about this is to imagine that within tests all variables are treated as local to the test; for those local variables that have the same name and type as a beat-scope variable, the value of the local variable is copied to the beat-scope variable if the test is successful.

Preconditions are evaluated in an arbitrary order. Precondition evaluation stops with the first precondition to succeed. Thus, for a beat with n preconditions, if the m^{th} (in some order) precondition (m <= n) succeeds, then visible bindings are established for any beat variables bound in that m^{th} precondition.

Priority tests are evaluated in the order of their associated priorities. Test evaluation order is unspecified for tests with the same priority. Visible variable bindings are established for any beat variables bound in the first successful priority test.

Weight tests are executed in the order of the associated weights. Test evaluation order is unspecified for tests with the same weight. Visible variable bindings are established for any beat variables bound in the first successful weight test.

The Beat Sequencer

Given a collection of beats represented in the beat language, the beat sequencer selects beats for sequencing. A sequencing decision is initiated when the current beat terminates, either successfully or by aborting (beat sequencing is automatically initiated at the

beginning of the story). Beat behaviors are responsible for monitoring their own success or failure criteria and informing the beat manager that the current beat has terminated.

The steps for making a beat sequencing decision are as follows:

1. Execute the init_action (if defined) on all beats that have not been previously sequenced (unused beats). This initializes any beat-specific state that may play a role in beat selection (e.g. affecting the outcome of a WME test).
2. Evaluate the preconditions for all the unused beats. This computes the set **Satisfied** of beats with satisfied preconditions.
3. Evaluate the priority tests of each beat in **Satisfied**. Assign each beat a priority as follows. If no priority test on the beat is satisfied (or no priority tests are defined), then assign the static priority, or 0 if no static priority is defined. If one or more priority tests are satisfied, assign a priority that is the max of the priorities associated with the satisfied priority tests. Collect the beats in **Satisfied** that are in the highest priority tier (share the same highest priority value) into the set **HighestPriority**. The selection algorithm will eventually select a beat by a weighted random draw from **HighestPriority**.
4. Score each beat in **HighestPriority** using the effects to compare the beat with the desired arc – this produces the set of scored satisfied beats **ScoredHighestPriority**. The details of this scoring algorithm appear below. The score defines the initial probability distribution for choosing a beat from **ScoredHighestPriority**. If no story arc is specified (or no beats have effects defined on them), then the initial distribution is flat (equal chance of choosing any beat from **ScoredHighestPriority**).
5. Evaluate the weight tests of each beat in **ScoredHighestPriority**. Assign each beat a weight as follows. If no weight test on the beat is satisfied (or no weight tests are defined), then assign the static weight, or 1.0 if no static weight is defined. If one or more weight tests are satisfied, assign a weight that is the max of the weights associated with the satisfied weight tests. Multiply each beat's score by the weight – this produces the set of weighted beats **WeightedScoredHighestPriority**.
6. Randomly draw a beat from **WeightedScoredHighestPriority** according to the probability distribution defined by the weighted score. The selected beat will be the next beat sequenced.

Beat Scoring Using Effects and Story Value Arcs

The initial probability of a beat being selected for sequencing, prior to the application of weighting terms, is determined by a beat's score, that is, by how well the beat's effects match the specified story arc. This section describes the algorithm for scoring a beat. Figure 8-2 depicts the simplest possible story target, one story value with a linear target arc. The story is trying to change the story value X. The initial value of X is 10, with a target value between 40 and 50. The minimum length of the story is 9 beats with a maximum length of 12 beats. In this case, the ideal story value arc is a line from the initial value of X to the average target value progressing over the maximum number of beats.

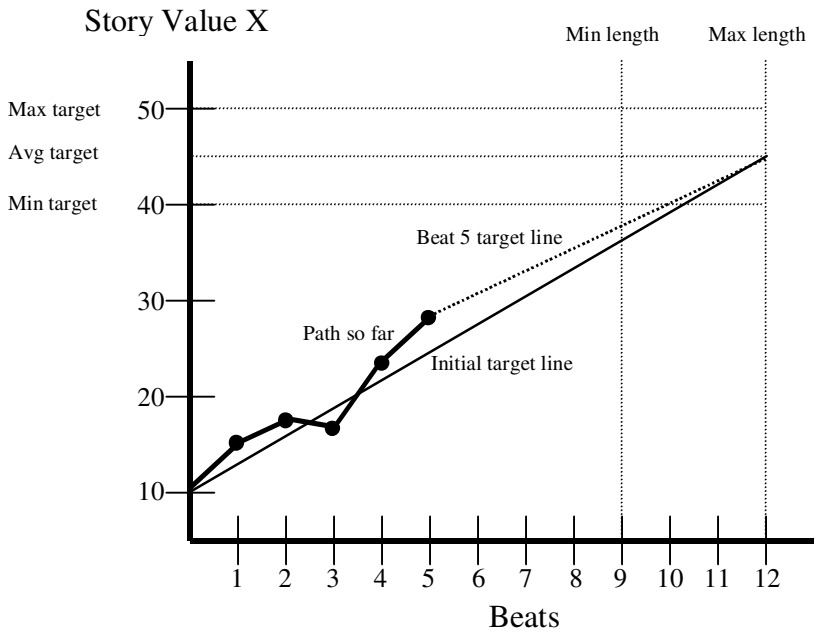

Figure 8-2. Example beat scoring situation with one linear story arc

The beat scoring algorithm makes use of the definitions in Table 8-1.

Definition	*Expression*
Initial value of X	$X_{initial}$
Average target value of X	X_{avg}
Value of X after beat n	X_n
Optimal value of X given by initial target line	X_{nopt}
Maximum number of beats	$beat_{max}$
Minimum number of beats	$beat_{min}$
Slope of the initial target line	$slope_{initial} = (X_{avg} - X_{initial}) / beat_{max}$
Slope of the adjusted target line for choosing the n+1 beat	$slope_{n+1} = (X_{avg} - X_n)/ (beat_{max} - n)$
The delta value change for story value X caused by a candidate beat	$deltaX_{beat}$
Candidate beat score	$score_{beat} = 1 / e^{\lvert slope_{n+1} - deltaX_{beat} \rvert}$

Table 8-1. Values employed by the beat scoring algorithm

At the beginning of the story, the beat manager wants to select beats so as to make the story value follow the initial target line. This means that ideally the very first beat would change the story value so as to leave it right on the target line, that is, $X_1 = X_{1opt}$. So the

beat manager wants to pick the beat whose $\text{delta}X_{beat}$ is closest to $\text{slope}_{initial}$, that is, the beat with the highest score_{beat}. Notice that the score function returns 1 when a beat's delta story value (as given by the effects slot) is equal to the desired target slope, moving towards zero as the beat's delta story value becomes more and more different from the desired target slope.

Now consider how beats are scored once the story is underway. For example, consider the scoring for selecting beat six (beat five has just successfully completed, so the situation looks like Figure 8-2). As beats have been sequenced, the actual story value X has not been exactly following the initial target line. The beat manager wants to get the story value arc back on track. A new target line is drawn from the current story value X_5 to the average target value X_{avg}. The new slope_{n+1} is used as the basis for computing the scores of candidate beats.

In the case of choosing a beat when the actual story value X_n is different from the optimal story value X_{nopt}, there are a number of strategies one could follow for trying to get the value back on track. Instead of drawing the new target line from X_n to X_{avg}, why not draw the new target line from X_n to X_{n+1opt}, that is, from the current story value to the optimal story value for the next beat? This approach would constantly try to force the value back to the initial target line as quickly as possible, thus minimizing the cumulative error. However, this approach would tend to make the actual value trajectory spiky. Whenever $X_n \neq X_{nopt}$, the system would tend to prefer larger delta_{Xbeat} than the approach described above. But one would like to maintain independent control over whether the story value arc is followed smoothly, without high frequency oscillations, or turbulently; to maintain independent control over turbulence, the base beat scoring algorithm shouldn't introduce something that feels like turbulence. The choice of drawing the new target line to X_{avg} tries to smoothly modify the trajectory such that the final trajectory has the same average slope as $\text{slope}_{initial}$ without introducing sharp changes.

One can imagine that the collection of beats might not allow the beat manager to do a good job tracking the desired value trajectory – at any point in time the candidate beat set just doesn't offer beats with good value increments. Regardless of how far off the value increments are, there will be some beat (or beats) in the candidate beat set that have the best (albeit small) score. As these low-scoring beats are chosen, the cumulative error[26] between the actual value arc and the desired value arc will steadily grow. Other than keeping track of the cumulative error, the beat manager provides no built-in mechanism for dealing with the situation of high cumulative error. Beat management could also fail by having no candidate beats available (beats with satisfied preconditions) given the current story state, or by not achieving a story value between the minimum and maximum within the maximum beat length. When one of these failure conditions occurs, it means that we, as authors, have not given the beat manager a collection of beats that allows it to achieve the specified targets given the player's interaction. Authors essentially give the system beat sequencing knowledge at two different levels, the selection knowledge on individual beats expressed in the beat description language, and the story arc. When a failure condition occurs, such as an overly large error term or an empty candidate set, this means that the beat collection is not rich enough to approximate the story arc given the player's interaction. Beat sequencing failures can be used while developing the story to determine that beats must be changed or new beats created, or that the target story arc should be changed. If the beat manager is being used as part of a hierarchical drama

[26] The beat manager keeps track of $\text{error}_n = \text{sqrt}(\sum_{i=1 \text{ to } n} (A_i - A_{iopt})^2)$ (the standard square root of sum of squares error), though any error term could be used.

manager (e.g. a plot point manager selects plot points where a plot point is defined by a collection of beats and target value arc(s)), then beat management failures can be used during story play to abort the larger hierarchical story unit and invoke a drama management decision at the next hierarchical level (completely analogous to beat abortion causing the beat manager to make another beat sequencing decision). *Façade*, however, doesn't make use of drama management above the level of beat sequencing; thus beat management failures are only used during development to discover inadequacies in the beat collection.

After having considered the simplest beat selection case, we can now modify the algorithm to support the more complicated cases.

Multiple Story Values. In general, the beat manager may try to change multiple story values during a story. Each story value has its own target arc. The only modification to the basic algorithm is that a beat's total score is computed from a combination of the individual story value scores. For example, if a beat manager is trying to change two story values X and Y, a candidate beat would have a separate score for each story value $score_{beatX}$ and $score_{beatY}$ as computed by the algorithm above. A score combining function is used to combine the two scores into the single total score:
$score_{beat} = f(score_{beatX}, score_{beatY})$. An obvious combination function is just to average the individual story value scores, though more complicated functions can be implemented as needed.

More Complex Arc Shapes. The scoring algorithm is easily modified to handle additional arc shapes by restricting the arc shapes to piecewise linear curves (see Figure 8-3).

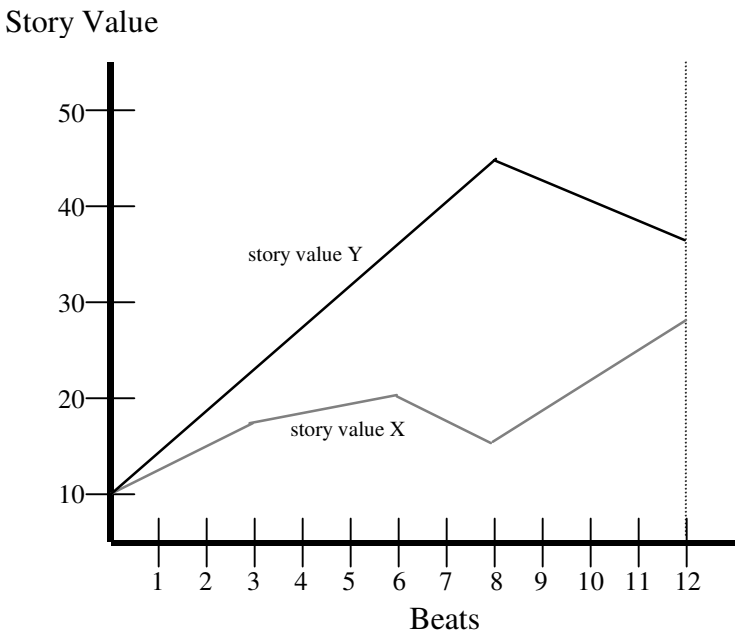

Figure 8-3. Piecewise linear target value arcs for two story values

Since each segment of a piecewise linear curve is a line, the algorithm above can be used for each segment. The only difference is that the target endpoint(s) used to compute $slope_{n+1}$ change as the beat count moves across different segments of the arc(s).

Turbulence, that is, strong swings in story values, can be introduced by creating rapid fluctuations in the piecewise linear target arc.

Polyvalent Beats. Polyvalent beats potentially cause different story value changes depending on what happens during the beat. Unlike standard beats, whose effects are represented as an implicitly and-ed together list of name/value pairs (e.g ($name_1$ $value_1$) … ($name_n$ $value_n$)), a polyvalent beat represents its effects as an or-ed list of and-ed name/value pairs. The effect of a polyvalent beat is thus somewhat uncertain; which list of effects will actually happen if the beat is sequenced is not known ahead of time. Presumably uncertainty about the player's response within the beat is responsible for the uncertainty about the effects. The scoring algorithm above can be modified to incorporate polyvalent beats by, for a given story value X, combining the individual $delta_{Xbeat}$ for each of the possible effects into a single total delta. An obvious combination function is to average the individual deltas, though more complex combination functions can be implemented as needed.

Story Memory

Just like WME tests in ABL agents, the WME tests in beats (preconditions, priority tests and weight tests) match against any publicly registered working memory. Most beat tests, however, match against the default beat manager memory, the story memory. The story memory is an instance of the same WorkingMemory class employed by ABL agents, and thus offers the same features, including (limited) support for episodic memory. In fact, episodic queries play a major role in the beat description idioms developed for *Façade* (described later in this chapter).

Automatically Maintained Drama Management State

Since beats make use of WME tests for preconditions, priority tests and weight tests, any arbitrary user-maintained WME state can be used for making beat management decisions. However, to facilitate writing beats, the beat manager automatically maintains some state that is useful for making beat management decisions. As the beat description idiom continue to grow and evolve, it is likely that additional state maintenance duties will be pushed onto the beat manager. Currently the beat manager maintains:

- BeatStartWME – Added to story memory when a beat is selected for sequencing. Contains a timestamp, a beat ID and a reference to the beat object (compiled form of a beat).
- BeatCompletedWME – Added to story memory when a beat succeeds. Contains a timestamp, a beat ID and a reference to the beat object.
- BeatAbortWME – Added to story memory when a beat aborts. Contains a timestamp, a beat ID and a reference to the beat object.
- BeatDistributionWME – Added to story memory during the beat selection cycle. Stores a timestamped copy of the beat distribution (collection of potential beats with probabilities) computed during the beat selection cycle.
- StoryStatusWME – A WME in story memory containing the beat count, beat ID of the previous beat, and beat ID of the current beat.

In addition to WME state, the beat manager provides a callback, getConflictSetAsBeatIDs() that returns the current conflict set as an array of beat IDs. The conflict set is the set of highest priority available beats from among which a beat will be chosen for sequencing. A conflict set is defined after the preconditions and priority tests have been executed; thus this callback can only be called within weight tests, select

actions, succeed actions and abort actions. If the callback is called within preconditions or priority tests, it returns null.

This concludes the general description of the beat manager. The next two sections describe the beat management idioms developed for *Façade*.

The *Façade* Story Design

Before describing the beat management idioms used in *Façade*, it is necessary to briefly describe the *Façade* story design. As the story design evolved, it put new constraints on, and suggested new opportunities for the beat manager, eventually resulting in the current beat manager design. And of course the very idea of beat management (globally structuring an interactive drama around dynamically sequenced beats) put constraints on and suggested new opportunities for the story design.

By "story design" we mean the structuring of a story as an interactive experience: the parceling of a story into resequenceable story pieces and the structuring of the story to support both local and global agency. For *Façade*, this means designing a collection of beats, including the design of the within-beat and across-beat interactivity supported by the beat collection, and specifying the story values and the relationship between beats and story values.

Spoiler warning: this section reveals detailed information about the story. If you have not yet played *Façade*, reading this section may spoil some of the pleasure and surprise of playing the story.

Story Topics

Façade's skeleton is made of story topics. The story topics define four interrelated foci through which the player interacts with the problems in Trip and Grace's marriage. The majority of beats in the beat collection are organized around these topics.

Artist/Advertising

Grace studied art and design in college and would have liked to pursue an art career. Her upper-middle class parents, however, didn't take this seriously as a career choice and pressured her to go into something "real" like advertising. Trip, whom she started going out with in her senior year, comes from a blue collar background – at several points during his childhood his family was in serious financial straights. This experience left him craving financial security and the yuppie accoutrements of a "successful" life. In college he studied business, and in fact hid his background from his friends, secretly working as a bartender to put himself through school. Because an art career is hardly the path to riches, he teamed up with Grace's parents in pressuring her to work in something "practical" like advertising or graphic design. Grace's resentment over his lack of support has been festering for ten years, though she also bares responsibility for lacking the self-confidence to stand up for her true desires, and for not truly being ready to give up the financial and material security she is used to in order to pursue this path. Her constant redecorating of their apartment is an (inadequate) outlet for her repressed interest in art. Recently she has started painting again without telling Trip – in fact, one of the new paintings on their wall, which she told Trip she bought, was actually painted by her.

Rocky Marriage

While Grace and Trip had genuine affection for each other in college, their marriage for the last ten years has been marked by repeated misunderstandings and negative interaction patterns. Shortly after meeting Trip in college, after they had only been dating for a few months, Grace took a painting class where she met Vince. While she couldn't seriously consider him as boyfriend, both because she was already dating Trip (though she didn't love Trip yet), and because her parents would never approve, she was quite attracted to him. Vince wooed her – the night before Christmas break of her senior year they slept together. During Christmas break, at Grace's parents house, Trip, after announcing that he had a great new finance job lined up in New York starting next summer, pulled out a diamond ring and proposed to Grace. With her family beaming on, Grace felt compelled to say yes, though she wasn't sure she loved Trip and felt guilty about recently having slept with Vince. Because of this , she feels like she jinxed the relationship from the beginning, though she feels angry that Trip put her on the spot. Over the years Trip, who genuinely loves Grace, has tried to "take care" of Grace, though to Grace this often feels like he's manipulating and controlling her. As the buried resentments have grown over the years, Grace has retreated into an "ice queen" persona, creating a negative interaction pattern in which Trip ever more desperately tries to "take care" of things and Grace ever more coldly withholds affection. They have sought marriage counseling in the past, but Grace broke it off. Recently Trip organized a vacation to Italy that was intended to be a kind of second honeymoon – it was a flop.

Façade

Trip and Grace collude in creating a social façade of material success, Trip because he craves it after a childhood of (what he perceives as) material deprivation, and Grace because it's what she's familiar with (she doesn't know how to break out of it). Trip enjoys visiting Grace's parents, and tries to emulate the snobbery of Grace's dad with his own fetishized cigar and wine collections. Grace hates watching how Trip acts around her parents. Grace enjoys visiting Trip's parents; she finds them "real" and enjoys the mashed potatoes and casserole dinners, and the loud boisterous conversation around the table. Trip is embarrassed by his family, but when he is finally able to let his guard down, Grace likes who he becomes. In college, shortly after they started dating, Grace and a group of her friends went "slumming" to local bars. At one of the bars she was surprised to see Trip bartending – Trip tried to hide his background from his friends and had not told Grace that he was working his way through college. She secretly watched him (he didn't see her) and admired his easy way with the customers, and his quiet efficiency. This incident made Trip more attractive to her, as she saw him as more "real" than the fraternity types she normally dated. Trip for his part is envious of Grace's background, though he also secretly resents it, feeling that he pulled himself up by his bootstraps while Grace had everything handed to her on a silver platter.

Trip's Affairs

Trip is a charismatic, attractive man who has a natural ability to charm people and get what he wants. He attracts women easily, and dated frequently in college. Over the years, as Grace has withdrawn her love and affection (including sex), Trip has had a number of affairs. His work requires frequent travel, and he has sometimes had affairs with women (often work colleagues) he meets on these business trips. Grace is somewhat aware of these affairs, but since part of the contract of their relationship is to not talk about

anything real, they never consciously acknowledge that the affairs are happening. However, "coincidently", Grace often has a redecorating fit while Trip is gone on business travels. Within the last year or so, Trip had an affair with a manager above him at the finance firm he worked at; the fallout from this affair resulted in him needing to leave the company. Trip got a job at the advertising firm Grace works at. They are now both working on the same project, print advertisements for a line of bridal gowns from a Spanish fashion designer, with Trip as the lead project manager and Grace as the lead creative. Trip has recently broken off an affair with Maria, a colleague he met in his trips to Spain for the bridal gown project. Maria gave him a gift of a brass bull, which he keeps on the work table. Trip told Grace that the bull was just a business present, but Grace suspects the truth and naturally doesn't like the brass bull. He really does want to save his marriage with Grace, realizes the affairs aren't helping, and has resolved to stop. He planned the disastrous Italy vacation as an attempt to start over. Maria, however, is not letting Trip off the hook so easily, and keeps calling him, much to Trip's distress. Just this morning Grace picked up the phone while Trip was on the line and heard Maria's voice, confirming Grace's suspicions.

Story Values: Tension and Affinity

Beats, in addition to conveying information in the four story topics, also change story values. The two explicitly tracked values in *Façade* are tension and affinity. Tension is the degree to which the stress in Trip and Grace's marriage is visible, the degree to which buried problems are coming to the surface. The climax occurs at a point of maximum tension. The drama manager softly guides the story along the tension arc appearing in Figure 8-4. Tension is an explicitly controlled story value appearing in the effect slots of beats. Tension takes on the integer values 1 (minimum tension) to 4 (maximum tension).

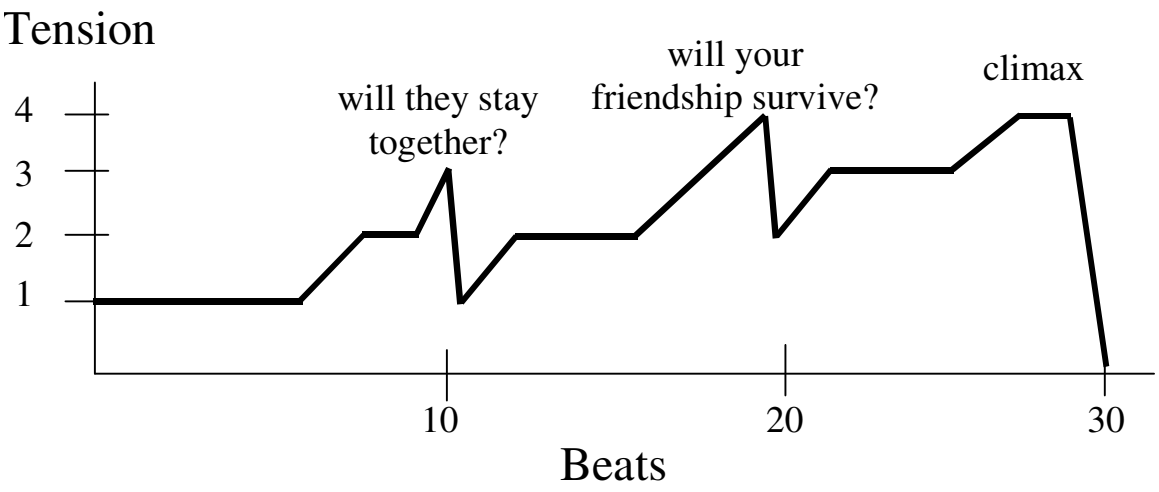

Figure 8-4. The *Façade* dramatic arc

The story is divided into three parts. The first part builds to a mini-climax that raises the dramatic question "Will Grace and Trip stay together?". The second part builds to a mini-climax that raises the dramatic question "Will your friendship with Grace and Trip survive?". The third part builds to the final story climax, resolving the two dramatic

questions. After each of the mini-climaxes, the tension momentarily relaxes, then builds quickly again to even greater heights than the last section.

Affinity is the degree to which Trip or Grace consider the player to be on their side. Affinity is zero-sum; the player can't have positive affinity with both Trip and Grace at the same time. At tension 1, the player can have affinity with Trip, be neutral, or have affinity with Grace. At higher tensions, the player either has affinity with Trip or with Grace (the player can't be neutral). Affinity is not explicitly managed by the drama manager. It can change from moment to moment, depending on how the player interacts within beats. In a sense, affinity is the local control through which the player navigates the story. Each beat sets up a little head game in which Trip and Grace use the player as a tool in their battle. Thus each beat provides opportunities to push Trip's or Grace's buttons, moving the affinity back and forth. Part of the pleasure of playing *Façade* will be pushing their buttons and watching them respond. But as the player pushes their buttons and changes affinity, this will influence beat sequencing decisions, thus changing what information is revealed and the global structure of the story.

At each tension there are one or more beats associated with each story topic. Within a single story topic, as the tension progresses, more of the façade is stripped away; in growing agitation and distress, Grace and Trip reveal more of the underlying problem associated with a story topic, both to the player and to themselves. For example, at tension 1, an *artist/advertising* beat may reveal that there is something unhealthy about Grace's redecorating; by tension 4 an *artist/advertising* beat reveals that Grace feels Trip ruined her life by dissuading her from art.

Story topic beats are organized as beat triplets (for tension 1) or beat doublets (for tensions greater than 1), one beat for each of the possible affinity values. The beats within each affinity set reveal roughly the same information, and use the same intra-beat logic (same structure of beat goals and handlers) – but the performance varies for the different members of the affinity set, including the exact dialog and nuances of the information revealed. The transition-out beat goals for story topic beats often change the affinity depending on how the player responded to the little interaction game established by the beat. However, a global mix-in response (see below) to an individual discourse act within the beat may also change the affinity. Affinity sets are designed to allow affinity switching in the middle of the beat. For example, if, during a tension 1, Grace affinity, *artist/advertising* beat, the player does something to change the affinity to neutral, the tension 1 beat triplet is designed such that when the Grace affinity beat aborts, the neutral affinity member of the triplet will be selected by the beat manager and has the appropriate transition-in beat goals to reestablish context and pick up with a neutral affinity performance.

Story topic beats don't just reveal information by having the characters blurt it out. Each beat is structured so as to provide clear interaction choices for the player, usually in the form of an affinity game. Affinity games are head games in which the character's snipe at each other in such a way as to offer the player clear opportunities for siding with one character or the other, giving the player an opportunity to reinforce the current affinity or change it. Thus each beat in a story topic affinity set is a little mini-story offering its own interaction game, and along the way revealing more information about the story topic while changing the tension.

Story topic affinity sets provide coverage. They guarantee that for any combination of affinity and tension, there is some beat for the beat manager to select that moves the story forward. Any topic can follow any other topic; in the event that the player doesn't bring

up a specific topic[27], the beat manager can select one randomly. But by their very nature of providing coverage, story topic affinity sets provide poor global agency. Since any one can follow another, their appearance in the story doesn't depend on the *longer term* structure of the story and the player's interaction. They are similar to the case of the fully permutable set, though the tension arc provides some global structure. The story topic affinity sets provide a skeleton guaranteeing coverage; additional beats are added to the beat collection to provide opportunities for global agency.

Additional Beats

Story topic singlets, support beats, and mini-climax beats all provide opportunities for global agency. Story topic singlets are single topic beats, available only at specific tensions and affinities, which reveal additional information about the topics. Often these beats will have additional selection constraints conditioned on temporal patterns of activity. For example, the *rocky marriage* beat *Romantic Proposal* is only available when Grace has affinity at tension 2. Further, Grace must have had the affinity for several beats in a row. In this beat, Trip, in desperation, tries to look good in the player's eyes by describing the "romantic" way he proposed in front of Grace's family. Grace hints that she wasn't really ready to marry Trip. The sequencing of this beat depends on longer term decisions that the player makes, in this case the longer term decision to continue backing Grace rather than just pushing the character's buttons willy-nilly. Such beats provide the player with an opportunity to have a more global effect on the story.

Support beats are non-story topic beats fulfilling other story functions. For example, when the player first arrives at the door, several beats may be required to handle the greeting. This greeting beat cluster must successfully complete before any story topic beats can start sequencing. Another support beat, *Encourage Player Wander*, becomes more probable if the player doesn't move around the room much. In this beat, Trip and Grace go to the kitchen to get some appetizers, encouraging the player to "take a look around". This hopefully encourages the player to move around the room and potentially physically favor an object (by looking at or picking up an object).

The mini-climax beats have a complex beat goal structure that is conditioned on which specific story topics happened in the story segment (part I or II), and potentially on other patterns of interaction, such as whether the player took the side of one character more than the other. While the story function of the mini-climax is to raise a major dramatic question ("Will Grace and Trip stay together?" for part I, "Will they still be friends with the player?" for part II), the way the question is raised refers back to the specific story topics and interaction patterns that occurred in such a way as to invite the player to reflect on the global structure of the story.

Global Mix-Ins

During a beat, there are many possible interactions that have no role within the beat and thus no beat-specific response. Rather than aborting the beat, which would make forward

[27] To bring up a story topic, the player may directly refer to it (e.g. "Is your marriage all right" → ReferTo(object: RockyMarriage)), but there are indirect ways to refer to story topics as well. Objects in the apartment are associated with different story topics, so physically favoring an object by walking up to it and looking at it becomes both an object reference and an indirect story topic reference. Additionally, there are satellite topics, such as children or divorce, that are associated with story topics and thus cause indirect references.

progress in the story difficult, or ignoring the interaction, which would decrease player agency, global mix-ins handle the non-beat-specific interaction. Global mix-in reactions are proposed by the global proposer context of phase II of NLP (see page 214), and handled by global handlers that interrupt the current beat goals, mix in some additional goals, reestablish context, and restart the beat goals. More detail about global mix-in reactions appears in Chapter 10. For the purposes of this chapter, it is only important to note that not all the action during a beat is coming from the beat behaviors – some of it is coming from global mix-ins. Further, the global mix-ins are themselves a source of additional story content – the player is rewarded, rather than punished, for interacting in a maner not handled by the local beat context.

Beat Collection for Part I

Given the general description of the story topics, story values, and beat types above, this section describes the collection of beats for part I. Examples of story sequences constructed from this collection of beats appears on page 170, after the discussion of beat description language idioms.

Story Topic Triplets and Doublets

In part I, story topic beats are organized into tension 1 triplets, one each for neutral, Grace and Trip affinity, and tension 2 doublets, one each for Trip and Grace affinity. Given particular content drawn from one of the story topics, such as Grace's decorating being a sublimation of her desire to be an artist, a collection of five beats (a tension 1 triplet and a tension 2 doublet) makes use of that content. In any one run-through of the story, only one of these five beats will actually be performed. For example, if the player experiences a beat about Grace's decorating at tension 1, the tension 2 decorating beats are unavailable. In principle, the tension 2 decorating beats could continue to progress the decorating topic, and could thus still be available after a tension 1 decorating beat has been performed. However, to save authorial effort, the tension 2 beats make use of the same interaction logic and basic content as the tension 1 beats. Though the tension 2 versions of the story topic beats are performed differently (since they are at tension 2), and reveal more information (since Trip and Grace are more upset), they are similar enough to the tension 1 versions that they both can't appear in the same run-through.

There are seven story topic beat collections, each consisting of a tension 1 triplet and a tension 2 double. Each beat collection draws from one of the four story topics. The seven beat collections are described below.

***Decorating* (*artist/advertising*).** Grace puts down her own decorating while Trip praises it. Grace's constant redecorating is a sublimation of her desire to be an artist. Grace wants the player to agree that something is wrong with the decorating (and thus implicitly agree that something is wrong with Grace denying her desires), while Trip wants the player to praise the decorating. The affinity changes towards whomever the player agrees with. Redecorating is discussed in the context of specific objects in the room, such as the couch, paintings on the wall, or Grace's trinket collection. Which object is the focus of the redecorating beat depends on player interaction.

***Ask About Drinks* (*façade*).** Trip offers the player something to drink from the bar. In the process, he shows off his "refined" tastes by suggesting fancy drinks. Grace counters with a different choice, while suggesting that T is acting like a fake yuppie and is too

much like her dad. Affinity changes towards whomever's drink suggestion the player accepts.

***Work Project* (*Trip's affairs*).** Trip brags about landing the new work project. In the process he brings up the brass bull, mentioning that it is a present from a client. Grace likes the project, but complains that Trip is too controlling. She also hints at her suspicions regarding the brass bull.

***Italy Vacation* (*rocky marriage*).** Trip tries to get the player to come over and look at the photo, which Trip has framed and hung on the wall, from their recent Italy vacation. If the player goes over to the picture, Trip asks the player to guess one word that describes the picture – he's looking for words like "love" and "romance". Grace tries to get the player to come over to another part of the room. If the player does go over to the picture, she makes sarcastic comments about the picture during the "guessing game".

***Work as Outlet* (*artist/advertising*).** Trip suggests that the new work project is a good outlet for Grace's "artistic tendencies". Grace is annoyed by this, claiming that advertising is not truly creative or satisfying. They force the player to take sides.

***Grace's Parents* (*façade*)**. Trip mentions how much he likes Grace's parents, particularly her dad, whom he finds suave and sophisticated. Grace thinks her parent's lifestyle is shallow. They force the player to take sides.

***Trip's Parents* (*rocky marriage*).** Grace describes how she likes Trip's parents, how she finds Trip's dad so "real", and how Trip's parents seem to have such a loving marriage. She complains that Trip never wants to visit them. Trip thinks his dad is financially irresponsible. They force the player to take sides.

Story Topic Singlets

The story topic affinity sets (triplets and doublets) insure coverage. Given any value for tension and affinity, there is always a story topic beat from some affinity set that is available to be sequenced. Story topic singlets provide opportunities for global agency. They are only available to be sequenced when specific patterns of activity have occurred.

***Romantic Proposal* (*rocky marriage*).** Only available if tension is 2 or can move from 1 to 2 (at least two tension 1 beats have happened), and the player has had affinity with Grace for the last three beats. Trip is desperate to look good to the player. He describes his "romantic" proposal to Grace in front of her parents. Grace hints that she wasn't ready. They force the player to take sides.

***Move Wedding Picture* (*artist/advertising*).** Only available if tension is 2 or can move from 1 to 2, and the player has had affinity with Trip for the last three beats. Grace is feeling desperate and upset. Trip casually brings up the wedding picture, mentioning how everyone comments on it when they first come over. A visibly upset Grace says that she wants to move it out of the room because it doesn't go with her new decoration scheme. She says that she should at least have control over *something* in her life. They force the player to take sides.

***Forgot Cheese* (*façade*).** Only available if the tension is 2 or can move from 1 to 2, and Trip is leaving the room for a moment. Trip may leave the room because he becomes upset during another beat or during a global mix-in. An upset Trip heads to the kitchen, ostensibly to bring out some cheese. He discovers there's no cheese left and nails Grace for forgetting to buy it.

***Veronica* (*Trip's affairs*).** Only available if the tension is 2 or can move from 1 to 2, and Grace is leaving the room for a moment. Grace may leave the room because she becomes upset during another beat or during a global mix-in. While Grace is out of the room, Trip invites the player to a party, asking the player to bring along a pretty co-

worker named Veronica, whom the player has introduced Trip to in the past. Grace, returning just at this moment, gets mad at both the player and Trip for "conspiring" to invite pretty, eligible women to the party.

Greeting Cluster

The greeting cluster is a collection of beats that handle the greeting at the door. Currently the greeting cluster consists of four beats that always occur in a fixed order at the beginning of the story. Additional beats can be added to the greeting cluster in order to support beat-level variation during the greeting.

Player Behind Door. *Façade* starts with the player standing outside Grace and Trip's apartment door. A muffled argument can be heard through the door. When the player knocks, the argument stops.

Trip Greets Player. Trip opens the door and effusively greets the player. Trip invites the player inside.

Trip Fetches Grace. A small, minimally interactive beat during which Trip runs into the kitchen to fetch Grace. Trip and Grace have a short, barely intelligible argument in the kitchen before emerging from the kitchen with broad smiles. This beat gives the player a moment to look around the apartment while Trip and Grace are in the kitchen.

Grace Greets Player. Grace, emerging from the kitchen, effusively greets the player.

Miscellaneous Support Beats

Part I currently has only one miscellaneous support beat. Additional support beats (such as the *Encourage Player Wander* beat described on page 160) can be added as desired.

Explain Dating Anniversary. Trip remembers that tonight is his and Grace's dating anniversary. Ten years ago tonight the player introduced Grace and Trip at a party. Trip asks the player "Do you remember that?". If the player "remembers", affinity moves towards Trip, otherwise affinity moves towards Grace.

Mini-Climax

The part I mini-climax raises the first dramatic question: "Will Grace and Trip stay together?". During the mini-climax, Grace and Trip have a big argument, using material from the beats that have occurred in the story so far as fodder. The argument summarizes their interpretation of what has happened so far. At the end of the argument, one of them storms out of the room. Who is more upset is a function of who's side the player has tended to be on.

Beat Management Idioms

With this brief description of the *Façade* story design, we can now look at the beat management idioms used to implement this design

Beat Clusters

Beat clusters are a small collection of beats that together accomplish some story function that must be atomically completed before other beats can be sequenced. An example is the greeting cluster, which accomplishes the social function of greeting the player when

she first arrives at the apartment[28]. *Façade* starts with the player standing outside the closed door of Grace and Trip's apartment. Through the door a spat can be heard – when the player knocks either Grace or Trip comes to the door and invites the player inside. Whoever didn't come to the door then comes out and greets the player. The beat cluster that accomplishes this is: *Player Behind Door*, *Trip Greets Player*, *Trip Fetches Grace*, and *Grace Greets Player*. When this cluster successfully completes the greeting, the rest of part I can start being sequenced.

Player Behind Door always happens at the beginning of the story. The representation of this beat in the beat description language appears in Figure 8-5.

```
defbeat PlayerBehindDoor {
    beatID eBeatID_PlayerBehindDoor;

    precondition { (StoryStatusWME beatCount == 1) }
    effects (tension 1);
    priority eBeatPriority_max;
}
```

Figure 8-5. Beat *Player Behind Door*

The precondition specifies that the beat is only available for sequencing at the beginning of the story, when the beat count is 1. Most beats won't care about the exact beat count and so won't test it in their preconditions; many other beats will therefore have satisfied preconditions at the beginning of the story. One way to prevent one of these other beats from being selected would be to thread a condition that the greeting must have happened through the preconditions of all other non-greeting beats (most of the beats in the total beat collection). But of course one would like to avoid having to touch every other beat; priority provides a way to represent this locally. The static priority of this beat is the maximum possible priority, so regardless of what other beats have satisfied preconditions, they won't be in the same priority tier as *Player Behind Door*, and so can't possibly be chosen. The effects specify that the value of tension will be 1 at the end of the beat – since the tension story value starts with a value of 1, and the tension arc wants tension to stay at 1 for awhile, this means that the delta tension of this beat is 0, which is the same as the slope of the target line, and thus matches perfectly (of course it wouldn't matter if it didn't since the beat is in the highest possible priority tier). The beat behaviors of *Player Behind Door* don't condition on any beat management state, so no BeatArgumentWMEs need to be created in story memory. Therefore no select_action is needed.

The rest of the beats in the greeting cluster are similar to *Trip Greets Player* in Figure 8-6. The precondition is used to chain the sequencing within the cluster; the previous beat must be the spat behind the door. The priority eBeatPriority_greeting is greater than any non-greeting beat, but less than the maximum priority. This ensures that we will stay within the greeting cluster until no greeting beats have satisfied preconditions, while ensuring that the spat happens first.

One can imagine adding more beats to the greeting cluster, in order to allow more flexibility in the greeting. For example, one could add beats in which Grace greets the

[28] Of course, the greeting beats don't only accomplish this social function. In a tightly structured drama, there is no room for wasted action; every beat must serve the purposes of the story in some way. In the case of the greeting cluster, the beats hint at some kind of marital trouble through both overly-enthusiastic greetings and assertions that "we're doing *great!*", as well as little quibbles.

player at the door, and Trip is the character who greets the player inside. In this case two beats, *Trip Greets Player at Door* and *Grace Greets Player at Door*, could potentially happen as the second beat. Both would have the same precondition – that the previous beat is the spat behind the closed door. In sequencing the second beat, both beats would appear in the final distribution, with the beat manager picking one randomly.

Using preconditions to chain directly on previous beat IDs is one of simplest and least flexible approaches to specifying beat logic. It is generally only useful for small beat clusters in which beats are guaranteed to complete successfully (not abort). If, within the cluster, it is possible for beats to abort, then the preconditions should test something more abstract than specific beat IDs. For example, in the greeting cluster, one could test whether Grace and Trip have both greeted the player or not, using succeed actions to set this state.

```
defbeat TripGreetsPlayer {
    beatID eBeatID_TripGreetsPlayer;

    precondition {
      (StoryStatusWME previousBeatID == eBeatID_PlayerBehindDoor)
    }
    effects (tension 1);
    priority eBeatPriority_greeting;
}
```

Figure 8-6. Beat *Trip Greets Player*

Support Beat Weight Boosts

The probability of sequencing a support beat often changes as a story segment progresses. Weight tests are used to capture this changing probability. For example, consider the beat *Explain Dating Anniversary*, in which Trip explains that today is their 10 year dating anniversary and in fact that you, the player, introduced them. As authors we feel that this beat makes most sense near the beginning of part I. While it still makes sense for it to appear later in part I (but still at tension 1), when playing the story multiple times, more often than not we would like this beat to be sequenced earlier rather than later. This beat appears in Figure 8-7.

```
defbeat ExplainDatingAnniversary {
    beatID eBeatID_ExplainDatingAnniversary;

    precondition { (TensionStoryValueWME value == 1) }
    effects (tension 1);
    weight_test 2.5 { (StoryStatusWME beatCount == 5) }
    weight_test 2.0 { (StoryStatusWME beatCount == 6) }
    weight_test 1.5 { (StoryStatusWME beatCount == 7) }
}
```

Figure 8-7. Beat *Explain Dating Anniversary*

Three weight tests are used to boost the priority early in the story. Given the greeting cluster above, four beats will always be sequenced from the cluster, so the beat count of the first non-greeting beat is 5. While this beat is potentially sequenceable at any point during tension 1, all other things being equal, it is 2.5 times as likely to be sequenced at beat 5, decreasing to no weight boost by beat 8. Often, however, all other things are *not* equal. There are other support beats that also have weight boosts early in the story; the

final probability of selecting this beat also depends on the number and weights of the other beats in the distribution. And priority trumps weight, so if there are any satisfied beats with priority higher than *Explain Dating Anniversary*, then it can't be chosen.

The weights associated with weight tests can be less than one, to capture conditions in which it should be less likely than usual to select a beat. And weight tests, being general WME tests, can certainly test more complex state than the simple beat count. For example, one can imagine a beat *Encourage Player Wander* in which Grace and Trip invite the player to look around the apartment while they both run off quickly to the kitchen to grab some appetizers. The weight test for this beat is:
weight_test 4 { (StoryStatusWME beatCount >6) !(DAWME id == eDAType_referTo) }.
This weight test specifies that *Encourage Player Wander* should become 4 times as likely to be sequenced if 6 beats or more have happened and the player hasn't referred to anything (a story topic, an object, etc.). The purpose of this beat is to encourage the player to wander around the apartment for a moment and hopefully refer to something (that will then influence beat selection) by physically favoring an object.

Story Topic Affinity Sets

A story topic affinity set is a collection of beats at a given tension that reveals information about a specific story topic in a manner appropriate for the different affinities possible at that tension (see Story Values: Tension and Affinity on page 158). Here we walk through a simplified example of the beat description language idiom for story topic affinity set beats, using the Grace affinity (GA) version of the tension 1 *Decorating* (*artist/advertising*) triplet as an example. The beat variable declarations and init action appear in Figure 8-8.

```
StoryMemory memory = DramaManager.getDramaManager().getStoryMemory();
BeatStartWME lastBeatStart = null;
BeatStartWME beatBeforeLastStart = null;
DAWME unhandledAct = null;
int daParam = eDAMiscParam_$NOPARAM;
BeatAbortWME abortWME = null;

init_action {
    unhandledAct = null;
    daParam = eDAMiscParam_$NOPARAM;
    abortWME = null;
    lastBeatStart = memory.findPrev("BeatStartWME", currentTime());
    if (lastBeatStart == null)
      beatBeforeLastStart = null;
    else
      beatBeforeLastStart = memory.findPrev("BeatStartWME",
        lastBeatStart.getTimestamp());
}
```

Figure 8-8. Beat variable declarations and init action for the *Decorating* (*artist/advertising*) tension 1, GA beat

The beat variables are used in the various beat tests, and, other than memory, which is just a convenience variable providing a short reference for the story memory, are initialized at the beginning of every beat description cycle in the init action. The variables lastBeatStart and beatBeforeLastStart are initialized using episodic memory queries on the story memory; they are bound to the BeatStartWMEs for the most recent two beats.

BeatStartWME is a subclass of TimestampedWME, and thus supports the various episodic memory queries defined on page 98 (as previously noted, WME test syntax does not yet support episodic queries, and thus such queries must currently be made by directly calling the appropriate support methods on a working memory).

The precondition and effects appear in Figure 8-9.

```
precondition{
    (AffinityWME affinity == eAffinity_grace)
    (TensionStoryValueWME value == 1)
    !(BeatCompletedWME  beatID >= eBeatID_ArtistAdvertising_FIRST_TENSION1
      beatID <= eBeatID_ArtistAdvertising_LAST_TENSION1)
}

effects (tension 1);
```

Figure 8-9. Precondition and effects for the *Decorating* (*artist/advertising*) tension 1, GA beat

The beat can only happen if Grace has the affinity, the tension is 1, and no other member of the affinity triplet has successfully completed. This last condition is tested by requiring that there be no BeatCompletedWME with a beat ID that is a member of the affinity triplet.

The priority tests appear in Figure 8-10.

```
priority_test eBeatPriority_affinitySwitch {
    (lastBeatStart != null)
    (lastBeatStart.getBeatID() >= eBeatID_ArtistAdvertising_FIRST_TENSION1
    || lastBeatStart.getBeatID() <= eBeatID_ArtistAdvertising_LAST_TENSION1)
    abortWME = (BeatAbortWME
      beatAbortReason == eAbortReason_affinitySwitch
      timestamp > lastBeatStart.getTimestamp())
}

priority_test eBeatPriority_discourseAct {
    unhandledAct = (DAWME timestamp :: daTimestamp
      handledStatus != eDAHandledStatus_dramaManagerHandled
      id == eDAType_ReferTo
      param1 == eDAReferToParam_artistAdv
      param1 :: daParam)
    (daTimestamp > lastBeatStart.getTimestamp())
}
```

Figure 8-10. Priority tests for the *Decorating* (*artist/advertising*) tension 1, GA beat

The priority affinitySwitch is greater than discourseAct. AffinitySwitch is the beat priority if another member of the affinity triplet (either the neutral or Trip affinity version) aborted because of an affinity change in the middle of the beat. In this case, we generally want to choose another member of the same triplet with the appropriate affinity. The test consists of checking that the last beat started is a member of the *Decorating* tension 1 affinity triplet (by checking the beat ID of the most recently started beat), and that this beat was aborted because of an affinity change (by checking that there is a BeatAbortWME added *after* the most recently started beat with the appropriate beatAbortReason). The reason for a beat abort is set by the beat behaviors of the beat doing the aborting. When the beat behaviors decide that the context has changed such

that the beat can no longer continue, the behaviors call a method on the beat manager indicating the abort and pass an argument indicating the abort reason.

DiscourseAct is the beat priority if the player has recently referred to the story topic, in this case *artist/advertising*. For example, if the player said "I like these paintings" (or just went over and looked at the paintings on the wall), this would generate both a reference to the specific object and a reference to the story topic *artist/advertising*, since the paintings are related to this story topic. The priority test tests whether there has been a ReferTo discourse act referring to *artist/advertising*, that has not yet been handled by the beat manager, and has occurred "recently", where recently means sometime since the last beat started. The handledStatus field of a discourse act WME is used to record which hierarchical level of the architecture has responded to the discourse act. When an act is first created, no one has responded to it and thus it is unhandled. If a beat or global handler responds to the discourse act, the discourse act WME is marked as having been *beat handled*. If the drama manager makes a sequencing decision based on a discourse act, the WME is marked as having been *drama manager handled*.

Together, the two priority tests represent the conditions under which this beat should be strongly preferred – if we are already in the story topic triplet and are switching affinities, or if the player has referred to the story topic.

The weight test appears in Figure 8-11.

```
weight_test eBeatWeight_previousDiscourseReferTo {
    unhandledAct = (DAWME timestamp :: daTimestamp
      handledStatus != eDAHandledStatus_dramaManagerHandled
      id == eDAType_ReferTo
      param1 == eDAReferToParam_artistAdv
      param1 :: daParam)
    (daTimestamp <= lastBeatStart.getTimestamp())
    (daTimestamp > beatBeforeLastStart.getTimestamp())
}
```

Figure 8-11. Weight test for the *Decorating* (*artist/advertising*) tension 1, GA beat

The weight test captures the idea that if there is an old reference to *artist/advertising* that the drama manager has not yet handled, then we want to prefer this beat by making it more probable, but not prefer it as strongly as bumping it up into a higher priority tier. For example, imagine that during a non-story-topic beat such as *Explain Dating Anniversary*, the player refers to both the paintings and a framed photograph that Trip put up from their recent vacation in Italy. At the next beat sequencing decision, both a *rocky marriage* beat and an *artist/advertising* beat are going to be bumped to discourse act priority. Let's say a member of the *rocky marriage* triplet is chosen. After it completes, there is still a non-drama-manager-handled discourse act referring to *artist/advertising*. But because it is somewhat old now, we as authors don't feel that the story level should be compelled to respond to this reference – since the player brought it up more than a beat ago, it is alright if some other beat unrelated to *artist/advertising* happens now. But, all things being equal, it seems appropriate that an *artist/advertising* beat happen more frequently than usual because of the old reference; the weight test captures this.

The select and succeed actions appear in Figure 8-12.

```
select_action {
    memory.deleteAllWMEClass("BeatArgumentWME");
    if (abortWME != null)
      memory.addWME(new BeatArgumentWME(eBeatArg_txnIn_AffinitySwitch);
    else if (daParam == eDAReferToParam_artistAdv)
      memory.addWME(new BeatArgumentWME(eBeatArg_txnIn_ReferTo));
    else
      memory.addWME(new BeatArgumentWME(eBeatArgument_txnIn_Generic));
}

succeed_action {
    if (unhandledAct != null)
      unhandledAct.setHandledStatus(eDAHandledStatus_dramaManagerHandled);
}
```

Figure 8-12. Select and succeed actions for the *Decorating* (*artist/advertising*) tension 1, GA beat

Here the select action is used to set up the arguments for the beat behaviors. The beat manager passes arguments to the beat behaviors via BeatArgumentWMEs. In this case, there is a single argument specifying which transition-in the beat should use. The various tests have bound the beat variables abortWME and daParam. These two variables are used to determine what kind of transition-in the beat should perform, an affinity switch transition that reestablishes the within-triplet context in the event of an affinity switch, the refer-to transition that relates the beat to the player utterance referring to the *artist/advertising* topic, or the generic transition. The succeed action performs any state maintenance duties the beat should perform when it succeeds, in this case, in the event that the sequencing decision was made based on a non-drama-manager-handled discourse act (determined by whether the beat variable unhandledAct is bound or not), marking the discourse act WME as drama manager handled.

Tension 2 story topic doublets have the same structure as tension 1 triplets except for the precondition and effects. Unlike tension 1 beats, which require that the current tension be 1, tension 2 beats are applicable at tension 1 or 2; if this wasn't true, there would be no way to move from tension 1 to 2. Additionally, the precondition requires that at least 2 tension 1 story topic beats have successfully completed. So once these two tension 1 beats have happened, tension 2 story topic beats are sequenceable. The effect of a tension 2 beat is to move the tension to 2; thus, if the tension is currently 1, the tension delta is 1, while if the tension is currently 2, the tension delta is 0. The tension arc will softly determine where the switch from tension 1 to 2 occurs through the influence of story value scoring on the probability of selection. If tension 2 beats are sequenceable before the arc has moved to tension 2, tension 1 beats will have a higher probability of being selected. After the arc has moved to tension 2, tension 2 beats will have a higher probability of selection. Thus, rather than specifying some hard condition under which the switch from tension 1 to tension 2 occurs, the arc provides soft guidance as to where the switch occurs.

Story Topic Singlets

Unlike story topic affinity sets, which are designed to guarantee that a beat that moves the story forward is always sequenceable, story topic singlets are only applicable in specific contexts that often depend on temporal patterns. The *Romantic Proposal* beat described above is an example of this – this beat is applicable when the player has been siding with

Grace for several beats, thus making Trip desperate to look good to the player. In this case the precondition would test whether Grace has had the affinity for three beats, and that the tension is 2. The static priority for this beat is discourseAct – if the precondition is satisfied, this beat has the same priority as story topic beats responding to a player utterance. Weight tests may be used to increase the probability of selecting this beat, such as in the situation that the player has referred to *rocky marriage*. Unlike story topic affinity sets, which all make use of the same structure as the *Decorating* beat above, the details of story topic singlet beats vary, defining different applicability contexts, priorities, and probabilities of selection depending on the details of the content of the beat.

Example Story Traces

This section describes two potential story traces for part I. The first part of this section describes three abstract traces at the level of beat sequencing. At the level of beat sequences, most of the logic responsible for producing the trace occurs in the beat manager. The second part of this section describes detailed, dialog-level traces for portions of two of the abstract traces. At the dialog level, the logic responsible for the trace occurs in the beat goals and handlers (beat behaviors –see Chapter 6 starting on page 105), natural language processing system (natural language understanding and reaction selection – see Chapter 10 starting on page 200), and beat manager.

Spoiler warning: this section reveals detailed information about the story. If you have not yet played *Façade*, reading this section may spoil some of the pleasure and surprise of playing the story.

Abstract Traces

Trace One

In trace one, the player is a woman named Susan.

- ***Player Behind Door*.**
- ***Trip Greets Player*.**
- ***Trip Fetches Grace*.**
- ***Grace Greets Player*.** Currently, the greeting cluster consists of four beats that always happen in this order. Here, the player does nothing to change affinity during the greeting, leaving the affinity neutral as we move out of the greeting cluster.
- ***Decorating* (*artist/advertising*) – Tension 1, Neutral Affinity.** *Decorating*, *Explain Dating Anniversary*, and *Ask About Drinks* all have weight boosts that make them more likely early in the story. Here the drama manager chooses the tension 1, neutral affinity *Decorating* beat. The player changes affinity towards Trip.
- ***Explain Dating Anniversary*.** *Explain Dating Anniversary* is not a triplet; the three possible affinity states are handled within one beat, using alternate dialog when needed. This is possible because, unlike affinity triplets, the *Explain Dating Anniversary* beat only exhibits mild variation as a function of affinity state. In this case, the Trip affinity dialog is used. The player "remembers" that she introduced Trip and Grace, maintaining affinity with Trip.

- ***Ask About Drinks* (*façade*) – Tension 1, Trip Affinity.** Though the player has not asked for a drink, the system proactively sequences this beat as a story topic beat. The player accepts Trip's drink suggestion, maintaining affinity.
- ***Move Wedding Picture* (*artist/advertising*) – Tension 2, Trip Affinity.** At this point, two tension 1 story topic beats have completed, so tension 2 beats become sequenceable (except for *Decorating* and *Ask About Drinks*, which have already been performed at tension 1). At eight beats into tension 1, the desired tension, as specified by the tension arc, is 2. Tension 2 beats get a higher score against the story arc, and thus have a higher probability of being selected. (though the probability of selecting a tension 1 beat is not zero). Additionally, because the player has maintained affinity with Trip for three beats in a row, the singlet *Move Wedding Picture* becomes sequenceable, and is in a higher-priority tier (equivalent to a discourse act reference to the beat topic) because of its static priority. Since this is the only beat in the higher-priority tier, the beat manager sequences it. The player takes Grace's side, causing the affinity to switch to Grace. This beat also causes the tension to move to 2.
- ***Work Project* (*Trip's affairs*) – Tension 2, Grace Affinity.** Since the tension is now 2, only tension 2 beats are available for sequencing; in this case *Work Project* is sequenced. The player takes Grace's side, maintaining Grace affinity.
- ***Mini-climax*.** Since two tension 2 beats have occurred, the part I mini-climax is sequenced.

Trace Two

In trace two, the player is a man named Dave.

- ***Player Behind Door*.**
- ***Trip Greets Player*.**
- ***Trip Fetches Grace*.**
- ***Grace Greets Player*.** Currently, the greeting cluster consists of four beats that always happen in this order. Here, the player does nothing to change affinity during the greeting, leaving the affinity neutral as we move out of the greeting cluster.
- ***Ask About Drinks* – Tension 1, Neutral Affinity.** The player asks for a drink, causing a priority test to raise the priority of *Ask About Drinks*. Since *Ask About Drinks* is the only beat in the higher-priority tier, it is sequenced. The player changes affinity towards Grace.
- ***Decorating* (*artist/advertising*) – Tension 1, Grace Affinity.** *Decorating*, *Explain Dating Anniversary*, and *Ask About Drinks* all have weight boosts that make them more likely early in the story; *Ask About Drinks* has of course already been used and so is unavailable. Here the drama manager chooses the tension 1, Grace affinity, *Decorating* beat. The player causes the affinity to change in the middle of the beat via a global push-too-far mix-in, resulting in the beat aborting. The push-too-far mix-in changes the affinity directly from Grace to Trip.
- ***Decorating* (*artist/advertising*) – Tension 1, Trip Affinity.** An affinity switch priority test pushes the Trip affinity version of *Decorating* into a higher-priority tier. The priority test is satisfied if another member of the affinity set just aborted because of an affinity switch. Since this is the only beat in the higher-priority tier, the beat manager sequences it. The player agrees with Grace, causing the affinity to change towards Grace (to neutral).

- ***Italy Vacation (rocky marriage)* – Tension 1, Neutral Affinity.** At this point, two tension 1 story topic beats have completed, so tension 2 beats become sequenceable (except for *Decorating* and *Ask About Drinks*, which have already been performed at tension 1). The desired story arc is still at tension 1 (it will beat tension 2 after the next beat), so tension 1 beats get a higher score against the story arc than tension 2 beats. The probability of selecting tension 1 beats is thus higher, though the probability of selecting tension 2 beats is *not* zero. In this case, a higher probability, tension 1, *Italy Vacation* beat is sequenced. The player refuses to play Trip's "guessing game" about the vacation photo, causing the affinity to change towards Grace.
- ***Grace's Parents (façade)* – Tension 2, Grace Affinity.** The desired tension is now at 2, resulting in tension 2 beats having a higher probability of being chosen than tension 1 beats. In this case, a higher probability *Grace's Parents* beat is sequenced. The player takes sides with Grace, causing the affinity to remain with Grace. This beat also causes the tension to move to 2.
- ***Trip's affairs (rocky marriage)* – Tension 2, Grace Affinity.** Since the tension is now 2, only tension 2 beats are available for sequencing; in this case *Trip's affairs* is sequenced. The player takes Grace's side, maintaining Grace affinity.
- ***Mini-climax*.** Since two tension 2 beats have occurred, the part I mini-climax is sequenced.

Concrete Traces

The concrete traces provide a dialog-level trace of the two abstract traces above. The concrete traces only progress through the tension 1 beats. Player interactions appear in bold. Commentary on the interaction appears in italics.

Trace One

In trace one, the player is a woman named Susan.

Beat: Player Behind Door

(dialog is heard offscreen and a bit muted, since it is heard through a closed door)
Trip: Where are the new wine glasses?
Grace: What for?
Trip: That should be obvious!
Grace: Oh God, Trip, don't turn this into a big production, please!
Trip: Jesus Grace, come on, I'm not asking a lot here!
(pause)
Grace: What – Trip, don't give me that look!
Trip: (walking away, upset) You're going to drive me crazy!
(*The fight behind the door is a fixed, sequential collection of lines within a single beat goal. There are only two beat-specific handlers, one that responds to a knock on the door, and one that responds to a timeout if the player does nothing.*)
(Susan knocks on the door)
(*The knock results in a handler adding a transition-out beat goal.*)
Trip: Oh, she's here!
Grace: What?! You said she's coming an hour from now!
Trip: No, she's right on time!
Grace: (walking away, upset) Trip...!

Beat: Trip Greets Player

Trip: (opens door) Susan! Hi! It's so…
Susan: Hi Trip.
Trip: … great to see you!
(*The player interrupts Trip in the middle of his line. His line is uninterruptible to the gist point, which is at the end of "…great to see you!" (see discussion of uninterruptible segments on page 114). If the player hadn't interrupted, Trip would have continued with "It's been a while, how's it going?". The natural language processing (NLP) system maps "Hi Trip" into a beat-specific transition-out greeting reaction. The handler for this reaction adds a transition-out beat goal consisting of the following three lines.*)
Trip: Yeah, hi!
Trip: Come on in!
(Susan walks into the apartment)
Trip: (a bit nervous) Uh, let me go get Grace...
(Trip walks into the kitchen)

Beat: Trip Fetches Grace

Trip: Grace … (unintelligible arguing)
Grace: (unintelligible arguing)
Trip: (unintelligable arguing)
Grace: (unintelligable arguing) ... always doing this to me!
Trip: Grace can we do this later, we –
Grace: Yes I'm coming, I'm ready... you just... argh!
(While Trip and Grace argue, Susan wanders around the apartment)
(Grace and Trip emerge smiling from the kitchen doorway)

Beat: Grace Greets Player

Grace: Player! Hi! How are you? God it's been a while!
Trip: Yeah, yeah, how are you doing?
Grace: I just asked her that...
(Susan kisses Trip)
(*The player has just discovered that if you move the cursor over Grace's or Trip's face, it turns into lips. The player tries kissing Trip. Grace and Trip are in the middle of the "quibble" beat goal, during which they argue about asking the player how she's doing. The "quibble" beat goal is interruptible, so the kiss interrupts the goal right away. The beat goal is not done, but the beat goal gist (see page 112) happens at the beginning of the beat goal (at the beginning of "Yeah, yeah, …"); thus this beat goal can be considered completed regardless of when it is interrupted. Global mix-ins are turned off during the greeting beats. The individual greeting beats are responsible for handling all interactions that occur. In this case, the kiss results in the selection of the "kiss" transition-out. A handler adds the appropriate transition-out beat goal.*)
Trip: (confused) Oh, uh, ha ha! You're kissing me! That's hilarious.
Grace: (look at Trip) Oh, uh... ha ha...
Grace: So, please, make yourself at home...
(*Grace's last line depends on whether the player is all the way into the living room. If the player is still standing by the door, Grace says "Well, come in, make yourself at home...". In this case the player has been wandering around the apartment, and so is already all the way into the living room.*)

Beat: Decorating (artist/advertising) – Tension 1, Neutral Affinity

Grace: So, Susan, I'm hoping you can help me understand where I went wrong with my new decorating. (bitter laugh)

Trip: (pacifying tone) Oh, Grace, let's not do that.

Susan: I think it looks great.

(*Beat specific template rules map this utterance to the* praise Grace *discourse act. This beat has a beat-specific mix-in for praising Grace before the beat gist. A handler sequences the appropriate beat goal for this mix-in.*)

Trip: See, I told you she would like it! There's nothing wrong with it.

(Grace gives the player a skeptical look)

(*The Decorating beat can refer to a number of different objects during the performance of the beat. Player interaction can choose the object. For example, one of the objects the beat can make use of is the couch. If, prior to this point, the player had somehow referred to the couch (by mentioning it, sitting on it, or looking at it), the beat would choose the couch as the focus object. In this case, the player hasn't referred to any of the decorating objects, so the beat picks the armoire randomly.*)

Grace: (sigh) You know, when I saw this armoire on the showroom floor, I thought it had such a clean, simple look to it...

Trip: I – I think it's fabulous. I really do.

Susan: It's great.

(*Again, this utterance is mapped to praising Grace. However, the beat-specific before-beat-gist praise mix-in has already been used, so this reaction can't be used again. Instead, a reusable beat-specific deflect is chosen. The deflect is used to respond to any player utterance for which a beat-specific reaction has already been used up.*)

(Grace and Trip look at the player for a moment)

(*After this deflect mix-in, the beat moves onto the next beat goal. However, since the beat flow was interrupted by a mix-in, the beat goal first performs a "reestablish" line, to reestablish the beat flow.*)

Grace: So this armoire looked so appealing when I bought it...

Grace: ...but looking at it here in the apartment, it just looks like... (laugh half-lighthearted, half-bitter) a monstrosity!

Trip: (under-breath, impatient sigh) uhh...

(*At this point the beat-gist has happened. Grace wants the player to agree with her that something is wrong with the room. If the player agrees with Grace, a transition-out is chosen that changes the affinity towards Grace. If the player praises Grace, a transition-out is chosen that changes the affinity towards Trip. If the player doesn't respond, eventually the beat times out, and a transition-out is chosen that doesn't change the affinity.*)

(Grace looks impatiently between the armoire and Susan, Trip fidgets)

Susan: Well, I like your wedding picture.

(*The player agrees with neither Grace nor Trip, but rather brings up another topic. The template rules map this utterance to the discourse act* referto wedding picture. *A global object mix-in reaction is chosen.*)

Trip: (proud) Yeah... you know our wedding picture is the first thing everyone notices when they enter the room.

Grace: (joking sarcastic) Well, geez, how could they miss it?

(pause)

Grace: (casual) You know I don't think it really goes with this room anymore.

Trip: (sigh)

(*After the mix-in, a line is performed to reestablish the beat flow.*)
Grace: (sigh) I'm sure I can return most of this, and try to start over again on this room...
Susan: Grace, it looks nice.
(*This utterance is mapped to* praise Grace*, resulting in the selection of the Trip affinity transition-out.*)
Grace: (annoyed) Uhh!
Trip: (smiling) There, Grace, I try to tell you, everybody loves your decorating!
(pause)
Grace: (frustrated, complacent sigh) Well I'm definitely returning everything in this room, it has to be totally redone.

Explain Dating Anniversary

(*Since Trip has affinity, a Trip affinity version of the transition-in is performed.*)
Trip: Heh heh, Susan, seeing you brings back good memories, you know?
Grace: (small smile, a bit muted) Yeah...
(*The Trip affinity version of the next beat goal is performed.*)
Trip: Whoa I just realized something!
Grace: (a bit suspicious) What...
Trip: Oh, tonight's a special night! A celebration in fact!
Grace: (a bit suspicious) What do you mean?
Susan: Well let's party!
(*The NLP system maps this to a* positive exclamation *discourse act. The beat gist has not happened yet, and there is no beat-specific reaction to* positive exclamation *before the beat gist, so a global, discourse act, mix-in reaction is chosen. A handler adds the mix-in beat goal to the beat.*)
Trip: All right!
(*The next beat goal first performs a line to reestablish the beat context after the mix-in.*)
Trip: So, yeah, I just realized!
Trip: Susan, remember, it was almost exactly ten years ago, tonight, that you introduced us. Senior year of college!
Grace: (annoyed and embarrassed) Oh... geez...
Trip: Remember that?
(*The beat gist has occurred. At this point, if the player agrees with Trip (by "remembering" introducing Trip and Grace), the affinity changes towards Trip. If the player disagrees with Trip (doesn't remember), the affinity changes towards Grace. If the player does not directly respond, the affinity stays the same.*)
Susan: Oh of course.
(*This is mapped to an* agree *discourse act, resulting in the selection of the Trip affinity transition-out. Trip maintains affinity*)
Trip: Ha ha! So we really want to thank you for years and years of...
Grace: (interrupts, half-joking, half-real) Pain.
Trip: (going along with it) Agony.
(pause)
Grace: (acting positive) Love.
Trip: (happy, relieved) Love.

Ask About Drinks – Tension 1, Trip Affinity

(*Since the player didn't directly ask for a drink before this beat, the default transition-in is used, in which Trip brings up the idea of having drinks.*)

Trip: I'm gonna fix us some drinks.
(*In the next beat goal, Trip makes a drink suggestion and brags about his drink making ability.*)
Trip: How does a martini sound?
Trip: I'm a real expert at fixing these, at least that's what everybody tells me.
Grace: We don't need to make a big production out of this, Trip.
(*They wait for five to ten seconds for the player to respond. The beat gist has not happened yet, however. The player's interaction doesn't yet result in the selection of the transition-out.*)
Susan: Sure.
Trip: Beautiful!
Grace: No no, Susan, maybe you'd like some juice, or a mineral water?
Trip: (dismayed, under breath) Oh come on...
(*Now the beat gist has happened. The player's reaction will select the transition-out and affect affinity.*)
Susan: No thanks.
(*The beat considers the player's disagreement to be directed at Grace (the last character who spoke), so the Trip affinity transition-out is chosen.*)
Trip: Okay! Good! I'll just whip up these bad boys real quick! (Trip walks behind the bar, humming)
(*Trip begins pursuing the fix drinks long-term behavior. He'll perform the next beat or so from behind the bar while making drinks. The body resource manager insures that physical actions resulting from the fix drinks behavior mix sensibly with physical actions from beat performance behaviors.*)
Grace: (sigh) (under breath, to Player) Trip thinks he's at his classiest when he's on the serving end of a swizzle stick.

Trace Two

Most beats have alternate lines of dialog, including alternate performances, for the various beat goals comprising beats. Where the same beat goals appear in this trace as in trace one, alternate dialog is used.

In this trace, the player is a man named Dave.

Beat: Player Behind Door

(dialog is heard offscreen and a bit muted, since it is heard through a closed door)
Grace: Trip, when are you going to get rid of this?
Trip: What, Grace... this?
Grace: Yes, you know how I feel about it –
Trip: I know I know I'll do it *right now*, alright?!
Grace: You know I've had to ask you about this several –
Trip: Get off my back! I'll get rid of it in just a minute!
Grace: (walking away, upset) Fine, Trip... fine...
(*The fight behind the door is a fixed, sequential collection of lines within a single beat goal. There are only two beat-specific handlers, one that responds to a knock on the door, and one that responds to a timeout if the player does nothing.*)
(Dave does nothing for 10 seconds)
(*A timeout is generated, causing the timeout handler to terminate the beat with success.*)

Beat: Trip Greets Player

(*A transition-in beat goal for the case of that the player doesn't knock during the previous beat is performed.*)

Trip: (opens door) Dave! Hey! I thought I heard someone out here! Great to see you! It's been a while, how's it going?

Dave: Hi. How are you?

(*The player utterance is recognized as two different discourse acts, a* greet *act and a* miscellaneous HowAreYou *act. The beat-specific reaction proposers propose two different reactions, a transition-out responding to the greet, and a transition out responding to the* HowAreYou*. The* HowAreYou *reaction is chosen because it has a higher reaction priority. A handler adds the appropriate transition-out beat goal to the beat.*)

Trip: Oh we're great. I mean really, really great.

Trip: (enthusiastic) So come on in!

(Dave walks into the apartment)

Trip: (a bit nervous) Uh, I'll – I'll go get Grace...

(Trip walks into the kitchen)

Beat: Trip Fetches Grace

Trip: Grace... (unintelligible arguing)

Grace: (unintelligible arguing)

Trip: (unintelligible arguing)

Dave: What's wrong?

(*During this beat the system ignores all player utterances; Trip and Grace are out of the room and thus can't hear the player.*)

Grace: (unintelligible arguing) ... does this have to happen?

(Dave starts walking towards the kitchen)

(*Player movement into the hallway towards the kitchen triggers a handler that ends the "unintelligible arguing" beat goal and adds a transition-out.*)

(Grace and Trip emerge smiling from the kitchen doorway)

Grace: No, no, here we are!

Beat: Grace Greets Player

Grace: Dave! Hi! How are you? Oh it's so nice to see you, it feels like it's been forever!

Trip: Yeah, it's been too long.

(Grace looks at Trip for a moment, then back at the player)

Grace: And I have to say, you look great.

Dave: Should I go?

(*This utterance is recognized as a* miscellaneous IsOKQuestion *act and results in the selection of an appropriate transition-out.*)

Grace: What? Oh, no, this is perfect, it's so nice to see you!

Grace: (enthusastic) So, good, …

Dave: Can I have some wine?

(*This utterance is recognized as a* miscellaneous AskForDrink *act. The beat ignores this utterance, but leaves the discourse act WME in story memory. The drama manager uses this drink request to choose the Ask About Drinks beat as the next beat. Grace finishes her line.*)

Grace: …make yourself at home!

Ask About Drinks – Tension 1, Neutral Affinity

(*An appropriate transition-in is chosen to respond to the direct drink request.*)
Trip: Dave, that's what I like about you, you get right down to business.
(*Trip wants to make a "fancy" drink – since Trip doesn't consider wine a "fancy" drink, in the next beat goal he proposes an alternative.*)
Trip: (wanting more) Oh, but we can do better than that...!
Trip: Can I interest you in a single malt Scotch? It's primo.
Trip: It's what we drink at these high-class poker games I go to with the execs at work.
(*They wait for five to ten seconds for the player to respond. The beat gist has not happened yet, however. The player's interaction doesn't yet result in the selection of the transition-out.*)
Dave: I hear you're great at poker.
(*This utterance is recognized as a* praise Trip *discourse act. There is no beat-specific response to this praise, so a global mix-in reaction is chosen.*)
Trip: Oh, ha ha, thanks –
Grace: Dave, you're always so sweet to us. Isn't he sweet.
(pause, smiles)
(*Trip and Grace continue waiting for a response to Trip's drink suggestion. After five to ten seconds, a timeout is generated. A handler responds to the timeout by adding an appropriate beat goal to the beat.*)
Trip: Uh, well, I'm just going to make you a Scotch.
Grace: Trip that's not what he wants.
Grace: Dave, you just want what you asked for, right?
(*Now the beat gist has happened. The player's reaction will select the transition-out and affect affinity.*)
Dave: Just wine please.
(*The player agrees with Grace, so the Grace affinity transition-out is chosen.*)
Grace: Okay, Trip, that's what he wants!
Trip: (letdown) Alright, alright!
Trip: (sigh) (grumbling) This reminds me of the time we had your book group friends over.
(Trip starts walking to the bar)
(*Trip begins pursuing the fix drinks long-term behavior. He'll perform the next beat or so from behind the bar while making drinks. The body resource manager insures that physical actions resulting from the fix drinks behavior mix sensibly with physical actions from beat performance behaviors.*)

Decorating (*artist/advertising*) – Tension 1, Grace Affinity

Grace: So, Dave, you can help me understand where I went wrong with my new decorating.
(bitter laugh)
Trip: (pacifying tone) Grace, oh, no, we don't need to do that.
(During these lines, Dave wanders over to look at the painting behind the couch)
(*The player looking at the painting for several seconds results in the generation of a* referto painting *discourse act. The painting is one of the objects the decorating beat can use as a focus object, so the painting is chosen as the beat's focus object.*)
Grace: (sigh) This new painting above the couch is something of an … experiment.
Trip: We just acquired it at an art auction.
Dave: Trip, you have great taste.
(*This is interpreted as a* praise Trip *discourse act. Since there are no beat-specific reactions to praising trip, a global praise mix-in is performed.*)

Trip: (pleased) Oh, ha ha, I –
Grace: Ah you seem to know how much Trip likes it when you praise him.
(pause)
Trip: (sigh)
(*Since a global mix-in interrupted the beat, a special line is performed to re-establish the current beat context.*)
Grace: So yeah, I just bought this painting...
(*And now the current beat goal picks up from where it left off.*)
Grace: ...but, god, (mild disgust) you know, it looks like an amateur painted it.
(*The beat gist has occurred.*)
Trip: (under-breath impatient sigh) well, you're the one who bought it...
Dave: Trip is awesome.
(*The player noticed that the second time he praised Trip, he got a different response. The player decides to see what happens if he keeps on praising Trip. Since this is the third praise in two beats, it results in a push-too-far reaction.*)
Grace: (annoyed) You're really laying it on thick tonight!
Trip: Grace, come on…
Grace: No, no, don't mind me.
(*The push-too-far reaction changes the affinity from Grace to Trip. Since the current beat requires that Grace have the affinity, the affinity change results in the current beat aborting. The drama manager sequences the Trip affinity member of the affinity set.*)

Decorating (*artist/advertising*) – Tension 1, Trip Affinity

(*The beat notes that the previous beat, which aborted because of an affinity change, is a member of the same affinity set. Since the beat gist of the previous beat has already occurred, this beat skips the transition-in and body beat goals, going straight to the wait-timeout beat goal; the wait-timeout beat goal waits for a player response to the beat gist, timing out if the player says nothing. First, a special line is performed to re-establish the beat context.*)
Grace: (sigh) I bet I can return most of this, and start over again on this room...
(Grace looks impatiently between the painting and Dave. Trip brings the player a glass of wine.)
(*The long-term fix-drink behavior finishes with Trip bringing the wine to the player.*)
Dave: Yeah, this room is ugly.
(*Grace gets the desired criticism, resulting in the Grace affinity transition out. Affinity moves towards Grace, becoming neutral.*)
Grace: Oh, how nice, I've been waiting so long for someone to say that!
Trip: (off-guard) Wait, what?
Grace: Trip, our friend just been refreshingly honest about my decorating, which I really appreciate.
Trip: (disconcerted, to Dave) Huh, I wouldn't have guessed you'd say that.
Grace: Trip, if you can't see it, don't worry about it. You've never had an eye for these things.
Trip: I – (sighs frustrated) uhh.

Italy Vacation (*rocky marriage*) – Tension 1, Neutral Affinity.

(*The player didn't directly refer to the vacation photo (either by talking about it or looking at it), so the default transition-in is used.*)
Trip: Oh, Dave, take a look at this photo I just put up from our trip to Italy a couple of weeks ago…
Grace: (sigh)

(*The next beat goal attempts to lure the player over to the photo. It eventually times out if the player doesn't go to the photo.*)
Trip: (to Player, gesturing at picture) C'mere, I want you to look at this!
Grace: (looks away)
(Dave walks over to photograph)
(*The player's movement to the photograph completes the previous beat goal. The rest of the beat goal (consisting of additional physical performance while Trip and Grace wait for the player to go to the photograph or not) is bypassed. The next beat goal establishes the guessing game.*)
(Grace starts walking towards the couch)
Grace: Dave, come over here and sit on the couch with me.
Trip: No, hold on. (gestures at picture) Now, Dave, what does this picture make you think of?
Grace: Oh, Trip, don't put our friend on the spot like that...
Trip: In a *word*, what does this *say* to you?
Dave: Trip, you're bizarre.
(*This utterance is mapped to a* criticize Trip *discourse act. There is no beat-specific response to this act, so a global mix-in is performed.*)
Trip: (puzzled) Wha... uh...
Grace: Oh, Trip, he's just teasing you... It's good – we all need that once in a while.
(pause)
Trip: (trying to act relieved) Ha ha... (pause, recover)
(*A line is performed to re-establish the beat context.*)
Trip: (gesturing at photograph) So in word, what does this say?
Dave: Trip, what are you talking about?
(*Once the guessing-game beat goal is active, all player input is interpreted as either multiple words (resulting in the response below), a "correct" single word, or an "incorrect" single word. This is accomplished by activating special high-priority template rules and reaction proposers at the beginning of the guessing game beat goal.*)
Trip: No, I just want one word...
Grace: How about "tiresome".
Trip: Grace, please. (to player) Just one word for this.
(Dave walks over to Grace)
(*A handler triggers on Dave leaving the vicinity of the photograph, terminating the guessing game beat goal, and selecting a transition-out that changes the affinity towards Grace. The details of the transition-out condition on whether, when the player leaves the vicinity of the photograph and thus terminates the guessing game, the player moves to the vicinity of Grace, or to some other part of the room. In this case, the player has physically moved near Grace.*)
Grace: Trip, darling, your obsession with that photo of yours is making our friend uncomfortable.
(brief pause)
Trip: (to himself) Romance... it says... romance...
Grace: (looking at picture, under-breath, to player): Dave, about Italy... you know, all the buildings there were just so old and crumbling... it was like everything there was falling apart... uhh.
Grace: But Trip says, "that's what makes it beautiful"... (sigh)

Related Work

Interactive Story

In reviewing the literature on interactive story systems, I focus here on systems that explicitly employ AI techniques to maintain/generate a narrative arc in response to interaction.

Laurel [Laurel 1986] provides a seminal description of a dramatic virtual world in which an AI system is used to structure interaction as a dramatic arc. While she did not implement a system, she did describe the dramatic requirements for interactive drama and explored what an architecture for such a system might look like.

Weyhrauch [Weyhrauch 1997] developed a game-tree search technique for drama management in an interactive world. This is arguably the first implemented system that actively maintains a dramatic arc in an interactive context. His system controls a story at the level of plot points, or major scenes in a story. The system has available to it a collection of system "moves" that, by manipulating characters and the physical state of the world, modify the probability of certain plot points appearing in the future. Whenever the system detects that a plot point transition occurs (the player has done something in the story world to make a plot point happen), it projects all future combinations of system moves and plot point occurrences (to the end of the story). Each total projected story (past plot point sequence plus future projections) is evaluated by an author-written function that computes the "goodness" of the total story. The system then makes the system move that maximizes this story "goodness". Weyhrauch's system is intended for the sequencing of plot points, relatively large granularity story units that often correspond to scenes. Architectural components called recognizers (not implemented in Weyhrauch's dissertation) are responsible for guiding activity *within* the plot point. *Façade* tightly integrates character-level and story-level control in a system that engages in frequent sequencing of small-granularity story units (beats). Within the Oz framework, the beat manager could be considered an implementation of refiners, the mechanism responsible for the more moment-by-moment mangement of activity within a scene.

Script-and-demon story systems combine the use of a script to specify linear or branching sequences of events with demons that are associated with events. The demons won't let an event happen until certain preconditions on the state of the world have been satisfied. Plot graphs [Kelso, Weyhrauch & Bates 1993], an early approach to drama in the Oz project, are one example of such a system. A plot graph lays out scenes in a directed acyclic graph (DAG). The arcs represent the must-precede relationship. Only after all preceding plot points have happened can the next plot point be entered. Associated with the arcs are hints and obstacles. These are ways that the drama manager can influence the world. Hints make it more likely that the user will move into the next scene; obstacles slow the user down. Demons recognize when a user has completed a scene. Another example, Pinhanez's *Interval Scripts* [Pinhanez 1997], represents the script by using a temporal calculus to represent temporal relationships among intervals. Some of these intervals are connected to sensors (demons) that wait for events to occur in the world; others are connected to actuators that make events happen in the world. A constraint propagation mechanism is used to determine the state of each interval (now, past, future, or some mixed state). When a sensor has the value `now`, it begins looking for its associated event to happen in the world. When an actuator has the value `now`, it makes its associated event happen in the world. In Galyean's *Dogmatix* [Galyean 1995], an analogy is made between the action selection problem in behavioral agents and the event

selection problem in plot control. At each point in time, a behavioral agent must select one (or in general, some small subset) behavior from its pool of possible behaviors. This selection is accomplished as a function of the internal state of the agent and the external state of the world. Analogously, at each point in time a plot selection mechanism must select an event to make happen out of the set of all events it could make happen. In Galyean's system, this selection is a function of story state variables (history), sensors (demons watching for events in the world), and *temporal relationships*. The temporal relations hierarchy, before, xor, and must-happen place a partial order on the possible sequences of events chosen by the selection mechanism. At each point in time, the event that has the highest "fitness" is chosen for execution.

The *DEFACTO* interactive story environment makes use of a rule-based system for story guidance [Sgouros 1999]. Generation rules encode knowledge about role related norms, social actions, and personal goals and relations. Based on the role and goal specifications of the cast of characters (including the player), the generation step generates possible character actions. For the player character, possible actions are presented in a menu. The evaluation step makes use of dramatic significance rules to select specific actions for the non-player characters from among the set of possible actions computed by the generate step. The cycle of generation and evaluation continues until no dramatically interesting situations are found. At this point the resolution rules compute the final outcome of the character interactions to complete the story.

Carmen's Bright IDEAS [Marsella 2000; Marsella, Johnson & LaBore 2000] is an interactive pedagogical drama designed to teach mothers of pediatric cancer patients a specific social problem solving technique (the Bright IDEAS method). The system uses a presentational rather than an immersive style of interaction. That is, rather than playing a character *within* the drama, the player influences the unfolding drama by manipulating the intentional states of a semi-autonomous character (the mother). The autonomous character of the therapist is also the story director. Besides making her own character level decisions, she also directly influences the unfolding story by changing the state of the mother through dialog and through privileged access to the mother's emotional state.

The *Erasmatron* [Crawford 2002] is an authoring system designed to allow writers with a non-technical background to write interactive stories. Characters are represented through collections of verbs (actions) that the character can execute. Action choices are made based on the current state of the world and the state of internal traits within an elaborate character trait system. The player selects actions from changing menus of available actions. There is no explicit representation or control over the story arc; the story arc is implicitly determined by the character actions and the actions made available to the player. Like most adventure games (whether text or graphical), the *Erasmatron* uses a discrete time model, in which the player is offered distinct interaction opportunities and in which both player and character actions execute instantaneously.

The *Mimesis* architecture [Young 2001] constructs story plans for real-time virtual worlds, currently the *Unreal Tournament* world. The story plans generated by the planner are annotated with a rich causal structure. The system monitors for player actions that might threaten causal links in the current story plan. If a threat is detected, the system either generates a new plan that accommodates the player action while still accomplishing the story objectives, or intervenes by causing the player action to fail and thus protect the threatened causal link.

The Mission Rehearsal Exercise Project [Swartout et. al. 2001] is building a wide-screen, surround-sound, immersive training environment for military peacekeeping operations. The narrative training scenarios make use of a combination of scripted and

autonomous chararacters. Story structure is managed by the story net, a structure similar to the Oz plot graphs described above. Nodes define contexts within which the player is free to act. To progress the story, conditions associated with the arcs of the story net must be satisfied in order to traverse the story to another node. Node traversals (movement along an arc) are communicated through non-interactive cut scenes.

Story Generation

AI work in story generation has focused on building architectures that non-interactively generate stories. The stories are most often expressed as narrative prose or in some high-level natural-language-like formalism. Each story generation architecture makes a different set of commitments regarding the fundamental units out of which stories are composed, and provides a different set of answers to the questions "What are the authorable units?", "How can they be combined?", and "What knowledge guides and constrains their combination?" For this reason, story generation provides a fertile ground for thinking about architectural commitments in any story-based system, including interactive drama.

Tale-Spin [Meehan 1976] generates stories in the domain of Aesop's Fables. *Tale-Spin* models the goals and problem solving processes of a set of characters. The stories generated essentially consist of the problem-solving traces of selected characters given initial goals. The major insight of *Tale-Spin* is that the problem-solving traces of a set of characters doesn't necessarily make a story. Additional knowledge about what constitutes a good story is needed.

Around the same time as *Tale-Spin*, people were exploring an alternate approach to story generation based on story grammars [Colby 1973; Rumelhart 1975]. Story grammars encode story structure for a given domain using formal grammar rewrite rules. However, it turns out that syntactic rewrite rules are not enough; the rules must be given semantic (extra-structural) annotations in order to appropriately constrain story recognition and generation. Modern work on story grammars attempts to better understand the relationship between syntactic and semantic knowledge within the grammar [Lang 1999; Lang 2002].

Universe [Lebowitz 1984; Lebowitz 1985] generates soap-operas, entangling its characters in endless episodic action. Characters are represented as collections of traits. Hierarchical story plans directly coordinate characters in story situations. While these plans have no notion of execution or run-time interaction, they are similar to beats in that they directly coordinate multiple characters.

Minstrel [Turner 1991; Turner 1994] is a case-based model of the human authoring process, writing stories in the King Arthur domain. The system solves a series of authorial goals (e.g. "Make Lancelot run into Guinevere in the forest") by searching for "similar" situations and analogically mapping them to the current situation.

Ani [Kahn 1979] is the only story generation system I know of that creates animated output. Given a formal story description from the Cinderella domain, *Ani* uses a constraint satisfaction process to design a 2D animation (characters are represented as geometrical shapes) that conveys the story.

Bailey [Bailey 1999] describes an in-progress system that generates stories by modeling the reader's response to the evolving story. Bailey also provides a nice categorization of story generation systems into world, author and story modeling systems, and contrasts his approach to reader modeling with each of these categories.

Brutus [Bringsjord & Ferrucci 2000] generates stories of betrayal. Thematic knowledge is used to guide the generation of a plot via forward simulation of characters

in a simulated world (ala *Tale-Spin*). Characters are represented by beliefs, goals, plans, and reactive behaviors (production rules). The sequence resulting from forward simulation is turned into a plot outline using story grammars. The plot outline is then turned into natural language using a grammar augmented with knowledge of literary effects.

Future Work

In non-AI-based interactive stories, the interactive story structure is often captured as a hand-authored story graph, where nodes represent story units (e.g. scenes or beats), and links represent the possible player-mediated transitions between story units. Drama managers replace this hand-authored graph with a story policy; the policy is responsible for selecting the next story unit to sequence as a function of the current story state, including features of the already-sequenced portion of the story. In a sense, this *virtualizes* the links of the story graph. From a god's eye view one can still imagine a graph capturing all possible sequences the drama manager can produce, but this graph is so complicated as to be intractable for the author to directly specify. By defining a story policy, the author can specify interactive stories of a complexity that would be impossible to specify as a story graph.

Drama management virtualizes the links (connections between story units), but still requires the author to hand-specify all the story units. The drama manager still requires a rich collection of story units in order to manage the large-scale structure of the story. For *Façade*, this means that the beat manager must have a rich collection of beats at its disposal. The more beats that are available for different situations, the more fine-grained control the beat manager can provide. In *Façade*, we found that the beat management bottleneck was not in the sophistication of the beat management logic, but rather in authoring enough beats to give the beat manager something to do. The complexity and richness of the virtual story graph (the god's eye view) depends not just on the link complexity, but also on the total number of nodes (story units). To move to the next level of interactive story richness and complexity, not only the links should be virtual, *but also the nodes*. That is, the story units (e.g. beats) should be generated on demand as they are needed by the drama manager. The story policy, rather than directly selecting a story unit as a function of the particular story situation, instead would specify the features of the desired story unit; the unit itself would be constructed out of smaller pieces so as to satisfy the specified features.

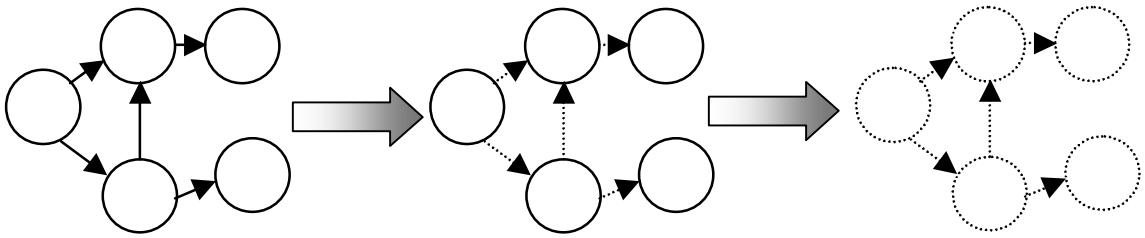

Figure 8-13. Explicit story graphs vs. drama management vs. generative drama

Future work should focus not so much on drama management (the logic for sequencing pre-written pieces), but rather on story generation. The interesting twist here is that, unlike previous work in story generation, rather than an entire story being generated non-interactively, here small pieces of the story must be generated on-demand as required by the drama manager. The trick of course is to maintain the potential for rich authorial control, providing multiple levels at which the author can customize the story unit generation process. One possible approach to explore is case-based generation techniques, in which the author specifies a small number of concrete story units, which are each annotated with symbolic knowledge describing the unit, as well as an adaptation process that modifies the pre-written story units as needed by the drama manager. The hope here is that by seeding the generation process with lovingly hand-crafted story units, the author can still maintain rich control over the details of the story performance while allowing the architecture to shoulder some of the combinatoric burden of authoring the entire collection of story units.

While exploring more generative story systems is to my mind the most important next drama management avenue to pursue, another interesting avenue to explore is adding hierarchical decision processes to the drama manger. In *Façade*, the drama manager is flat – beat sequencing is the only story-level decision process. A hierarchical drama manager could factor considerations of different aspects of the global story structure into different decision processes. One simple such factoring would be to place considerations at different story scales into different processes. For example, a forward search-based drama manager such as Weyhrauch's [Weyhrauch 1997] could be used to manage large scale story structure (e.g. scenes); the sequencing of the larger granularity, more abstract story unit results in the activation of a collection of beats and story value arcs to be used by the beat manager to accomplish that more abstract story unit. But scale-based factoring is only one such possibility. Other factorings may divide story reasoning into different representations and decision processes, not as a function of scale, but because different aspects of the story are most efficaciously represented by different representations (this is an issue of authorial affordance – see Chapter 7). For example, in *Façade*, a rule-based system might be used to capture the rules relating actions such as breaking trust or bonding with the player, character traits such as confidence, and character decisions such as staying in the marriage or leaving. This knowledge may be more naturally expressed as rules rather than spread across the various beat tests. The conclusions reached by this rule-based layer could then be used to modulate the beat manager.

CHAPTER 9
EXPRESSIVE AI: DISCIPLINARY ISSUES

Introduction

Chapter 4 introduced Expressive AI, discussing the differences between art practice and AI research and arguing that an AI-based art practice should not privilege specific technical disciplines within AI. Chapter 7 used the concepts of interpretive and authorial affordance to describe how Expressive AI can simultaneously focus on internal system structure and audience experience. This chapter completes the discussion of Expressive AI by examining the technical, conceptual and aesthetic goals of this practice, and exploring disciplinary issues involved in combining AI research and art.

Why Use AI in Cultural Production?

Up to this point the practice of Expressive AI has been described as one combining both a focus on meaning-making and the authorial affordances of AI architectures. However, this begs the question of why an artist would want to use AI in cultural production at all. Different digital artists focus on many different aspects of computational media, including networked communications, storage and retrieval of large amounts of information in databases, and image generation or manipulation. This section describes why I have singled out AI as a particularly interesting technical tradition to utilize within an art practice.

For Expressive AI, the essence of the computer as a representational medium is not the ability to intervene in the production of three dimensional forms or visual imagery, nor the ability to interact with a participant/observer, nor the ability to control electro-mechanical systems, nor the ability to mediate signals sent from distant locations, but rather computation, that is, processes of mechanical manipulation to which observers can ascribe meaning.

Deep Interaction on Human Terms

AI-based artwork operates in human contexts, deeply participating in the flow of meaning within these contexts in way not possible without AI techniques. For example, text learning methods provide *Office Plant #1* (Appendix B) with a window onto the human interpretation of email, enabling it to react to the social and emotional content of email. Thus *Office Plant #1* is responsive to some of the meanings flowing through an office context, providing an alien commentary on the slice of an office worker's day that is visible through their email activity. Contrast this with a visualization of network activity below the threshold of human meaning. For example, *Office Plant #1* could have been built to respond purely to the number of packets per unit time (packet density) flowing on a user's network. When the packet density is below some threshold, the plant would be in the "rest" position, moving to the "bud" position when the packet density is medium, and the "bloom" position when the packet density is high. Network activity is only roughly correlated with the activities of an office worker. Packet density may

increase because of automated processes, such as a network backup of the hard drive. Even when the packet density increases because of meaningful action, such as web surfing, many of these packets may actually be associated with animated advertisements or popup windows, which are actually not meaningful or significant to the user. Packet density is below the threshold of human meaning – it is the *content* of the packets, perhaps comprising an email from a friend, or an mp3 of a new song from a favorite group, that are meaningful. By using AI techniques, specifically statistical text classification, *Office Plant #1* is able to actively interpret email activity in a manor consonant with human interpretation, and thus to actively participate in (a portion of) the flow of meaning in an office context.

An interactive artwork (system) exists in a shared context with human participants. As the system responds to this shared context, the human participants interpret the system as in some sense understanding the shared situation and acting on this understanding. This notion is similar to the conversational model of interactivity proposed by Crawford [Crawford 1992; Crawford 1993] and applied to interactive art by Stern [Stern 2001]. In this model, interactive art *listens* to the audience, *thinks* about what it heard, and *speaks* its thoughts back to the audience[29]. To say that an artwork is interactive is to say that the work and the audience converse with each other, in a cycle in which each listens, thinks and speaks. AI methods enable deeper conversations through sophisticated mechanisms for "listening" and "thinking". In the course of this conversation, the interpretive and generative mechanisms of an AI-based artwork support the ascription of subject status to the work. For example, the various architectural mechanisms of *Façade* all work together to allow the human player to ascribe subject status to Trip and Grace, to see them as responsive and alive.

However, to say that an AI-based artwork supports the ascription of subject status is *not* to assert the simple identity that an AI-based work is a human subject. An AI-based work is a new kind of subject, inhabiting the liminal space, the constantly negotiated boundary, between living and dead, between intention and brute causality. As Hayles argues in [Hayles 1999], the disciplines of cybernetics, AI and artificial life have dissolved any sense of a simple, hard, clean boundary between the human subject and the non-human world. Instead, these once separate realms interpenetrate, their shifting boundary contingent and mediated. AI-based works function at this uncertain boundary. These alien subjects may not be explicitly represented as human. *Office Plant #1* engages in its own simple email meditations, connected to the human world but not human. *Terminal Time* (Appendix C) is an uncontrollable genie, taking the audience responses and running amok with its own mad logic, constructing not the history that the audience desires, but an off-kilter history consistent with their interaction. Even when, as in *Façade* or *Subjective Avatars* (Appendix A), human representations are employed, the AI system still has an alien subject status – it is not a human being, but rather a *character*. One of the principle innovations of the Oz project was to introduce the idea of believability into AI discourse [Bates 1994], changing the focus from imitating the behavior of a "generic human", to constructing rich and compelling presentations of behavior that foster the willing suspension of disbelief.

Tap into Rich Practice of Doubled Machines

Chapter 7 discusses the idea of the doubled system, that every AI system is composed of both a code machine and a paired, entangled, rhetorical machine. The rhetorical machine

[29] The word "conversation" isn't limited to literal speech, but includes any meaningful exchange.

embeds the technological system in broader systems of meaning, narrating the operation of the code machine and defining a notion of progress, a collection of incremental technical constructions to try next. The discipline of AI has a rich history of constructing these doubled machines, in the process borrowing language and concepts from a wide array of fields. For the AI-based artist, this rich practice of technical and rhetorical construction serves as a conceptual resource, supporting the artist in entangling computation with human meaning.

Exploring the Human by Making

The field of AI is the modern incarnation of an age old quest or dream, the dream of building an image of the human in the machine. It is this dream, fueled by science fiction representations of AI such as Hal 9000 or Commander Data, that is the initial inspiration for many researchers entering the field. This dream is not just about modeling rational problem solvers, but about building machines that in some sense engage us socially, have emotions and desires, and, through our interactions with them, tell us something about ourselves. AI is a way of exploring what it means to be human by *building systems*. An AI architecture is a machine to think with, a concrete theory and representation of some aspect of the human world. Art also explores what it means to be human by *building concrete representations* of some aspect of the human world. Artists often explore aspects of humanity that have been under-explored or ignored in AI research.

Both AI and art are ways of knowing-by-making. In AI, writing programs becomes a way of working through a concept and developing new ideas. Schank, in his description of the research agenda at Yale, writes:

> Thus, for us, theory creation is a process of thought, followed by programming, then by additional thought, with each serving the other. Thus AI really operated under a novel view of science. Normal scientific method holds that first a theory is postulated, and then tested and found to be right or wrong. But in AI our theories are never that complete, because the processes we are theorizing about are so complex. Thus our tests are never completely decisive. We build programs that show us what to concentrate on in building the next program. [Schank & Reisbeck 1981: 4]

Artists work through conceptual and aesthetic issues through the practice of building concrete representations. Combining these two ways of knowing-by-making opens up a new practice that takes seriously the problem of building robust intelligences that function outside of the lab, engaging human participants in intellectually and aesthetically satisfying interactions which, hopefully, teach us something about ourselves.

Build Microworlds with Human Significance

Building microworlds was an AI approach popular in the 1970s. The idea was to build simple, constrained, artificial worlds in which an AI system could exhibit its competence. It was hoped that it would be possible to slowly scale up from systems that exhibit competence in a microworld, to systems exhibiting competence in the real world. The microworld research agenda has been widely criticized (e.g. [Dreyfus 1999]); it did not prove possible to scale systems up from microworlds. However, the microworld concept can be useful in Expressive AI. An AI-based art piece may be a microworld with human significance. The “micro” nature of the world makes certain AI techniques tractable. As long as the microworld has some cultural interest, the system still functions as an artwork. This is simply the recognition that an artwork is not the “real world”, but is

rather a representational space crafted out of the world. The AI techniques used in an artwork only have to function within the specific artistic context defined by the piece. For example, in *Subjective Avatars* and *Façade*, the agents only have to operate within the specific dramatic context defined by the storyworld.

Videogames

Videogames are a booming entertainment form, with sales for videogames hitting $6.35 billion in 2001[30]. As high resolution animation and physics modeling become standard, AI is becoming an important differentiator between games, with game creators often touting their AI in their marketing campaigns. Some forms of AI-based interactive art are already similar to computer games. Interactive drama is related to the already established form of the adventure game, though it differs in its focus on the first-person experience of a dramatic arc rather than goal-based puzzle solving. Other systems, such as *Office Plant #1*, share a focus on long-term engagement with virtual pets such as *Dogz* and *Catz* [Stern, Frank & Resner 1998], though virtual pets are intended for circumscribed, high-intensity interaction, while *Office Plant #1* provides continuous, ambient commentary. These similarities hint at future hybridizations of AI-based art and videogames, becoming, like cinema, a popular art form, experienced not in gallery and museum spaces, but rather in arcades, bars, restaurants, and homes.

Forms of AI-based Artwork

In this section I briefly describe some of the modes or genres of AI-based art. This list is not intended to be exhaustive, but rather to suggest the possibilities of AI-based art production. The modes described here are not mutually exclusive; a single piece may simultaneously explore multiple modes or genres.

Procedural Portraits of Human Meaning-Making

A procedural portrait is a dynamic representation of some psychological, social or cultural process. For example, *Terminal Time* is a procedural portrait of the ideologically-biased construction of mainstream historical documentaries. While the constructed documentary, and the relationship between this documentary and the audience input, are the directly sensed portion of the audience experience, the mechanisms by which the system constructs documentaries are also part of the piece. *Terminal Time's* architecture is a caricature of totalizing thought, of the mono-maniacal pursuit of an ideological position. How *Terminal Time* constructs biased histories is as important a part of the conceptual exploration of the piece as are the histories that are actually constructed.

Harold Cohen's *Aaron* [McCorduck 1991] is another example of procedural portraiture. Cohen abandoned his painting career in the late 1960's to spend the rest of his life working on *Aaron*, a painting program that autonomously generates paintings in Cohen's style. But Cohen was not motivated by the end result, by the idea of a painting produced by a machine, but rather by the process of painting. Writing *Aaron* was a prolonged meditation on his own mental processes when painting, an attempt to understand his own image-making behavior.

[30] http://www.idsa.com/2001SalesData.html

> What the computer provided was a way of externalizing, stabilizing my speculations about image-making behavior: not only my own behavior, but what I thought I could see operating in drawings generally, and especially in children's drawings and so-called primitive art. [McCorduck 1991: 78]

The notion of procedural portraiture is similar to longstanding research approaches in AI that build models of psychological processes, such as models of human problem solving behavior when faced with laboratory tasks. But in traditional AI research these models are taken as *simulations*, possessing a one-to-one correspondence with the system being simulated and thus, in a functionalist framework, *identical* to the system being simulated. But simulation is just one of many possible relationships that can exist between a representation and the world. Representations can simplify, embellish, highlight, subvert, satirize – procedural portraits do not make claims of identity with that which they represent, but are rather thought experiments, dynamic representations of some aspect of human activity.

Characters

AI-based characters are dynamic portrayals of humans or animals, with the focus not on realism but believability. Two of the earliest (and best known) AI-based characters are the *Doctor* script of *Eliza* [Weizenbaum 1966], and *Parry* [Colby 1975]. Both *Doctor* and *Parry* are chatterbots, programs that process textual natural language input and produce text output. *Doctor* used simple pattern matching techniques to respond to user input in the guise of a non-directive Rogerian psychotherapist. *Parry* portrayed a paranoid schizophrenic. Technically, *Parry* was more advanced than *Doctor* in that it maintained internal state beyond individual player utterances, pursuing its own delusions as well as maintaining a simple emotional state. Both programs demonstrated that relatively simple techniques can support the suspension of disbelief, allowing users to participate in the illusion.

Doctor and *Parry* are in some sense the first believable agents. Contemporary work on believable agents tends to focus on real-time, animated representations. For example, Naoko Tosa has built a series of screen-based characters that respond to the participant's emotional state. One of her best-known characters is *Neuro Baby* [Tosa 1993]. *Neuro Baby* responds to tone of voice, becoming upset if addressed in a harsh or angry tone of voice, and happy if addressed in a soothing tone of voice. Additional work on screen-based believable agents has been surveyed on page 100.

Alien Presence

Characters leverage viewer expectations of human and animal forms and behaviors in order to achieve believability, making use of known character types from other media representations, such as movies, animated films or comics. Alien presences, like characters, are dynamic depictions of living creatures. But unlike characters, alien presences have novel forms and/or behavioral repertoires. While perhaps borrowing bits of form and behavior from the animal or human world, they primarily have their own perceptions, reactions, concerns and thoughts, divorced from the everyday human world. For the participant, the pleasure of interacting with an alien presence is to enter its alien world, to discover how it perceives and acts. *Office Plant #1* is an example of an alien presence, offering an ambient commentary on its owner's email stream.

One of the earliest examples is Gordon Pask's *Colloquy of Mobiles*, presented at the Cybernetic Serendipity show in 1968 [Pask 1971]. The mobiles form a simple social

system. Communication takes place by modulating colored lights and audible tones. The mobiles attempt to maintain a "territory" – this results in a combination of competition and cooperation between the mobiles. Left to their own devices, the mobiles pursue their own goals, rotating, flashing lights and tones at each other, and exhibiting various forms of social organization. Audience members entering the environment were provided with devices to generate the lights and tones of the mobile's language, and were thus able to participate in the alien society.

Simon Penny's *Petit Mal* [Penny 1997] is an autonomous robotic being that interacts in physical space with audience members. Penny explicitly sought to avoid anthropomorphism, biomorphism, or zoomorphism, wanting instead "… to present the viewer with a phenomenon which was clearly sentient, while also being itself, a machine, not masquerading as a dog or president" [Penny 2000]. *Petit Mal's* body consists of a double pendulum system that dynamically stabilizes its two-wheeled stance. Audience members project rich and complex readings of *Petit Mal's* internal life – much of the complexity of its "personality" arises from the physical dynamics of its alien form.

David Rokeby's *Giver of Names* [Huhtamo 1998] is a computer-vision based work that offers poetic commentary on the objects placed in front of its camera. The audience is provided with a collection of objects that can be placed on a podium in front of the camera. The colors, forms and juxtapositions of the objects stimulate nodes in a semantic network; the poetic commentary results from the activated concepts in the semantic net. After listening to the commentary for awhile, audience members begin to discern the private, idiosyncratic structure of *Giver of Names'* conceptual space.

Narrative

Story-telling and narrative are fundamental to human experience [Mateas & Sengers 2002]. As children, we are immersed in stories and learn to approach the world via narrative frameworks. As adults, we order events and find meaning by assimilating them to more-or-less familiar narratives. AI-based narrative art explores narrative meanings by generating, modeling and understanding narrative structures and meanings.

Façade, with its incorporation of autonomous character and player activity into a story framework, and *Terminal Time*, with its generation of ideologically biased documentary histories, are examples of AI-based narrative art. Chapter 8 includes a brief survey of AI-based interactive narrative and story generation work on page 181. The literature on story generation offers a useful collection of ideas and technical approaches for AI-based narrative art.

Robotic Art

Robotic art is concerned with building physical, often sculptural, systems whose behavioral responses are a function of the system's perception of the environment. Robotic art is concerned with the aesthetics of physical behavior. Jack Burnham described the possibility of an aesthetics based on cybernetics and artificial intelligence in 1968 [Burnham 1968; Penny 1999].

The Senster, mentioned briefly on page 1, is an early example of robotic art. Inahtowicz was interested in how humans ascribe intention to physical motion and behavior:

> This is not the place to present the general argument that all perception is dependent in some way on an interpretation of physical movement, but all the pieces I have made so far and all that I am planning to make aim ultimately at making the

> spectator aware of just how refined our appreciation of motion is and how precisely we are capable of interpreting the intention behind even the simplest motion. For an artificial system to display a similar sense of purpose, it is necessary for it to have a means of observing and interpreting the state of its environment. [Ihnatowicz 1986: 4-5]

Robotic art often explores the responsive behavior of abstract sculptural forms that have little or no relationship to human or animal forms. For this reason, robotic art is often an alien presence, an abstract responsive "life form" with its own idiosyncratic behaviors and drives appropriate to its physical form.

Meta-Art

Meta-art systems produce art as output; that is, they autonomously create writing, visual imagery, music, etc., that might be considered art in its own right. The goal of meta-art is often conceptual, an exploration of the dynamics of creativity within a specific genre; the generated output itself is often not the primary goal.

One body of work has explored the generation of writing, such as poetry or stories. Masterson's haiku generator [Masterson 1971] is an early example of poetry generation. More recent examples include Loss Glazier's sound poetry generator[31] (sound poetry is interested purely in the sound of often nonsense words, rather than in semantic meaning), Kurzweil's *Cybernetic Poet*, which learns language models from example poems and generates poetry in that style[32], and Manurung's work, which generates poetry using a natural language generation framework that makes use of both syntactic and semantic knowledge [Manurung, Ritchie & Thompson 2000]. Additional story generation work is surveyed on page 183.

There is a vast body of work in music generation, a survey of which falls far outside the scope of this thesis. Notable examples include David Cope's *EMI* (Experiments in Musical Intelligence), a composition system that analyzes musical examples in terms of their stylistic components and recombines these components to generate new music in the same style [Cope 1996], and George Lewis' *Voyager*, an improvisation system that performs with human musicians [Roads 1985].

There is a similarly vast body of work on systems that generate drawings or paintings. *Aaron* [McCorduck 1991], described above, is a prominent example. Visual grammars have been developed to describe the painting style of painters such as Wassily Kandinsky, Joan Miro, Juan Gris and Richard Diebenkorn [Lauzzana & Pocock-Williams 1988; Kirsch & Kirsch 1988]; programs implementing these visual grammars are then able to generate endless variations on these styles.

Expressive AI as Discipline

So far this chapter has described why AI is a particularly interesting technical tradition to combine with an art practice, and offered a brief survey of some of the genres or modes of AI-based art. The last section of this chapter describes Expressive AI as a critical technical practice, a research and art practice that simultaneously makes technical contributions within AI while engaging in critical reflection on the foundations of the discipline.

[31] http://epc.buffalo.edu/authors/glazier/java/iowa/
[32] http://www.kurzweilcyberart.com/poetry/rkcp_overview.php3

Critiques of AI

A number of critics have argued that AI is fundamentally impossible, or, if possible, immoral. Here I position Expressive AI relative to a few of these critiques.

Dreyfus [Dreyfus 1999] provides a critique of the strong symbolic AI position – the claim that general human intelligence can be produced in a machine through the formal manipulation of symbolic structures. He is concerned with the question "What can computers do?", and argues that computers (at least computers making use of formal symbol manipulation) can never achieve general human intelligence. The central point of his argument is the phenomenological insight that human intelligence occurs against a vast, unarticulated background of assumptions about the current social and physical situation. The shifting interpretation of sensory data and the selection of action takes place against this background. The nature of symbolic representation requires that this open-ended background be pre-analyzed into a finite set of relevant features, relationships and contexts. But the power of human intelligence comes precisely from the ability to determine relevance on the fly, to flexibly determine the important features of the context as a function of the current task and situation, and to do this without needing to enumerate some large (or infinite) set of all possibly relevant features. That is, humans can *generate* relevance within a context, whereas symbolic AI programs must be handed a pre-digested representation in which all potentially relevant features of the context have been enumerated by the programmer, along with the appropriate rules for selecting which features to attend to for different tasks. Therefore, by this argument, the entire project of strong symbolic AI is impossible. Dreyfus illustrates his arguments by describing a number of systems in the history of symbolic AI, the overly optimistic proclamations of the programs' designers, and the disappointment as the programs failed to scale. Dreyfus argues that this failure to scale is an inevitable consequence of the fundamental inability of formal symbolic manipulation to flexibly discover and make use of relevant features out of an unarticulated (non-pre-digested) background.

Winograd and Flores [Winograd & Flores 1988] similarly base their arguments on the idea that human behavior takes place against a non-finitely enumerable background. But they extend this argument beyond AI to include all of computer system design. Current computer system design is grounded in the rationalistic tradition, one which they characterize as solving problems via the following steps [Winograd & Flores 1988: 15]:

1. Characterize the situation in terms of identifiable objects with well-defined properties.
2. Find general rules that apply to situations in terms of those objects and properties.
3. Apply the rules logically to the situation of concern, drawing conclusions about what should be done.

They provide an alternative grounding for design based in phenomenology. They wish to shift the question of computer system design from "What can computers do?" to "What can people do with computers?".

Weizenbaum's [Weizenbaum 1976] arguments are concerned with the moral dimension of AI. For him, the important question is "What ought computers be made to do?". His moral concerns arose as a result of people's reactions to the *Doctor* script of *Eliza* (described above). He wrote *Eliza* as an experiment in how far one might go using simple processing techniques in building a conversational computer system. The *Doctor* script was intended as a shallow imitation of an initial interview with a Rogerian psychotherapist. Weizenbaum was shocked by responses to *Eliza*, specifically:

1. Some practicing psychologists believed that extensions of *Doctor* could eventually become a nearly automatic form of psychotherapy. "What must a

psychiatrist who makes such a suggestion think he is doing while treating a patient, when he can view the simplest mechanical parody of a single interviewing technique as having captured anything of the essence of a human encounter?" [Weizenbaum 1976: 6].

2. People conversing with *Doctor* strongly anthropomorphized the computer and become emotionally involved in the interaction. On one occasion, Weizenbaum's secretary asked to be left alone with the system to discuss her problems. "What I had not realized is that extremely short exposures to a relatively simple computer program could induce powerful delusional thinking in quite normal people." [Weizenbaum 1976: 7]
3. There was a widespread popular belief that *Eliza* demonstrated a general solution to the problem of natural language understanding. "This reaction to Eliza showed me more vividly than anything I had seen hitherto the enormously exaggerated attributions an even well-educated audience is capable of making, even strives to make, to a technology it does not understand." [Weizenbaum 1976: 7]

These reactions to *Eliza* lead Weizenbaum to explore moral issues surrounding AI and eventually to abandon the field. The crux of his critique is that AI is infused with an ethic of instrumental reason, a belief that the full richness of the human condition can be reduced to the value-free computation of optimal action. But computer science, while offering a new framework for understanding ourselves, is only one framework among many.

> I … affirm that the computer is a powerful new metaphor for helping us to understand many aspects of the world, but that it enslaves the mind that has no other metaphors and few other resources to call on. The world is many things, and no single framework is large enough to contain them all, neither that of man's science nor that of his poetry, neither that of calculating reason nor that of pure intuition. [Weizenbaum 1976: 277]

As the "soft" understanding of the human condition held by the humanities and arts is replaced by the "hard" understandings of the sciences, including AI, humanity's conception of itself, its subjecthood, is reduced to a mechanical state in which there is no room for empathy, understanding, love, compassion, courage, but only the efficient pursuit of quantifiable goals.

> If the teacher, if anyone, is to be an example of a whole person to others, he must first strive to be a whole person. Without the courage to confront one's inner as well as one's outer worlds, such wholeness is impossible to achieve. Instrumental reason alone cannot lead to it. And there precisely is a crucial difference between man and machine: Man, in order to become whole, must be forever an explorer of both his inner and outer realities. His life is full of risks, but risks he has the courage to accept, because, like the explorer, he learns to trust his own capacities to endure, to overcome. What could it mean to speak of risk, courage, trust, endurance and overcoming when one speaks of machines? [Weizenbaum 1976: 280]

As an AI-based artist I can not ignore these critiques, can not pretend that AI's relationship to the arts and humanities is unproblematic. Much of contemporary technology-based art is a critique of the rationalistic worldview implicit in technology. How do I situate Expressive AI relative to these critiques?

For me the fundamental move is to view AI systems as *procedural representations* of some aspect of the human lifeworld (the amalgam of the physical world plus culture). In the same way that a painting, sculpture or film represents some aspect of the world, and,

through this representation, leads the viewer to a new understanding, a new sense of the significance of the represented person, object or situation, so an AI-based artwork offers the viewer a procedural, possibly interactive representation. Expressive AI does not claim that these representations operate "exactly like a person", nor does it need to in order for these representations to have significance. An AI-based artist can make use of a technique, such as the formal manipulation of symbolic structures, without being committed to the ideology that the technique captures everything it means to be human.

As discussed in Chapter 7, every AI system consists of both a code machine and a rhetorical machine that interprets the operations of the code machine. The architectural surplus of the code system results from the embedding of code signs within broader, connotative meaning systems; words such as "goal", "knowledge", "embodiment", and "emotion" simultaneously refer to specific technical configurations and operations, and aspects of human experience. The productive difference between the technical and human meaning of these terms results in technical innovation, new code machines and new rhetorical constructions for narrating the code machine. The reduction of human subjecthood and experience that troubles these critics occurs when the terms in the rhetorical systems are *naturalized in their technical meanings*. When these happens, the productive difference between the technical and human use of these terms is erased, doing violence to the original richness of the term, reducing its productiveness as a technical term, and contributing to rigidity in human notions of subjectivity. Expressive AI, through its concern with interpretive and authorial affordance, consciously investigates both the code and rhetorical machines, and thus avoids the simple technical reduction of terms in the rhetorical machine.

To the degree that AI research is committed to the functionalist program of reading the operations of AI programs as *identical* to what takes place in human beings, it may seem that Expressive AI, by consciously avoiding this identity and focusing on representation, expression, and interpretation, is no longer AI, that is, can no longer claim to contribute to the field of AI. However, I do believe that Expressive AI continues to be AI research, and that, in fact, Expressive AI, by explicitly viewing AI systems as procedural representations rather than models in the strict sense, is doing what AI has *really* been doing all along, rather than what it sometimes *says* it has been doing. While over the years various influential AI researchers have claimed that AI systems truly model human intelligence (often accompanied by overly optimistic estimates of when general human intelligence will be achieved), if you look at what AI researchers actually do, and how the experience of constructing and running their systems results in new ideas and new system construction, AI programs really look more like thought experiments than empirical experiments, explorations rather than models. Schank certainly describes the work in the AI lab at Yale in this light (see page 188, this chapter). Dennett has similarly argued that AI practice is really one of thought experimentation [Dennett 1997; Dennett 1998a; Dennett 1998b].

> Most AI projects are explorations of ways things might be done, and as such are more like thought experiments than empirical experiments. They differ from philosophical thought experiments not primarily in their content, but in their methodology: they replace some – not all – of the "intuitive", "plausible" hand-waving background assumptions of philosophical thought experiments by constraints dictated by the demand that the model be made to run on a computer. These constraints of time and space, and the exigencies of specification can be traded off against each other in practically limitless ways, so that new "virtual machines" or "virtual architectures" are imposed on the underlying serial

> architecture of the digital computer. … There is very little chance that a philosopher will be surprised (and more pointedly, disappointed) by the results of his own thought experiment, but this happens all the time in AI. [Dennett 1998b: 271-272].

Chapman [Chapman 1990] similarly argues that some of the most important contributions in AI have not been theorems, empirical experiments, the creation of demonstrably superior technological solutions to problems, or philosophically rigorous arguments, but rather have been *approaches*. "An approach is a way of doing research: a way of looking at phenomena, of choosing issues to explore, of stating and attacking problems." [Chapman 1990: 180]. Chapman argues that the implementation of a program serves as an illustration of an approach. Further, he argues, the activity of designing the program itself produces knowledge; design is a way of understanding the world.

AI has always been an exploratory practice of knowing-by-making. Expressive AI just makes this explicit.

Though the critiques described in this section do not directly impact Expressive AI (primarily because Expressive AI does not buy into the ideologies that are the subjects of these critiques), such critiques are still quite useful. In any given AI system the rhetorical and code machines are tightly intertwined – understanding how they intertwine, discovering the hidden philosophical and ideological assumptions carried by the meshwork of code and language, takes active, deliberate work. Critiques such as the ones surveyed here unpack these assumptions (in this case the assumptions of classical, symbolic AI), and help Expressive AI to critically reflect on the foundations of its own practice.

Critical Technical Practice

Artists can engage computer technology in a number of ways. One way is to use the computer as a tool to manipulate and generate images, sounds, and interactive multimedia presentations. This does not require a deep engagement with computer science, but rather a facility with tools written by others. Another mode of engagement is work that is about science and technology. Such work appropriates well understood techniques, ideas and tools to make malfunctioning assemblages that (often humorously) expose the ideological underpinnings of the technology. However, such work does not move forward in offering new, positive dimensions for technological and scientific development. As an artist, I employ a third mode of engagement with AI in which art practice is seen as research, as a mode of inquiry generating new knowledge within AI. Wilson similarly explores art practice as a mode of research in general [Wilson 2002], and a mode of AI research in particular [Wilson 1995].

Expressive AI is a form of critical technical practice (CTP), that is, a technical practice that actively reflects on its own philosophical underpinnings and, by bringing in humanistic and artistic knowledge, approaches, and techniques, results in new technological systems that would not otherwise be built. This reflective critique consciously constructs new, contingent myths to heuristically guide the practice.

Since any CTP is a *technical practice*, an art practice conceived as a CTP does require an ability to genuinely and deeply engage in technological and scientific practice. An electronic media art practice that seeks to be a CTP does require the artist to have the skills and knowledge to engage in activities such as programming, designing new algorithms, designing circuits and robotic hardware, devising and employing the rhetorical and narrative machinery that is an integral part of these novel designs, and

communicating with the "mainstream" practitioners of computer science, engineering, robotics, etc.

Other CTPs

Expressive AI is not the only CTP in AI today. These other critical practices certainly inform my thinking about Expressive AI.

Agre is the first researcher to combine a sustained cultural theoretic inquiry into AI's foundations with continuing work in AI system building. Agre's thesis work in AI found the assumptions of the standard AI view of planning problematic when applied to the dynamics of everyday life [Agre 1988]. Based on this analysis, he developed an alternative architecture that continually re-decides what to do using a dependency maintenance network with deictic, rather than absolute and objective, representations of world objects as inputs. His dissertation work was the culmination of a number of sometimes personally painful years spent attempting to de-familiarize the assumptions implicit in AI work in order to reconnect AI to lived experience [Agre 1997b]. After additional years continuing this work following his dissertation, he developed a deconstructive approach to examining AI foundations, coining the phrase "critical technical practice" to refer to any such attempt to incorporate reflection on foundations into day-to-day technical practice itself [Agre 1997a].

Sengers engages in Cultural Informatics [Sengers 1999b], that is, a technical practice grounded in an understanding of the relationship between computer science research and broader culture. To date her work has focused on exploring the gap between subjective experience and the mechanization of thought. Employing cultural theory to unpack the construction of the subject/object split, she examines how this split manifests in AI in the construction of autonomous agents. Based on this analysis, she builds agents organized around alternative principles [Sengers 1998a; Sengers 1999a].

Penny engages in reflexive engineering, combining art practice with robotics. He is interested in exploring assumptions in the technical notion of an agent and building agent-based artwork built on alternative assumptions [Penny 2000]. Work such as the robotic installation *Petit Mal* has examined the notion of physical embodiment, specifically exploring how much of the intelligence exhibited in the robot's interactions with viewers is a result of the physical design of the robot and the physicality of the viewer's interaction with the robot [Penny 1997]. Penny has also commented on the disciplinary and cultural issues involved in being an artist engaged in a technical practice [Penny 1995].

Sack employs a cultural studies perspective on language to engage in the computer analysis of human language use. The cultural studies perspective leads him to work on problems that are marginalized by mainstream linguistics. For example, *Spindoctor* [Sack 2000a] uses statistical techniques to classify ideological point-of-view expressed in news stories about the Nicaraguan Contras. The *Conversation Map* employs a social network approach to automatically analyze large scale, distributed conversations taking place in netnews groups [Sack 2000b; Sack 2002].

Structure of Expressive AI as a CTP

In Chapter 4 I took pains to undermine any claim interactionist AI might have for being peculiarly suited for artistic practice by diagnosing the link that exists between cultural theoretic critiques of Enlightenment rationality and interactionist AI. This may have left the reader with the impression that I am hostile to cultural theoretic studies of AI. This is

not the case. Culture theory is extremely valuable for unpacking hidden assumptions lurking in AI practice. Understanding these assumptions allows an artist to gain a free relation to AI technology, to avoid being forced into the "natural" interpretation of the technology that has been historically constructed. It is only the implicit claim that a particular technology is suited for artistic expression that Expressive AI rejects. Cultural studies of AI help a practitioner to maintain a free relation to technology – this, however is a process, not an achievable end. There is no final, "perfect" AI to be found, for artistic or any other purpose.

Expressive AI sits on three legs: technical practice, art practice, and cultural theory . Cultural theory serves as the transducer, the relay, between art practice and technical practice.

Science Studies (cultural studies of science) seeks to demystify the ideological structures informing scientific practice, to understand the political and social mechanisms through which knowledge construction occurs in science. This is primarily a project of negation, exposing the naturalized structures constructing science as an autonomous, non-contingent, value-neutral activity. Criticism (i.e. demystification) alone is not enough to be CTP. The toolbox of the cultural theorist is primarily a toolbox of negation. There is certainly a mass of naturalized ideology surrounding technical practices, and negating this naturalized ideology is precisely what makes cultural theory part of CTP. But a CTP continues to produce positive knowledge within its domain. Expressive AI continues to produce AI research that can be understood as new knowledge by other AI practitioners. But this new knowledge is now being produced within the maelstrom of an always contingent, infinitely ramifying, self-reflexive loop, rather than from the stable, unmoving comfort of naturalized bedrock. Knowledge production is not a project of negation, but rather of construction. Tools of negation alone will not suffice.

The three-legged stool of art, AI and cultural theory provides a combination of tools of construction and tools of negation. Art and AI provide two "positive" modes of knowledge production that mutually inform and alter each other; cultural theory keeps things boiling, preventing sedimentation and the eventual naturalization of a bedrock, providing a language for uncovering and articulating assumptions, and serving as a transducer/relay between the two practices.

This tripartite structure is visible in other CTPs, such as Sengers' and Agre's work. In Sengers' case [Sengers 1998a], the two positive polls are narrative psychology and AI, while the negative poll is schizo-analysis. Schizo-analysis is used to diagnose a problem in reactive agent architectures, while Brunner's narrative psychology, when understood within the heuristic matrix established by her schizo-analysis of agents, produces new knowledge in both AI (a technical solution and, more importantly, a story about that technical solution), and narrative psychology (says something new about Brunner's theory through concrete actualization). In Agre's case [Agre 1997a], the two positive polls are the phenomenology of everyday life (ethnography, micro-sociology) and AI, while the negative poll is deconstruction. Deconstruction is used to invert hierarchical oppositions latent in AI research (particularly the inside/outside master narrative of mind/world), while phenomenology of everyday life, when understood in the heuristic matrix produced by the deconstructive analysis, produces new knowledge in both AI (a technical solution and accompanying story about the technical solution) and ethnography (says something new about ethnographic understandings of life routines through concrete actualization).

Conclusion

Expressive AI is a new interdiscipline of AI-based cultural production combining art practice and AI research practice. Expressive AI changes the focus from an AI system as a thing in itself (presumably demonstrating some essential feature of intelligence), to the system as a representation of some aspect of the human world, as a device for communication between author and audience. The technical practice of building the artifact becomes one of exploring the manifold relationships that exist between interpretive and authorial affordance in the material realization of procedural systems. Expressive AI does not single out a particular technical tradition as being peculiarly suited to culture production. Rather, as a critical technical practice, Expressive AI is a stance or viewpoint from which all of AI can be rethought and transformed.

CHAPTER 10
NATURAL LANGUAGE PROCESSING IN *FAÇADE*

The *Façade* natural language processing (NLP) system accepts surface text utterances from the player and decides what reaction(s) the characters should have to the utterance. For example, if the player types "Grace isn't telling the truth", the NLP system is responsible for determining that this is a form of criticism, and deciding what reaction Grace and Trip should have to Grace being criticized in the current context. General natural language understanding is of course a notoriously difficult problem. Building a system that could understand open-ended natural language utterances would require common sense reasoning, the huge open-ended mass of sensory-motor competencies, knowledge and reasoning skills that human beings make use of in their everyday dealings with the world. While *Façade* is a micro-domain, a dramatically-heightened representation of a specific situation, not the whole world, there are still no general theories, techniques or systems that can handle the syntactic, semantic and pragmatic breadth of the language use that occurs in *Façade*. Instead, *Façade* makes use of specific (non-general), a-theoretical, author-intensive techniques to understand natural language typed by the player.

As mentioned in Chapter 3, our approach in *Façade* is to view the natural language understanding problem as a dialog management problem. The system focuses on the pragmatic effects of language, how a player's utterances affect Trip and Grace, and thus change the evolving dramatic situation, rather than the syntax (form) or semantics (truth assertions) of the player text. In *Façade*, NLP is divided into two phases: phase I maps surface text into speech acts (here called discourse acts) that represent the pragmatic effects of an utterance, while phase II maps discourse acts into one or more character responses.

Phase I: Surface Text to Discourse Acts

Forward chaining rules map surface text to discourse acts. Some rules map specific patterns of surface text directly to intermediate meaning representations, while other rules combine intermediate meanings to form more complex meanings. Eventually the process of mapping islands of surface text into intermediate representations, and combining these representations, produces the final meaning, consisting of one or more discourse acts.

For example, imagine that the player types "Hello Grace". One of the discourse acts, (DAGreet ?character), represents the player greeting a character – we want the system to map the text "Hello Grace" into (DAGreet Grace). This could be accomplished with the rules in Figure 10-1.

```
"hello" → iGreet
"grace" → iCharacter(Grace)
iGreet AND iCharacter(?x) → DAGreet(?x)
```

Figure 10-1. Pseudo-code for simple greeting rules

The first rule matches on the appearance of the word "hello" anywhere in the text and asserts an iGreet[33] fact. This captures the fact that "greeting-like" language appeared in the text. The second rule matches on the appearance of the word "grace" anywhere in the text and asserts an (iCharacter Grace) fact. This captures the fact that a reference to Grace appeared in the text. The third rule matches on the occurrence of the two intermediate facts and asserts a (DAGreet Grace) fact, indicating that a greeting was directed at Grace. The third rule makes use of a variable; ?x binds to the argument of iCharacter on the left hand side and brings this value over to the assertion of DAGreet on the right hand side of the rule. In order to capture more ways of saying hello (e.g. "how're you doing?", "how's it going?", "what's up", etc.), additional iGreet rules can be written without changing the iCharacter or DAGreet rules.

Discourse Acts

Our current set of discourse acts appears in Table 10-1 below. All player utterances map into one or more of these discourse acts. Phase I processing is a strong many-to-few mapping – the huge, rich range of all possible strings a player could type is mapped onto this small set of discourse acts. It's as if the system hears the player speak like a cave-man – when the player types "Grace, you're so uptight about that", or "That wedding picture makes a real statement", or "You two should really try to communicate better", the system hears "me *criticize* grace", or "me *refer to* wedding picture", or "me *give advice*", respectively. Besides ignoring any nuance in the tone of the player's text, the system also focuses almost exclusively on the pragmatics of the utterance, ignoring most of the semantics (denotative meaning).

Most of the discourse acts can be directed at a character (indicated by the variable ?character in the table below). For example, the text "Certainly, Grace", is an *agree* act directed at Grace – the agreement with Grace is represented (DAAgree Grace). If the player just typed "Certainly", the agreement is not directed at any specific character – this undirected agreement is represented (DAAgree none).

Repesentation of Discourse Acts	***Pragmatic Meaning of Discourse Acts***
(DAAgree ?character)	Agree with a character. Examples: "Be my guest", "certainly", "I would love to"
(DADisagree ?character)	Disagree with a character. Examples: "No way", "Don't make me laugh", "Not by a long shot"
(DAPositiveExcl ?character)	A positive exclamation, potentially directed at a character. Examples: "Yeah", "hotdog", "breath of fresh air"
(DANegExcl ?character)	A negative exclamation, potentially directed at a character. Examples: "Damn", "how awful", "I can't stomach that", "I'm at the end of my rope"
(DAExpress ?character ?type)	Express an emotion, potentially directed at a character. The emotion types are *happy* ("I'm thrilled", ":-)"), *sad* ("That makes me blue", ":-("), *laughter* ("ha ha", "lol[34]"), and *angry* ("That pisses me off", "grrrr").

[33] By convention, facts representing intermediate meanings begin with the letter "i" (for intermediate), while facts representing discourse acts begin with "DA" (for discourse act).
[34] "Lol" is chat room speak for "laugh out loud."

Repesentation of Discourse Acts	*Pragmatic Meaning of Discourse Acts*
(DAMaybeUnsure ?character)	Unsure or indecisive, potentially directed at a character. This discourse act is usually a response to a question. Examples: "I don't know", "maybe", "I guess so"
(DADontUnderstand ?character)	Don't understand a character utterance, or the current situation. Examples: "I'm confused", "I can't make heads or tails of it", "What are you talking about"
(DAThank ?character)	Thank a character. Examples: "Thank you", "Thanks a lot"
(DAApologize ?character)	Apologize to a character. Examples: "I'm sorry", "I've got egg on my face", "Open mouth, insert foot"
(DAPraise ?character)	Praise a character. Examples: "You're a genius", "You're a real sweetheart", "That's a great idea"
(DACriticize ?character ?level)	Criticize a character. There are two levels of criticism, *light* ("You're stuck up", "Don't be so up tight"), and *harsh* ("Idiot", "What a dipshit")
(DAFlirt ?character)	Flirt with a character. Examples: "You look gorgeous", "Let's get together alone sometime"
(DAPacify ?character)	Pacify a character. Examples: "Calm down", "Keep your shirt on", "Take it easy"
(DAAlly ?character)	Ally with a character. Examples: "I like you", "You are my friend"
(DAOppose ?character)	Oppose a character. Examples: "Kiss off", "You're my enemy", "I hate you"
(DAJudgement ?character)	Judge a character. Examples: "You are wrong", "Trip is cheating", "Grace is lying"
(DAAdvice ?character)	Give advice to a character. Examples: "You should get a divorce", "Try to work it out"
(DAReferTo ?character ?object)	Refer to an object, potentially directed at a character. There are a number of different objects in the room, including the *couch* ("I like the couch"), the *wedding picture* ("Your wedding picture looks nice"), and *paintings* ("Where did you get these paintings").
(DAGreet ?character)	Greet a character. Examples: "Hello", "What's up"
(DAIntimate ?character)	Ask a character to share their thoughts or feelings with you. Examples: "What's wrong", "Let it all out", "Talk to me"
(DAGoodbye ?character)	Say goodbye to a character. Examples: "Catch you later", "So long", "I'm out of here"
(DAJokeTestLimits ?character)	Utterances that purposely try to break the system, perhaps by referring directly to the fact that these are computer characters. Potentially directed at a character. Examples: "Who is your creator", "What's the meaning of life", "God is dead"
(DAInappropriate ?character)	Utterances containing vulgar or otherwise inappropriate words or phrases (inappropriate for the social situation in *Façade*). Examples: "blow job", "slut"

Repesentation of Discourse Acts	*Pragmatic Meaning of Discourse Acts*
(DAMisc ?character ?type)	Miscellaneous specialized discourse acts, often specific to certain beats or contexts. The *type* encodes the specialized act. New DAMisc types are created as needed. Current types include *ask for drink* ("I'd like a drink"), and *should I leave* ("Is this a bad time", "Should I leave").
(DASystemCannotUnderstand)	Catch-all for all utterances that trigger no other discourse acts.

Table 10-1. *Façade* discourse acts

The Template Language

The rules that map surface text to discourse acts are written in a custom rule language that compiles to Jess [Friedman-Hill 1995-2002], a Java implementation of the CLIPS rule language[35] [NASA 1985-2002]. The custom rule language is a superset of Jess, adding an embedded template description language that allows compact descriptions of surface text patterns to appear on the left hand side of rules. The template rule compiler accepts a file containing Jess rules in which templates appear on the left hand sides of rules, and produces a file containing "pure" Jess rules, in which template tests have themselves been transformed into additional rules. Any rules in the original file that *don't* perform a template test on the left hand side pass through the template compiler unchanged. The initial implementation of the template compiler was done by Mehmet Fidanboylu, an undergraduate who did his senior thesis (*Natural Language Understanding for Interactive Drama*) under my direction.

A typical Jess rule, and a rule making use of the template sub-language, appear in Figure 10-2. The first rule in Figure 10-2 is from a Jess solution for the monkey and bananas toy domain, a planning domain in which a monkey overcomes a number of obstacles to grab bananas that are initially out of reach. Jess rules, like any standard forward chaining rule language, consist of a left hand side (to the left of the =>, abbreviated LHS), which specifies a condition that must be true in order for the rule to fire, and a right hand side (to the right of the =>, abbreviated RHS), which lists a set of actions to perform if the condition on the LHS is true. Conditions are tested and actions are performed against a knowledge base, not unlike the working memory of an ABL agent. The LHS of the monkey and bananas rule tests whether the knowledge base currently contains a goal fact to hold some object (the monkey has the goal to hold the object), a fact that some chest contains the object (a fact about the world), and that the knowledge base doesn't contain a goal fact to unlock the chest (the monkey doesn't currently have the goal to unlock the chest). If this compound condition is satisfied, the RHS asserts (adds) a new goal fact to the knowledge base indicating that the monkey now holds a goal to unlock the chest. Fact tests on the LHS can contain variables (indicated by tokens beginning with ?); like WME tests (e.g. Figure 5-3 on page 68), variable bindings chain across individual fact tests. Any bindings established on the LHS are available on the RHS (like ?chest in the example).

[35] While most rule collections written in Jess would run in implementations of Clips (and vice-versa), Jess provides additional features not present in Clips, such as an interface between Jess and Java that allows Jess rules and Java objects to mutually invoke each other. This capability is in fact used during phase II of NLP, reaction selection.

```
;; Jess rule from monkey and bananas domain
(defrule unlock-chest-to-hold-object ""
    (goal-is-to (action hold) (argument-1 ?obj))
    (chest (name ?chest) (contents ?obj))
    (not (goal-is-to (action unlock) (argument-1 ?chest)))
    =>
    (assert (goal-is-to (action unlock) (argument-1 ?chest))))

;; A template rule for agreement
(defrule global-agree-rule1
    (template (toc (love (to | it | (nothing more)))))
    =>
    (assert (iAgree)))
```

Figure 10-2. Examples of Jess rules with and without the template sub-language

The second rule in Figure 10-2 contains a template test on the LHS. The template test looks similar to a fact test (a test of the fact template), except that the body of the template fact, instead of being slots containing values, is instead a specification of a pattern that will be tested over the input string. If the input string matches this pattern, the RHS asserts an iAgree fact (an intermediate fact) that will eventually be combined with an intermediate fact indicating the character addressed (potentially the null character none) to produce the final DAAgree fact. The pattern in this case is one of the patterns for an *agree* discourse act – surface text is an instance of DAAgree if it contains "love to" or "love it" or "love nothing more". Thus if the player types "I would love it", or "I love nothing more than that, Grace", or "coffee dog love it boing boing", an iAgree fact is asserted. The troubling promiscuity of template techniques, which tend to accept (map to meaning expressions) highly ungrammatical utterances, is discussed on page 213.

The Pattern Language

The sub-language for specifying template patterns consists of a combination of regular expressions and occurs expressions. Regular expressions are sensitive to the positions of terms in a pattern –for the regular expression (love it) to match a string, "it" must appear immediately following "love" in the string. Occurs expressions don't care about position, only term occurrence – for the occurs expression (tand love it) to match a string, "love" and "it" must both appear in the string, but do not have to be contiguous and can appear in any order. The occurs expressions are inspired by the simple template matching language in the Oz text world. Examples of all the template pattern language expressions appear in Table 10-2.

Example Expressions	*Meaning*
(X Y)	An *and* regular expression. Matches an input string if it consists of X immediately followed by Y.
(X | Y)	An *or* regular expression. Matches an input string if it consists of X or Y.
([X])	An *optional* regular expression. Matches an input string if it consists of X or nothing.

Example Expressions	*Meaning*
*	A *match all* wildcard. Matches any number of words in an input string.
?	A *match one* wildcard. Matches any one word in an input string.
(tand X Y)	An *and* occurs expression (**tand** is short for template-and). Matches an input string if it contains X and Y in any order.
(tor X Y)	An *or* occurs expression (**tor** is short for template-or). Matches a input string if it contains X or Y.
(toc X)	An *occurrence* occurs expression (**toc** is short for template-occurs). Matches an input string if it contains X.
(tnot x)	A *not* occurs expression (**tnot** is short for template-not). Matches an input string if X does not occur in it.

Table 10-2. Template pattern language expressions

The tor, tand and toc expressions are syntactic sugar for classes of regular expressions: (tor X Y) can be rewritten as ((* X *) | (* Y *)) (toc is a special case of tor), and (tand X Y) can be rewritten as ((* X * Y *) | (* Y * X *)).

Of course the various expressions can be recursively nested to produce compound patterns such as (tand (X (tnot Y)) Z) or (X | ([(tor Y Z)] W)).

When occurs expressions appear at the top level of a compound pattern, they test across the whole input string. But when the occurs expression is embedded in a regular expression, the occurs expression only tests across the substring delimited by the regular expression. For example, in the pattern (i love you (tnot kill) *), the occurs pattern (tnot kill) is tested against the portion of the string after the match against (i love you). So when testing this pattern against the string "I love you so very much", the (tnot kill) matches if the word "kill" does not occur in "so very much" (which in fact it doesn't). But the pattern does not match "I love you so much I could kill", since the word "kill" *does* occur in the substring "so much I could kill".

When the top-level expression of a template is a regular expression, the regular expression must span the string in order to match. For example, in the template rule in Figure 10-2, if the template was (love (to | it | (nothing more)) (without the toc), it would only match the exact strings "love to", "love it", or "love nothing more", but *not* strings like "I would love nothing more than that".

The basic atomic terms in a pattern are individual words, such as the words "thank" and "you" in the pattern (thank you *). To save template rule authors from having to type quote marks all over the place, words appear directly without quote marks in patterns. This is the reason why the keywords tand, tor, tnot, and toc are used – if the normal English words *and*, *or*, *not* and *occurs* were used, there would be no way to tell the difference between a regular expression and an occurs expression. But individual words are not the only atomic terms in the pattern language: besides matching an individual word, one can also (recursively) match a fact, match an entire collection of words

determined by WordNet [Fellbaum 1998] expansion, and match stemmed forms of words.

Matching Positional Facts

Facts asserted by template rules can be *positional facts*. A positional fact includes information about the substring range matched by the rule that asserted the fact. Positional facts can then be matched as atomic terms within other templates. An example appears in Figure 10-3.

```
;; Rule for recognizing positional "is" fact
(defrule positional_Is
    (template (tor  am are is seem seems sound sounds look looks))
=>
    (assert (iIs ?startpos ?endpos)))

;; Rule for recognizing positional "positive description" fact
(defrule positional_PersonPositiveDescription
    (template (tor buddy clever comrade confidant friend genius go-getter
      intelligent kind pal smart sweetheart))
=>
    (assert (iPersonPosDesc ?startpos ?endpos)))

;; Rule for recognizing praise
(defrule Praise_you_are_positive
    (template ({iPersonPosDesc} | (you [{iIs}] [a | my] {iPersonPosDesc} *)))
=>
    (assert (iPraise)))
```

Figure 10-3. Example rule for recognizing *praise* discourse act using positional facts

The first rule, positional_Is, asserts a positional fact when an "is" word is recognized. "Is" words are words used in conversation to indicate state of being, such as "is" "seem", or "look". For example, "You *look* short", "You *are* short", and "You *seem* short", to a first approximation, all mean the same thing (and for the coarse semantics of *Façade*, first approximations are all we're looking for). When any of these "is" words appears in an input string, the fact (iIs ?startpos ?endpos) is asserted. The appearance of the special variables ?startpos and ?endpos is what makes this a positional fact – these variables are bound to the starting and ending position of the substring matched by the template[36]. In this case, since the template consists of a tor expression testing single words, the starting and ending position will both be the same value, the position of the "is" word within the string.

The second rule, positional_PesonPositiveDescription, asserts a positional fact when a word or phrase that is a positive description of a person appears.

The last rule, Praise_you_are_positive, is one of the rules for recognizing *praise* discourse acts. This particular praise rule recognizes praises that consist of only a positive description (e.g. "friend"), or of an assertion of the form "You are <positive description>" (e.g. "You are a friend"). The template matches on positional facts – positional fact matches are indicated by curly braces. So the regular expression (you [{iIs}] [like | a | my] {iPersonPosDesc} *) matches any string beginning with the word "you",

[36] One can imagine cleaner mechanisms for making a fact a positional fact, such as a declaration syntax for declaring a fact as positional, with the automatic inclusion of positional information whenever such a declared fact is asserted; this would avoid the messy "special" variables. This is an easy minor improvement to make to the template compiler.

optionally followed by a substring that was recognized as an iIs (intermediate fact for *is*) by some other rule, optionally followed by "like", "a" or "my", followed by a substring that was recognized as an iPersonPosDesc (intermediate fact for positive person description) by some other rule. This pattern matches strings such as "You are a genius", and "You sound like a good egg". When an atomic term appear in curly braces, this tells the template compiler to treat the terms as the names for positional facts, and to use the positional information (start position and end position) associated with the fact to determine if the pattern matches.

Positional facts make the template language as powerful as a context-free grammar. However, the *spirit* of the template language is to recognize simple text patterns and map them to discourse acts, not to build grammars that accept all and only the set of grammatically correct strings. Given how broad the language use is in *Façade*, and given the disfluencies of conversational typed language, building such a grammar would be a daunting undertaking. And even assuming you had such a grammar, the goal is not just to determine if a string is grammatical or not, but to determine what discourse act the string accomplishes. The only near-term hope for recognizing the pragmatics of a broad set of utterances (mapping utterances into discourse acts) is to relatively directly map shallow surface text patterns (which may include recursive structure, such as the praise example in Figure 10-3) into these acts.

WordNet Expansions

To increase the number of strings matched by a given template, patterns can request that atomic terms be expanded to include synonyms (words with the same meaning) and hyponyms (words with more specific meanings). The WordNet [Fellbaum 1998] lexical thesaurus is used to perform term expansions.

WordNet consists of a collection of nodes called synsets organized into a labeled directed graph. Each synset corresponds to a different meaning (called a *sense* in WordNet terminology), and contains the English words and phrases that express that sense, a dictionary definition of the sense, and the part of speech of the synset. Since words can have multiple meanings, a word may belong to multiple synsets. For example, the word "alcohol" has two noun senses in WordNet:

1. "a liquor or brew containing alcohol as the active agent" – synset members are "alcohol", "alcoholic beverage", "drink", "intoxicant", "inebriant"
2. "any of a series of volatile hydroxyl compounds that are made from hydrocarbons by distillation – the only synset member is "alcohol"

Synsets are organized in a labeled directed graph, where each link label corresponds to a different semantic relationship between two synsets. The possible semantic relationships (link types) depend on the part of speech of the two synsets. For nouns, the 11 link types include *hypernym* (more general synsets of which the given synset is a specific kind), *hyponyms* (opposite of hypernym, that is, synsets that are specific kinds of the given synset), and *part meronyms* (synsets for objects of which this synset is a part). For verbs, the 6 link types include *hypernyms*, *hyponyms* and *antonyms* (synsets for actions opposite to the given synset). For adjectives, the 6 link types include *antonyms* and *similar* (synsets for adjectives similar to the given synset). For adverbs, the two link types are antonym and *derived-from* (the adjective from which the adverb was derived, if any).

Examples of template term expansions appear in Table 10-3. Since different senses of the same word may belong to multiple parts of speech, template authors must explicitly tag the term expansions with the part of speech to use for the term expansion. If, for the

given part of speech, there are still multiple senses of the term, the template language currently performs term expansions across all senses[37].

Term expansion expression	*Meaning*
<drink:N>	Matches all synonyms of the noun *drink* (members of noun synsets that drink belongs to) and all synonyms recursively reachable via hyponym links. Matches 750 words and phrases including "milkshake", "orange juice", "beer", and "anisette de Bordeaux".
<drink:V>	Matches all synonyms of the verb *drink* (members of verb synsets that drink belongs to) and all synonyms recursively reachable via hyponym links. Matches 46 words and phrases including "imbibe", "sip", "swill", and "tipple".
<soft:ADJ>	Matches all synonyms of the adjective *soft* (members of adjective synsets that soft belongs to) and all synonyms recursively reachable via similar links. Matches 95 words and phrases including "mushy", "subdued", "easy", and "tender".
<quickly:ADV>	Matches all synonyms of the adverb quickly (members of adverb synsets that quickly belongs to). Matches 8 words and phrases including "rapidly", "speedily" and "cursorily".

Table 10-3. Example term expansion expressions

An example rule using term expansion expressions appears in Figure 10-4.

```
(defrule drink-request
   (template (i * <want:V> * (toc <alcohol:N>)))
   =>
   (assert (iRequest eRequestType drink)))
```

Figure 10-4. Example rule using term expansion

With the term expansion on the verb "want" and the noun "alcohol", the template matches a large number of strings including: "I lust for your best red Bordeaux", "I hanker after vin Ordinaire", "I cry out in need of Armagnac", "I crave whiskey on the rocks".

The template compiler expands terms offline during compilation. This saves having to search the WordNet graph online during template matching.

[37] Within a part of speech, WordNet senses are uniquely numbered. If expanding terms across all senses within a part of speech turns out to offer insufficient control, it is a simple matter to extend the term expansion syntax to include both a part of speech tag and a sense number.

Stemming

Sometimes a template rule will not care about word stems, such as noun pluralization or verb conjugation. Just knowing the root form of the word is enough for the pattern to match. The syntax <root:S> is used to indicate that any form of a root word should constitute a match. For example, "seem", "seems", "seeming", and "seemed" are all matched by <seem:S>. Using this root syntax, the positional-Is rule at the top of Figure 10-3 on page 206 can be rewritten as shown in Figure 10-5.

```
(defrule positional_Is
   (template (tor   <be:S> <seem:S> <sound:S> <look:S>))
=>
   (assert (iIs ?startpos ?endpos)))
```

Figure 10-5. Example rule matching word roots

Term Retraction

When a template matches terms (words and patterns) within a string, it can remove terms to prevent other templates from matching on the same terms. This is particularly useful when matching idiomatic phrases – since the words in an idiomatic phrase are being used in an unusual way, they could potentially trigger other templates that match on the same words according to their more usual meanings. The retraction operator "–" can be placed in front of any term in a template pattern. If the input string is matched by the template (the LHS of the rule matches), the marked words are retracted from the internal representation of the string. The retraction operator can be thought of as creating an implicit retraction action on the right hand side of the rule. Figure 10-6 shows an example rule making use of retraction. The template matches the agreement idiom "by all means". If the LHS matches, the facts corresponding to the contiguous text "by all means" are deleted from the surface text.

```
(defrule agreement-idiom
   (template (toc -(by all means)))
   =>
   (assert (iAgree)))
```

Figure 10-6. Example rule using retraction operator

Miscellaneous Declarations

Besides the pattern language for specifying templates, the template language also includes two declarations: defbeat, and defmap.

The defbeat declaration is used to associate a collection of template rules with a beat name (a label). By switching between named collections of beat-specific rules at runtime, different beat contexts can "listen" for different patterns of surface text, or map the same patterns of surface text to different discourse acts.

The defmap declaration associates a fact name with a WME class. Whenever a fact is asserted that has a WME class association, a WME is created with fields filled in with values from the fact and placed in story memory. This is the mechanism by which discourse acts recognized by the template system are communicated to the ABL agents and drama manager.

Compilation Strategy

Input strings are represented as a collection of facts, one fact for each word. Each unique word is represented as a unique word occurrence fact; each of these unique facts includes the start position and end position of the word, which are always the same[38]. For example, for performing template matching on the input string "I want a glass of wine", the string is represented as a collection of six facts: (wo-i 1 1) (wo-want 2 2) (wo-a 3 3) (wo-glass 4 4) (wo-of 5 5) (wo-wine 6 6). The fact names for each word in a string are constructed by appending the word onto "wo-" (for *word occurrence*).

The template compiler compiles template patterns into collections of rules whose LHSs ultimately test word occurrence facts. Thus word occurrence (wo) facts are chained together by generated rules to produce intermediate phrase occurrence (po) facts. The intermediate phrase occurrence facts are ultimately tested on the LHS of the authored rules that assert discourse act facts. By representing the individual words of an input string as word occurrence facts, and compiling templates into networks of rules that test these facts, the expensive string tests that would potentially have to be run for every template rule LHS are eliminated. Further matching efficiencies are gained through Jess' use of the Rete [Forgy 1982] matching algorithm. Some example compilation strategies for transforming template patterns into rules appear in Table 10-4.

Pattern Expression	*Compilation Strategy (pseudo-code)*
expr1 | expr2	expr1(?startpos ?endpos) → po-regexp-or(?startpos ?endpos) expr2(?startpos ?endpos) → po-regexp-or(?startpos ?endpos)
(expr1 expr2)	expr1(?startpos1 ?endpos1) expr2(?startpos2 ?endpos2) ?startpos2 = ?endpos1 + 1 → po-regexp-and(?startpos1 ?endpos2)
(tand expr1 expr2)	expr1(?startpos1 ?endpos1) expr2(?startpos2 ?endpos2) → po-occursexp-and(min(?startpos1 ?startpos2) max(?endpos1 ?endpos2))
(regexp1 occursexp2)	po-regexp1(?startpos1 ?endpos1) po-occursexp2(?startpos2 ?endpos2) ?startpos2 > ?endpos1 → po-regexp2(?startpos1 ?endpos2)

Table 10-4. Example compilation strategies for template patterns

In the rule pseudo-code for the different compilation strategies, if the expressions matched on the LHS are themselves patterns, then the match occurs against phrase occurrence (po) facts. Each unique pattern has a corresponding unique phrase occurrence

[38] Even though words always have the same start and end position, word occurrence facts explicitly represent the start and end position in order to make it easy for rules to combine word occurrence facts with phrase facts, which *do* have different start and end positions.

fact. If the expressions matched on the LHS are words, then the match occurs against word occurrence (wo) facts.

Runtime Processing

This section describes how matching an input string against templates fits into the rest of the ABL architecture at runtime, when a player is playing *Façade*.

When the drama manager sequences a new beat, any beat-specific template rules associated with the old beat are deactivated (retracted) and any beat-specific template rules associated with the new beat are activated (asserted). The defbeat declaration is used in template files to associate a collection of template rules with a specific beat. Most of the time a large number of general rules are always active, with a smaller number of beat rules used to listen for language specific to the beat context.

When the player types text, an ABL behavior grabs the text (using a text sensor to get the text from the animation engine) and passes it to the rule engine to match active template rules against the string[39]. Each word in the input string is asserted as a word occurrence fact in the rule engine. The following preprocessing steps are performed as the word occurrence facts are asserted. First, the string is converted to lower case. Second, all contractions are expanded (e.g. "can't" becomes "can not"). Finally, stemming is performed on all words. A WordNet stemming package is used, which stems using a combination of simple rule-based stemming similar to the Porter algorithm [Porter 1980] and lookup in an exception list for roots that transform irregularly under stemming. For stemmed words, both the original form and the root are asserted, the original word as a word occurrence fact, and the root as a root occurrence fact. For example, if the input string is "Trip seems pensive", both (wo-seems 2 2) and (ro-seem 2 2) (ro for *root occurrence*)are asserted. Asserting both the original and root form allows some template patterns to match on exact word forms and other patterns to match on any form of a root (patterns that use the <root:S> syntax). In addition to the word and root occurrence facts, a _wordcount fact is asserted that stores the number of words in the input string. Some template rules, in addition to testing a template pattern, also test the _wordcount fact to restrict matches to input strings less than some length. Finally, the template rules are allowed to run – processing continues until the rule engine achieves quiescence (no more rules fire).

As template rules fire, many of them will assert intermediate facts, that is, facts corresponding to sub-discourse act patterns (for example, all the rules in Figure 10-3 assert intermediate facts). Occasionally a rule will fire that asserts a final fact, that is, a fact that directly corresponds to a discourse act. When a final fact is asserted, the template runtime system automatically creates a WME corresponding to the discourse act, and adds the WME to story memory. Final facts are declared using the defmap construct, which associates a fact and WME type. In *Façade*, one WME type is used to represent all discourse acts, using field values to distinguish between the various acts. So for *Façade* there is only one defmap declaration: (defmap DA DAWME).

When the player types text, the ABL characters are aware of it in two ways. First, as the player types, a sensor indicates that there is keyboard activity, but does not provide access to the individual keystrokes. Since characters are aware of keyboard activity, this allows them to pause and look expectantly at the player while the player types, as if they hear the player speaking, and are waiting for the player to finish before responding. When

[39] The rule engine is wrapped by a Java object – the ABL behavior passes the string to the rule engine simply by calling a method on the wrapper object from within a mental act.

the player hits the return key, the text sensor retrieves the input string and passes it to phase I (the rule engine containing template rules) for processing. As discourse acts are recognized by the template rules, discourse act WMEs pop into existence in the story memory; ABL demons waiting on discourse act WMEs pass them to phase II to determine what reaction to perform.

A given input string may be ambiguous, resulting in the recognition of multiple discourse acts. It is up to phase II, where a reaction is selected, to decide what reaction to perform if there are multiple discourse acts.

Idiom for General Template Rules

The general (non-beat specific) template rules are organized in the following way. First, a collection of high salience template rules recognize generically useful patterns, such as the "is" rule in Figure 10-5. Salience is similar to ABL priority. Jess has a decision cycle (analogous to the ABL cycle) that repeatedly tests the LHSs of all rules to determine which ones could potentially fire (the conflict set), and chooses one rule from the conflict set to actually fire. Salience declarations can be used to declare that some rules should be preferred over others; the Jess decision cycle always chooses from among the highest salience rules in the conflict set. Since the template language is just a sub-language embedded in Jess, template rules can make use of all Jess constructs, including salience declarations. Salience declarations can be used to stage template rules, with higher-salience rules recognizing lower-level features than lower-salience rules.

After general low-level patterns such as "is words" have been recognized, the next tier of rules recognizes idiomatic expressions, such as the agreement idiom in Figure 10-6. Each idiomatic rule uses the retraction operator to retract the idiomatic expression once it is found – this prevents other rules from incorrectly matching on individual words in the idiomatic expression. Idiomatic expressions are treated as if they are compound words that directly express some particular meaning; we don't want other template rules to accidentally match the substructure of this compound word. There are generally a large number of idiomatic expression rules for each discourse act (e.g. agree) or sub-discourse act (e.g. idioms for "person positive expression", which are used by the praise rule in Figure 10-3). To compile an initial list of such idioms, we examined the phrase resources [Magnuson 1995-2002; Oliver 1995-2002; PhraseFinder 2002; Phrase Thesaurus 2002] to compile a large number of expressions (~ 9000 phrases); ~1000 of these were applicable to our domain and were categorized in terms of our discourse acts.

The final tier of rules consists of keyword and combination rules. Keyword rules watch for individual words or short, non-idiomatic phrases that are indicative of a discourse act or sub-discourse act. The positional_PersonPositiveDescription rule in Figure 10-3 is an example of a sub-discourse act keyword rule, while the rule in Figure 10-7 is an example of a discourse act keyword rule. In an attempt to keep down the number of false positives, discourse act keyword rules tend to impose a limit on the total number of words that can appear in the surface utterance. Unlike sub-discourse act keword rules, which can depend on combination rules further along the chain to impose additional constraints, discourse act keyword rules try to directly recognize discourse acts based on the presence of keywords or short phrases. The longer the utterance, the more likely such rules will be fooled because something else in the utterance contradicts the keyword assumption.

```
(defrule giAgree_Keyword
   (template (tor absolutely agree agreement allright  alright
     always approve aye bigtime certainly cinch definitely
     (of course) fine clearly completely yeah yep yes yessir
     yup oui dig word  (my pleasure) boogie greenlight sure))
   (_wordcount ?count&:(< ?count 4))
 =>
   (assert (iAgree)))
```

Figure 10-7. Keyword rule for agreement (simplified)

Combination rules, such as the Praise_you_are_positive rule in Figure 10-3, combine intermediate positional facts, and impose constraints on the relative positions of these facts, to recognize discourse acts.

Templates And Ungrammatical Inputs

Template rules tend to be promiscuous, mapping a large number of ungrammatical inputs to discourse acts. For example, the rule in Figure 10-4, besides mapping sentences such as "I want a glass of your best Bordeaux", or "I cry out for scotch" to a drink request, will also map "I monitor want table fruitcake whiskey" to a drink request, resulting in Trip saying "Coming right up" and heading to the bar. Trip responding to such a sentence without batting an eye is potentially a believability problem. An author may thus try to tighten up the template to accept only the grammatically correct ways, that follow this general form, of requesting a drink (there are certainly other ways of requesting a drink that don't start with "I", and thus need a new template). Imagine removing the "*" from between "I" and "want" in the template in order to eliminate the possibility of spurious words occurring between them. But now "I really want whiskey" is not matched. But if the rule is fixed up to handle an optional "really", it still doesn't handle "I think I want a glass of whiskey". And if some local template patch is included to accept this case, it still doesn't handle "I sure want a glass of whiskey". And so on… The pattern is clear; given a template rule that tries to tightly match a set of grammatically correct utterances, one can always invent cases the rule should match yet doesn't. Loosening the template rules to accept a wider range of grammatically correct utterances (with the same number of rules) inevitably allows ungrammatical utterances to also match. We seem to be in a dilemma –tight matching introduces the believability problem of the characters saying "huh?" to utterances it seems like they should understand (if a character understands "I want a whiskey", the character better understand "I really want a whiskey"), while loose matching introduces the believability problem of the characters responding to radically ungrammatical utterances without batting an eye.

Given this choice, the template rules for *Façade* err on the side of being overly permissive. This is based on the design approach that it is more interesting for the characters to eke some meaning out of a broad set of utterances, and thus have some interesting response for this broad set, than to only have interesting responses for a narrow set, and respond with some form of "huh?" to the rest. And constructing the radically ungrammatical utterances really requires knowing something about the templates and using this knowledge to purposely construct sentences that exploit this template structure. For example, it is difficult to even think of typing "I monitor want table fruitcake whiskey" unless one suspects there is a template of the form (I * want *

whiskey) and is purposely creating a sentence to test this hypothesis. While players will sometimes try to break NLP by seeing what kinds of ludicrous sentences they can get the characters to respond to, the templates are designed not to robustly support this meta-activity, but rather to extract meaning from a broad collection of "natural" utterances likely to arise during the course of playing *Façade*.

Phase II: Discourse Acts to Reactions

Phase II of the NLP system is responsible for reaction selection, choosing one or more character reactions to the discourse acts recognized in phase I. In those cases where phase I generates multiple discourse acts for ambiguous surface text, phase II is responsible for deciding which discourse act to respond to, or, if responding to multiple discourse acts, choosing responses that work together. The details of the performance of the chosen reactions are managed by ABL behaviors. As discussed in Chapter 6 on page 112, handlers trigger on reaction selection, and perform the necessary beat goal modifications to respond to the reaction.

Reaction Selection Architecture

The reaction selector provides a framework within which authors can specify reaction selection logic. The framework consists of a number of different components. Authors specify their particular reaction selection logic by authoring different components and appropriately activating and deactivating these components over the course of the player experience. The reaction selection architecture is first described abstractly, in an implementation-free manner, followed by a brief description of the *Façade* implementation of this architecture.

Reactions

Reactions are declarative representations of character reactions. They abstractly describe the reaction, including a simple declarative representation of the properties of the performance of the reaction (what the ABL behaviors will do if this reaction is selected).These reaction properties are used by the selection logic to decide which reaction to perform, as well as by the ABL handlers, which use the properties of a selected reaction to modulate the reaction behavior.

Different reaction types make use of different properties. In *Façade*, however, all reaction types are described using the minimum set of properties in Figure 10-8. The *responder* and *attitude* properties are examples of declarative representations of reaction performance. Together these two properties describe who the "main" character of the reaction behavior is (who is doing most of the talking) and the general tone or attitude of the responder during the reaction. The *is multi* and *priority* properties are examples of reaction properties internal to the selection architecture. While these two properties don't directly describe aspects of the performance of the reaction, they are used by the reaction selection logic to help choose a reaction. The *is multi* property divides reactions into two categories, *normal* reactions that can occur on their own and can not occur with other *normal* reactions, and short *multi* reactions that can occur in tandem with a *normal* reaction. The specifics of how multi reactions are used in *Façade* is described on page 217. *Priority* represents the importance of the reaction vis-à-vis other reactions. The *available* flag indicates whether the reaction has previously occurred or not. When a

selected reaction is actually performed, the performance behavior marks the reaction as no longer available. Since all dialog in *Façade* is pre-recorded, it is important to *not* reuse reactions to prevent the characters from seeming robotic – this is typically only an issue for the global reactions, since they are available across almost all the beats. Proposers can test the *available* flag in order to avoid proposing a reaction that has already occurred.

1. **Reaction type** – The reaction type (e.g. a global satellite topic mix-in or a beat-specific transition-out).
2. **Priority –** The importance of this reaction vis-à-vis other reactions.
3. **Responder –** The primary character in the reaction.
4. **Attitude –** The responder's reaction attitude (positive, negative or neutral).
5. **Is Multi –** A boolean property; true if the reaction is a *multi* reaction that can occur with other reactions, false otherwise.
6. **Available –** a boolean flag indicating whether this reaction is still available.
7. **DA –** The discourse act that was most proximally responsible for activating this reaction.

Figure 10-8. Standard reaction properties in *Façade*

Contexts

Reactions to discourse acts are proposed within discourse *contexts*. Associated with every context is a collection of reaction proposers, which propose reactions to discourse acts, and a context priority, which is used to choose the context from which a final reaction will be chosen.

The contexts used in *Façade* are shown in Figure 10-9 in *increasing* context priority.

1. **Global** – The most general context; it is typically active across all beats.
2. **Beat –** The context associated with a specific beat; proposes beat-specific reactions to discourse acts.
3. **Reaction –** A short-lived, high-priority context activated *after* reactions.

Figure 10-9. Contexts used in *Façade*

Contexts are activated and deactivated over time. The beat context changes with every beat, sometimes changing more frequently if there are different conversational contexts *within* the beat (e.g. a beat that poses a question to the player will have a different conversational context before and after the question). The short-lived, high-priority reaction contexts are used if there are likely player responses to the previous reaction that the system should be prepared to respond to for some short period of time. For example, suppose the player typed "I just got married", resulting in the recognition of a *refer to* discourse act to the satellite topic *marriage*. Assuming the current beat has no beat specific reaction to DAReferTo(eCharacter_none, eSatelliteTopic_marriage), a global satellite topic reaction may be chosen, resulting in Trip saying "You know, the funny thing I've noticed about marriage is that it's not for everybody" while Grace looks at him and frowns. For a short period of time after this reaction, an obvious player response is to agree (e.g. "Yeah, you're right") or disagree (e.g. "I beg to differ") with Trip. To handle these responses, the behavior performing the reaction would activate a short lived reaction context containing proposers that propose special reactions to *agree* and *disagree* discourse acts. In order to prevent this digression from the beat from turning

into an open ended conversation, these second order reactions always try to deflect the topic and return to the beat. For example, if the player did respond to the marriage satellite reaction, perhaps by disagreeing, in the second order reaction Grace would say "Enough talk about marriage – it's so depressing", and return immediately to the beat.

Reaction Proposers

Each context contains a number of *reaction proposers*. A reaction proposer proposes a reaction in response to a discourse act. It may additionally make use of other state in the system, such as the current tension level or affinity. Each proposer makes a local decision independently of the other proposers in the system.

Pseudo-code for an example proposer appears in Figure 10-10.

```
Let ?questionCharacter = the character who posed the question
Let ?nonQuestionCharacter = the character who didn't pose the question
Let ?da = the discourse act

if
    DAPraise(?questionCharacter) or
    DAPositiveExcl(?eitherCharacter) or
    DAExpress(?eitherCharacter, happy) or
    DAAlly(?questionCharacter) or
    DAFlirt(?questionCharacter) or
    DACriticize(?nonQuestionCharacter, light) or
    DANegativeExcl(?nonQuestionCharacter) or
    DAExpress(?nonQuestionCharacter, angry) or
    DAExpress(?nonQuestionCharacter, sad)
then propose
    AffinityGameReaction(type: TxnOut_Affinity, responder: ?questionCharacter,
      attitude: positive, priority: agreementPri, isMulti: false, DA: ?da)
```

Figure 10-10. Pseudo-code for an affinity game proposer

This proposer appears in the affinity game context, a context activated in beats that give the player a forced choice of siding with Trip or siding with Grace. For example, in the decorating beat Grace, in an unconscious attempt to reveal that her constant redecorating of the apartment is really an inadequate sublimation of her interest in art, may complain about her latest decorating scheme, ultimately posing a question to the player, "Don't you think this couch just doesn't work?", while Trip counters, "Grace, it looks fine!". At this point the affinity game question has been posed and the affinity game context is activated. In this context, most player utterances are interpreted as siding with Trip or siding with Grace. In the proposer pseudo-code in Figure 10-10, Grace is the ?questionCharacter (the character who posed the affinity game question) and Trip is the ?nonQuestionCharacter. In the event that the player directs one of a number of positive discourse acts towards the ?questionCharacter, or one of a number of negative discourse acts towards the ?nonQuestionCharacter, this proposer proposes a beat transition-out reaction (transition-outs are described on page 109) in which the ?questionCharacter gains affinity (or keeps it if he/she already has affinity). This is just one of several proposers in the affinity game context.

Note that the example proposer, besides conditioning on the current discourse act, also conditions on the identity of the character who posed the affinity game question. Proposers are free to make use of other state in addition to the recognized discourse act. If proposers in the currently active contexts depend on additional state, it is up to the ABL behaviors (and potentially, the beat manager) to maintain this state.

Reaction Priority Mappers

After the reaction proposers have finished their job, each discourse context contains a number of proposed reactions, each with a priority specified by their proposer. Since the reaction proposers operate independently of each other, the priorities associated with these reactions have been chosen independently of each other. Sometimes an author will want to conditionally modify these reaction priorities as a function of the other proposed reaction candidates. That is, the importance of proposed reactions may not be determinable locally, but may depend on what other options are currently active. *Reaction priority mappers* provide this functionality. Reaction priority mappers can be activated and deactivated at any time. Typically the logic of the mapper will be specific to a given beat or small collection of beats.

Context Priority Mappers

After all the proposers in every active context have had a chance to propose reactions, and all active reaction priority mappers have run, there is now a collection of priority-ordered discourse contexts within each of which is a collection of priority-ordered proposed reactions. Similar to reaction priority mapping, sometimes an author will want to conditionally modify the context priorities as a function of the reactions that have been proposed across all contexts. That is, the static priority associated with each context may not adequately characterize the importance of the context in all situations. *Context priority mappers* provide this functionality. Like reaction priority mappers, context priority mappers can be activated and deactivated at any time. Typically the logic of the mapper will be specific to a given beat or small collection of beats.

Reaction Selector

After all the reaction proposers and any active reaction and context priority mappers have had a chance to run, there exists a (potentially reordered) collection of contexts, within each of which is a (potentially reordered) collection of proposed reactions. It is the job of the *reaction selector* to examine this set of contexts and reactions and choose one or more reactions to actually perform. Like every other component in the reaction selection architecture, the reaction selector can be dynamically changed. If different parts of an interactive drama make use of radically different reaction types with which are associated radically different properties, then the reaction selector would have to change for these different parts. However, for *Façade*, a default selector has been written that makes use only of the properties in Figure 10-8. It is expected that this standard selector will suffice most of the time.

The default selector first chooses the highest priority normal (non-multi) reaction from the highest priority context. If the highest priority context contains multiple normal reactions that share the same highest priority, one is chosen at random. Then the selector chooses the highest priority *multi* reaction from the highest priority context, with the additional constraint that the multi reaction must either have both the same responder and attitude as the chosen normal reaction, or both a different responder and attitude from the chosen normal reaction. For example, if the responder of the chosen normal reaction is Grace, and her attitude in the reaction is positive, then the multi reaction must either have Grace as the responder with a positive attitude, or Trip as the responder with a negative attitude. Once the reaction selector has chosen either a single normal reaction (if no multi

reaction satisfying the constraints is available), or both a normal and multi reaction, reaction selection is done. At this point, the characters perform the selected reactions.

The reaction selection algorithm is summarized in Figure 10-11.

1. The reaction proposers within each active discourse context propose zero or more reaction candidates.
2. If a reaction priority mapper is active, the mapper reorders the reaction priorities within each context.
3. If a context priority mapper is active, the mapper reorders context priorities.
4. The active reaction selector selects one or more reactions. The standard selector chooses both:
    a. The highest priority normal reaction in the highest priority context
    b. The highest priority multi reaction (from the highest priority context) that also satisfies the responder and attitude constraints.

Figure 10-11. Reaction selection algorithm

Implementation and Integration

For *Façade*, the reaction selection framework is implemented in Jess. Proposers, mappers, and selectors are written as Jess rules. Contexts are implemented as Jess modules, where a module is a named collection of rules that can be activated and deactivated. A collection of control rules implements the algorithm in Figure 10-11.

As described on page 211, phase I processing results in the creation of discourse act WMEs in story memory. If the performance is currently interruptible (determined by the LetMeFinishOn and LetMeFinishOff cues as described on page 112), an ABL demon grabs the discourse act WMEs and passes them to phase II. If the performance is currently uninterruptable, the ABL demon waits to pass on the WMEs to phase II until the performance *is* interruptable. Once a reaction(s) has been selected, a reaction WME is created in story memory. Reaction WMEs trigger ABL handler behaviors (meta-behaviors), which modify beat goals to incorporate the reaction.

Jess, which is implemented in Java, provides a number of facilities for integrating Jess with other Java applications. All the components of Jess, including the rule engine itself, are Java classes. This makes it easy to embed Jess rule engines in Java programs and programmatically control the engine through the method interface. Two instances of the Jess rule engine, one for each phase of NLP, are stored in global variables within an ABL agent. The two rule engines can be thought of as special purpose decision making machinery within an ABL mind. ABL behaviors access the rule engines by calling methods on the engines in mental acts.

Additionally, the Jess language provides support for calling external Java methods. This makes it easy for Jess rules to create WMEs as a means for communicating back to the ABL behaviors. Jess also provides support for shadow facts, automatically updated Jess facts that mirror the information stored in Java objects (as long as the objects conform to the Java bean standard). This makes it easy for phase I and phase II rules to access any WME state needed during rule processing.

Global Mix-Ins

As an example of reaction selection logic, consider the global mix-ins. Global mix-in reactions respond to discourse acts that aren't handled within the current beat context.

These reactions are proposed within the global context, which is active almost all the time and generally has the lowest context priority. When responding to a discourse act, the specific mix-in performed often depends as well on the current affinity and tension (see the description of *Façade's* story values on page 158). This section provides a simplified description of the global mix-in reactions and the logic used to select these reactions. There are seven types of global mix-ins: object, satellite topic, discourse act, push-too-far, pattern, specific-deflect, and generic-deflect.

Object mix-ins react to references to objects. Each object is associated with a character; for example, Trip is associated with the bar, while Grace is associated with the paintings. In an object mix-in reaction, the associated character comments on the object, revealing a bit of back-story in the process. For example, imagine that during the "decorating" beat, the player moves up to the bar and looks at it for a moment. This results in a (DAReferTo character_none, object_bar) being generated (player physical actions can also generate discourse acts). There is no beat-specific meaning to references to the bar within the decorating beat, so no reactions are proposed within the beat-specific context. The general context would propose a bar object mix-in (and possibly other mix-ins). The performance of the bar mix-in appears in Figure 10-12.

Trip: (smiling) You know, this bar was a gift from Grace's father.. he's quite a cocktail-meister I can tell you that.
Grace: (sigh)

Figure 10-12. The tension 1 *bar* object mix-in

Satellite topics are conversation topics relevant to *Façade's* domain (e.g. marriage, divorce, infidelity, career), but unlike story topics, are not central to the story progression. Recall that story topics (page 156) are integral to the story progression, and are thus the framework around which beats are defined. *Satellite topic* mix-ins react to references to the various satellite topics. For example, if the player referred to the satellite topic *infidelity*, then an infidelity mix-in would be proposed. The performance of the neutral and Grace affinity infidelity mix-in appears in Figure 10-13.

Grace: I recently read that in Europe, where affairs are supposed to be, you know, accepted....
Grace: (looks at Trip) ... that the women there actually don't accept it.
Grace: (looks at Player) They don't like it any more than we do here.

Figure 10-13. The tension 1, neutral and Grace affinity, *infidelity* satellite mix-in

Discourse act mix-ins react to discourse acts not handled in the beat context. For example, if a beat context has no reaction for a praise discourse act directed at Grace, and Grace currently has affinity with the player, then the *praise Grace* reaction whose performance appears in Figure 10-14 is proposed.

Grace: (smiling) Oh! well... thanks…
Trip: I think you're trying to get on Grace's good side tonight.
(pause)
Trip: (clear throat)

Figure 10-14. The tension 1, Grace affinity, *praise Grace* discourse act mix-in

Push-too-far reactions happen if the player "harps" on a specific topic too much by bringing up the topic several times in a short period of time. Each of the objects and satellite topics is associated with one of the four story topics: *rocky marriage*, *artist/advertising*, *Trip's affairs*, and *façade*. Discourse acts are grouped into more general, abstract acts, such as praise-ally-Trip, criticize-oppose-Trip, etc. If too many indirect references to a story topic or abstract discourse act occur within some period of time, a push-too-far reaction is proposed. In a push-too-far reaction, the "bugged" character (the character for whom the topic or abstract discourse act is a hot button) responds negatively. For example, suppose that the player refers to the wedding picture twice in a row within a beat that has no beat-specific reaction to the wedding picture references. The first picture reference causes an object reaction. The same object reaction can't be used again to respond to the next picture reference because it's already been used up at this tension level and affinity. But the wedding picture is associated with the story topic *rocky marriage*. If there have already been several indirect references to *rocky marriage*, then the second wedding picture reference will cause a push-too-far for the *rocky marriage* topic to be proposed. *Rocky marriage* is a hot button for Trip. The performance of the *rocky marriage* push-too-far reaction appears in Figure 10-15.

Trip: (annoyed) Goddamn, what you are you hinting at? I'm getting really sick of it.
Grace: Trip! What are you –
Trip: Look I'll leave you two to keep chatting about love and marriage and all that good shit.
(Trip heads for the kitchen)
Grace: What? Where are you going?
Trip: (walking away) I'm just going to get something from the kitchen.

Figure 10-15. Tension 1 *rocky marriage* push-too-far mix-in

Pattern reactions are currently the only *multi*-reaction. They respond to patterns of interaction, such as the player being quiet, or the player making lots of negative or positive comments (i.e. typing utterances that are recognized as "negative" discourse acts such as *oppose* or *criticize*, or "positive" discourse acts such as *praise* or *flirt*). Once a discourse pattern has been recognized, a pattern reaction is proposed in which the character who currently *doesn't* have affinity is the responder. The pattern reactions all have a negative tone; the character lacking affinity criticizes the player for exhibiting the pattern. If the pattern reaction has a consistent tone with the selected normal (non-multi) reaction, then both the pattern (multi) reaction and the normal reaction are selected. For example, suppose the player praises Trip during a beat in which Trip already has affinity. If the current beat is an affinity game and the question has already been posed, the beat context will propose a transition-out in which Trip maintains affinity. But if the number of positive comments made by the player has passed the pattern threshold, then the global context will also propose a pattern reaction (among other reactions) in which Grace

comments negatively on the pattern. Since, in this case, the responders and attitudes of the beat reaction and the pattern reaction are consistent (Trip is positive, Grace is negative), both reactions will be selected and performed. Grace will first say "(mildly sarcastic) My, aren't you *bubbly* tonight", followed by the beat reaction.

Like push-too-far, *specific-deflect* mix-ins are also associated with the story topics and abstract discourse-acts. Specific-deflects respond to multiple (indirect) references to the same topic (or multiple occurrences of the same abstract discourse act) in those cases where the most specific reaction is not available (perhaps has been already used up) and push-too-far has not yet been triggered. For example, suppose that the player flirts with Grace twice in a row in a beat that has no beat-specific response for flirt. The first flirt causes a discourse act reaction. The same discourse act reaction can't respond to the next flirt because it's already been used up at this tension level and affinity. Assuming that there haven't been enough other discourse acts to trigger a push-too-far reaction, the *flirt Grace* specific-deflect would be proposed. The performance of the Grace affinity version of the *flirt Grace* deflect appears in Figure 10-16.

Grace: (surprised look, smiling)
Trip: (annoyed) Okay, c'mon, that's enough of that. It was kind of funny the first time, but –
Grace: It's alright Trip. [He's|She's] just having a little fun, that's all.
Trip: (pause) (act like nothing's wrong) Yeah... okay... (strained smile)

Figure 10-16. The tension 1, Grace affinity, *flirt Grace* specific deflect

Generic-deflect reactions are chosen if no better reaction is available; they have the lowest reaction priority in the general context. Generic-deflect reactions acknowledge that the player said something, but try to blow it off and move on with the story. Two example generic-deflects are "Uh huh… Anyway…" and "Yeah… As I was saying…"

The global reactions always try to reveal something interesting about the backstory and about the characters' psychology. During a beat, the full performance a player experiences consists of both material from within the beat and global reactions to player interaction not handled within the beat. As far as the player is concerned, it's all part of the story. We purposely reserve some of the story content for the global mix-ins; we want to reward the player for interacting, not punish her for momentarily stepping outside of the beat.

Global mix-ins can have side effects, such as changing the affinity or influencing beat sequencing. In the example rocky marriage push-too-far mix-in above, the affinity potentially changes (if Trip had affinity with the player, he loses it) and he temporarily leaves the room. This would cause the current beat to abort and influence the beat manager to select a beat in which only Grace is present.

All of the global reactions, except for generic-deflect reactions, can only be used once. The reactions (declarative representations of reactions, see Figure 10-8) keep track of whether they have been used or not. Global proposers only propose reactions that are still available. Since generic-deflect is the backstop, the reaction of last resort if the characters have nothing better to say, it is always available. The are a number of alternative performance behaviors for the generic-deflect reaction.

Selection Examples

This section provides a couple of examples of how the phase I and phase II pipelines work together to propose a reaction to a player utterance. For these examples, imagine that an affinity game beat, such as the decorating beat, is happening.

In the first example, the player types "I don't think so" before the beat gist. The template system recognizes this as a (DADisagree none). Since the beat gist has not occurred, the affinity game question hasn't been posed yet; Grace has started complaining about her decorating (e.g. "This new decorating scheme just doesn't work"), and Trip has started demurring (e.g. "Oh Grace"), but Trip and Grace haven't yet asked the player to take a side. So the proposers in the beat context can't yet propose a beat transition-out (the proposer rules have to test whether the beat gist has happened). But the decorating beat does have a custom before-beat-gist reaction for disagree, in which Trip says "See Grace, everybody thinks it looks fine". The reactions proposed in the two active contexts appear in Figure 10-17. Since no priority mappers are active, and no multi reactions have been proposed, the highest priority reaction from the highest priority context is selected, in this case the beat-specific *disagree* reaction proposed in the beat context.

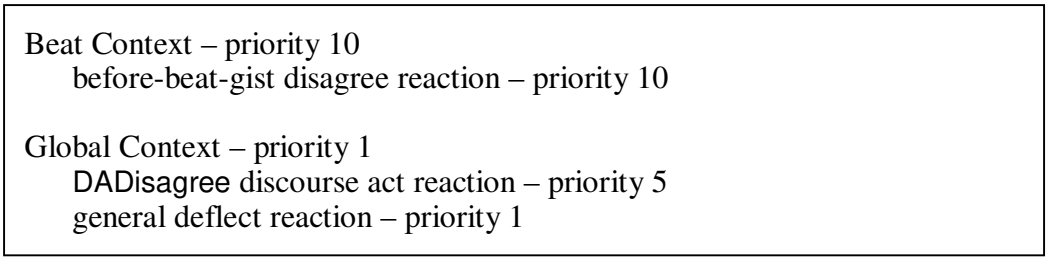

Figure 10-17. Proposed reactions to DADisagree

Now imagine the same situation, except that the player has typed several negative utterances (e.g. disagree, criticize, negative exclamation), over the last couple of minutes. In this case, the proposed reactions to the DADisagree appear in Figure 10-18.

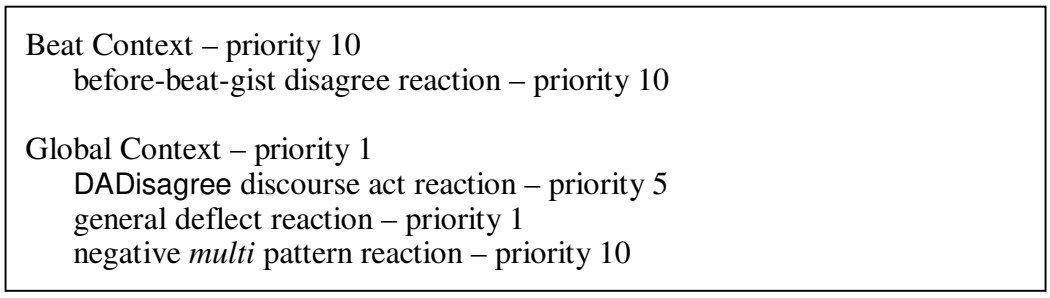

Figure 10-18. Proposed reactions to DADisagree after multiple negative utterances

A *multi* pattern reaction to the pattern of "negative" utterances is additionally proposed in the global context. If Grace currently has the affinity, then the pattern responder is Trip; if Trip currently has the affinity, then the pattern responder is Grace. If the pattern responder is Trip, then the pattern reaction won't be chosen, since its attitude and responder (responder: Trip, attitude: negative) is inconsistent with the attitude and responder of the beat-specific disagree (responder: Trip, attitude: positive). If the pattern

responder is Grace, then the attitude and responder of the beat-specific disagree and pattern reaction are consistent; both will be selected and performed.

Related Work

Our approach in *Façade* is to view the natural language understanding problem as a dialog management problem. Dialog management focuses on the pragmatic effects of language (what a language utterance does to the world) rather than on the syntax (the form of surface text) or semantics (meaning) of language. Dialog management views a language use situation as consisting of a discourse context within which conversants exchange speech acts. AI system-building work in dialog management focuses on the design of architectures that handle the domain-independent discourse bookkeeping while appropriately applying domain-dependent knowledge. While these systems tend to deeply model a narrow discourse domain (usually centered around the completion of some task), their architectural lessons provided useful guidance for thinking about dialog management in *Façade*.

Collagen [Rich & Sidner 1998; Lesh, Rich & Sidner 1999] is an application-independent framework for maintaining discourse state in collaborative user-agent interaction. SharedPlans are used to represent the shared problem-solving process. Discourse state consists of an active plan tree, focus stack, and history list. Non-monotonic modification of the active plan tree is handled via a truth-maintenance system. The focus stack contains currently open discourse segments (an "open" segment can have acts added to it). The history list contains top-level segments that have been popped off the focus stack. Usually the root of the plan tree is the purpose of the base segment of the focus stack and subsegments correspond to subtrees (except for interruptions). The partial SharedPlans associated with discourse segments are modified by discourse acts. Discourse acts are selected by the user through a menu system rather than being recognized from surface text.

The TRAINS series of systems [Allen et. al. 1995; Ferguson et. al. 1996] are mixed-initiative, conversationally proficient planning assistants that work with a user to manage a railway transportation system. A parser accepts natural language (the later systems use speech recognition) and parses directly to a set of candidate speech acts based on the surface features of the utterance. The discourse state is maintained in a tree of discourse representation structures, each of which consists of first-order predicate assertions of the shared facts of the conversation plus a list of potential referents for resolution. After de-indexing (reference resolution), the candidate speech acts go to the dialog manager, which disambiguates and potentially modifies the candidate speech acts as a function of the current conversational state. The state is then updated, including the maintenance of conversational obligations (e.g. the obligation to acknowledge a request). The rest of the system consists of the domain planning system that modifies the jointly constructed domain plan, and the execution planner that decides what utterances to make to the user. [Allen et. al. 1998] presents a general architecture for dialog based on architectural lessons learned from the TRAINS systems.

Traum [Traum 1999a; Traum 1999b] and Traum and Hinkelman [Traum & Hinkelman 1992] discuss general issues in the use of the speech act framework in conversation systems. For *Façade*, the most interesting aspect of the discussion is the generalization of the notion of speech acts to hierarchical discourse acts, which range in

granularity from grounding acts, through the classical speech acts, and up to argumentation acts, conversation threads and discourse scripts.

Future Work

The following list details the natural language processing issues that are currently being avoided in *Façade*.

- No deep natural language generation (e.g. [Callaway & Lester 1995; Kantrowitz & Bates 1992]). All character dialog is pre-written.
- No robust parsing of the surface text (e.g. [Lavie & Tomita 1996; Sleator & Temperley 1993]). Surface text is directly mapped to semantic and pragmatic representations by template-based techniques.
- No deep natural language understanding. This includes techniques based on the internal state of the agent, and techniques that trace the complex web of common-sense inferences implied by deep understanding. Instead, short chains of context-specific rules move from surface-text towards context-specific discourse acts.
- No multi-modal, sub-utterance discourse acts. This includes multi-modal understanding based on fusing eye-movement, body-movement and speech information (e.g. [Thorison 1996]), and the use of backchannel utterances such as "uh-huh", "hmm…" etc. to provide feedback during utterances. The interface prevents the player from engaging in complex body movements while simultaneously talking (typing).
- No multi-modal generation coordinating body movement, eye movement, speech, etc. (e.g. [Lester et. al. 1999; Loyall & Bates 1997; Thorison 1996; Cassell et. al. 1999]). Rather, for each utterance, hand-authored performance behaviors produce the physical performance that goes with the utterance.

The bottom line is that in handling the combinatorics of natural language conversation, much of the combinatorics is pushed on the authors rather than handled by architectural mechanisms. Any of these research areas are possible areas for future work.

CHAPTER 11
CONCLUSION

In this chapter, I conclude with a brief discussion of the major contributions and findings of the work presented in this dissertation. More specific related and future work discussions appear at the ends of Chapter 5 (believable agents), Chapter 8 (drama management), and Chapter 10 (natural language processing).

Interactive Drama

Here I summarize the major contributions and findings made during the construction of *Façade*.

The Dramatic Beat as an Architectural Entity

In the theory of dramatic writing, the *beat* is the smallest story unit, the smallest unit of character performance that changes the story. Within-beat character performance may convey character traits; for example, a line of dialog may convey what the character is thinking, or the body language while walking may convey that the character is impatient. But this within-beat behavior does not on its own cause a story change, does not, for example, increase the conflict between two characters, or convey a changing emotional dynamic between two character. It is only the performance of a whole beat, which may often involve the coordinated actions of multiple characters, that affects a story change.

The *Façade* architecture operationalizes the dramatic beat. As an architectural entity, beats simultaneously represent the story-level knowledge necessary to decide when a beat should be included in a story, and, in the event that the beat is included in a story, the character-level knowledge necessary to perform the beat and respond to within-beat player interaction. The *Façade* architecture provides one implementation of this idea, using ABL behaviors to represent the character performance knowledge (the ABL language is described in Chapter 5, ABL idioms for authoring beat behaviors are described in Chapter 6), and the beat description language described in Chapter 8 to represent the story sequencing knowledge. Independent of this particular implementation, the key contribution of architectural support for beats is the idea of a representational level that describes coordinated character activity along with the story-level effect of this activity.

Joint Goals and Behaviors

Beats often communicate their story-level effect via the coordinated activity of multiple characters. Agent languages are typically organized around strong autonomy, that is, around the idea that agents independently choose actions based on their private internal state plus the state of the world. Any coordination happens via sensing or via ad hoc manipulation of internal state (e.g. characters reading and writing each other's minds). Unfortunately, both of these approaches are susceptible to coordination failure – sensor-based coordination requires arbitrary plan recognition, while ad hoc state manipulation requires the character author to resolve the coordination problem for each coordination

instance. As described in Chapter 5, coordinating reactive planning agents is difficult – many non-intuitive potential race-conditions arise that, if not handled properly, can result in coordination failures. Ideally, the character author should not have to continuously resolve the coordination problem. The approach taken in ABL, the believable agent language developed for this thesis, is to provide language support for multi-agent coordination in the form of joint goals and behaviors. When the character author wants to coordinate multiple characters, she can just make use of the coordination support built into the language.

A possible objection to pushing coordination into a believable agent language is that coordination should be personality-specific; by providing a generic coordination solution in the language, the language is taking away this degree of authorial control. However, the joint goal mechanism places no limits on the apparent coordinated activity (or lack thereof) of a collection of characters. Joint goals provide a mechanism for guaranteeing synchronized entry into and exit from a goal (identified by its signature) across a team of agents. A joint goal may correspond to an explicit character goal (a character goal that should be apparent to an audience) or to a lower level, "unconscious" goal. Thus, two characters that are having a violent disagreement may coordinate their fight with joint goals, even though the characters don't have "conscious" character goals to coordinate. Or, two characters who are humorously failing to coordinate, such as Laurel and Hardy trying to set up a ladder to climb in a window, may tightly coordinate their apparent coordination failures with joint goals. Though a humorous coordination failure is being communicated to the audience, the fine details of the coordination failure are themselves under explicit authorial control and are tightly synchronized. Joint goals provide explicit authorial control over teams of characters, supporting a variably coupled, tunable degree of connection across the team, from the extreme of the characters forming a single mind (if every goal is joint), to the extreme of a collection of completely independent minds (if the keyword `joint` is never used). Rather than taking control away from an author, joint goals and behaviors provide *more* authorial control; instead of the degree of independence of a collection of agents (the degree of one-mind or many-minds) being fixed by the architecture, the author can tune the degree of independence of the team.

Story Design

In the process of building *Façade*, we have explored the issues of story design in a beat-based architecture. As described in Chapter 2 and the first part of Chapter 8, our goal is to create an interactive drama that provides both local agency (i.e. immediate and rewarding response to local player interaction, such as the player saying something or manipulating an object) and global agency (i.e. player activity over time influences the longer term, large scale story structure). Within the *Façade* architecture, global agency is provided both by beat sequencing and by special "summary" beats that summarize the history of player interaction (the part I and part II mini-climaxes plus the final story climax). The beat manager is responsible for maintaining the *tension* story value, sequencing beats in such a way as to approximate the desired tension arc. Local agency is handled *within* each beat. Each beat is a little interaction game, setting up a context in which player interaction is mapped to changes in the *affinity* story value. Additionally, the player learns things about Trip, Grace, and their relationship through the dialog performed in each beat. In general, each beat doesn't incorporate every possible discourse act into the beat-specific interaction game. Global content, in the form of global mix-ins whose performance mixes in with the current beat, responds to player interaction that is not handled by a beat.

Authorial Burden

The authorial burden for authoring in the *Façade* architecture is high. Future research should focus on decreasing authorial effort in three places: beat authorship, performance behaviors, and natural language understanding rules (template rules).

In order to manage the large scale structure of the story, the beat manager must have a rich collection of beats at its disposal. The more beats that are available for different situations, the more fine-grained control the beat manager can provide. In *Façade*, we found that the beat management bottleneck was not in the sophistication of the beat management logic, but rather in authoring enough beats to give the beat manager something to do. Future work should focus not so much on drama management (the logic for sequencing pre-written pieces), but rather on story generation (generating story pieces). What is needed is a framework within which story pieces (e.g. beats) are generated as they are needed by the drama manager. The trick of course is to maintain the potential for rich authorial control, providing multiple levels at which the author can customize the story piece generation process.

The beat performance behaviors (described in Chapter 6) perform individual lines of dialog or responses to dialog lines, providing fine-grained control over such details as when emphasis gestures are used, on what individual word(s) of the line reactions such as a frown or smile occur, and the longer-term reactions (e.g. emotional reactions) to the line. Currently these performance behaviors have to be hand-authored for every possible line of dialog that can be spoken. Since there is no representation of the content of the lines of dialog, there is no way for a more automated process to generate the dialog line performance and reactions. This is a highly non-trivial problem. Given the breadth and depth of the content of *Façade* (e.g. romantic relationships, family, a variety of head games), there are no representations to even formally represent such content, let alone theories that, based on the content plus the details of a personality, decide how the line should be performed or what the reaction to hearing the line should be. For the short term (say, next five years), the only approach I can see is to work with simpler story domains, and explore shallow representations that support author-customizable, semi-automatic performance processes. Note that these performance processes could be implemented as ABL behaviors (a rich goal hierarchy that reasons about the dialog content to produce performance) or could be, to some degree, pushed into the beat generator, where the generator performs the reasoning about the content and generates “unwound” ABL code (similar to the code that is currently hand generated) to hand off to the characters.

Finally, template-based natural language understanding predictably requires a large authoring investment. There is no magic bullet here – the only option is to chip away at the problem, exploring incrementally deeper approaches.

Expressive AI

Here I summarize the major positions of Expressive AI, my art and research practice.

Critiques of AI

Expressive AI is not allied with any one critical position regarding AI. Rather, the full terrain of AI system-building practice is taken as a conceptual resource. In my own work, I have made use of AI techniques from a number of distinct AI traditions, including symbolic AI (*Terminal Time*, see Appendix C), machine learning (*Office Plant #1*, see

Appendix B), and reactive-planning (*Façade* and *Subjective Avatars*, see Appendix A). None of the various AI traditions is viewed as being peculiarly suited for cultural production. Rather, each tradition offers a way of thinking about computation and its relationship to human activity.

Various critiques of AI, such as the embodied action critique of symbolic AI (see Chapter 4), make explicit some of the implicit philosophical assumptions underlying an AI technique. For Expressive AI, such analyses are useful for making explicit the relationship between ways of *talking* about an AI technique, and the *computational manipulations* enabled by a technique. But the critiques are not seen as categorically ruling out an approach, or identifying the "correct" approach to use for art making.

Interpretive and Authorial Affordance

Expressive AI is concerned with the relationship between the interpretive and authorial affordances enabled by different AI architectures. Interpretive affordances are provided by the readable signs produced for an audience by an AI-based art piece. For example, the interpretive affordances of *Façade* are provided by the character's animated action, by the words they speak, and by the relationship between their actions and words and the player's activity (movement, object manipulation, and typed text).

The authorial affordance offered by an architecture depends on the relationship between the program code (code signs) and the audience output, and the relationship between the program code and the ways of talking about program code.

In the first case, an architecture provides good authorial affordance if it provides representations that allow the author to talk about the desired properties of audience signs, while simultaneously actually producing the desired signs as the representation is manipulated. For example, in *Façade*, the audience experience is thought of as divided into dramatic beats. Such an analysis could conceivably be entirely post-hoc – a story architecture may know nothing about beats, but the story produced by the architecture could be analyzed in terms of beats. The beats identified by this post-hoc analysis may often be somewhat large and loose, since the beats in this case are entirely an analytic category, without the story being generated (or managed) with beats in mind. In the case of *Façade*, we desire that the story be tightly organized, with as little extraneous activity (activity not contributing to the dramatic structure) as possible, while still remaining open to audience interaction. Thus a desired property of a story produced by the *Façade* architecture is that it have a tight beat organization – to offer a good authorial affordance with respect to this property, the architecture should offer a way of talking about beats. And it does – beats are an architectural entity. Character knowledge and story sequencing knowledge is explicitly organized around beats, and the structure of a story produced by a player interacting with the system is generated by beat processing.

But the correspondence between code-level structures and desired audience experience is not enough to create a good authorial affordance. Additionally, the code structures produced by an architecture must productively connect with a broader discourse. Every architecture is doubled, consisting of both the literal code machine, and a rhetorical machine that provides the lens, the way of talking and thinking, about the code machine. The discourse that surrounds the code machine is necessary for an AI architecture to function; the discursive interpretation allows raw, uninterpreted computation to be read in terms of human traits and properties, such as "goals", "behaviors", "knowledge", "action", "learning", etc. The rhetorical machine embeds the code machine in broader systems of human meaning. An Expressive AI practitioner consciously manipulates the rhetorical machine, searching for ways of talking about an

architecture that productively suggests idioms (ways of using the architecture) as well as incremental changes to the architecture. Of course, one can't talk about an architecture in any arbitrary way – the code machine and the rhetorical machine mutually constrain each other, and thus must be changed together.

As an example of this second component of authorial affordance, consider again beats in *Façade*. Purely at the code level, the collection of beats plus the beat manager provide a mechanism for sequencing beats and dynamically changing character behavior libraries as beats are sequenced. But how should beat behaviors be structured? By using the term "beat", one begins to tap into the rich meanings that surround the word "beat" in dramatic theory. In dramatic beats, story values change over the course of the beat. The story values change in response to the within-beat actions and reactions of characters. This suggests that specific sub-beat actions must be communicated to the audience in order to accomplish the value change. This, combined with the ways of talking about goals and behaviors in ABL, suggests the idea of beat goals – sub-beat units that must be accomplished in order for the beat to happen. Of course beats in an interactive drama must respond to player interaction – somehow player interaction must be incorporated into the default beat goal structure. In talking about ABL, the abstract behavior tree (ABT) at any point in time is said to represent the current intention structure of an agent, what it wants to accomplish, what it is currently "thinking" about. If, while a beat is active, an important part of the ABT is the current beat goals, this, combined with the idea of the ABT as a representational structure, suggests the dynamic manipulation of the beat goals within the ABT in response to player activity. This idea results in both incremental changes in ABL to support the necessary reflection[40], and the idiom of beat goals plus handlers for organizing beat behaviors. Note that if the beats in *Façade* had been called "story pieces" or "story units", and had thus not explicitly tapped into the ideas surrounding the dramatic beat, a very different chain of association may have been followed during the design of *Façade*, resulting in a different behavioral organization within the story pieces.

In Expressive AI, architectures are crafted just so as to provide the appropriate interpretive and authorial affordances. AI provides a rich collection of techniques (code + rhetoric) for interpreting computation in human terms. Paying explicit attention to the doubled nature of these techniques becomes a productive part of the design process.

Expressive AI *is* AI Research

Expressive AI is not the "mere" application of off-the-shelf AI techniques to art and entertainment applications. Rather, Expressive AI is a critical technical practice, a way of doing AI research that reflects on the foundations of AI and changes the way AI is done. AI has always been in the business of knowing-by-making, exploring what it means to be human by building systems. Expressive AI just makes this explicit, combining the thought experiments of the AI researcher with the conceptual and aesthetic experiments of the artist. The implicit discourse surrounding AI systems is made explicit, and thus becomes a resource for design and system building. Combining art and AI, both ways of knowing-by-making, opens up new research questions, provides a novel perspective on old questions, and enables new modes of artistic expression. The firm boundary normally separating "art" and "science" is blurred, not in the simple sense that they borrow from

[40] While reflection is a generic feature one considers when implementing any language (part of the way of talking and thinking about "computer language"), it was the idea of interaction handlers that motivated the concrete design of reflection in ABL.

the other (e.g. an AI researcher borrowing some good ideas from art), but in the more complex sense that AI and art becoming two components of a single, integrated practice.

APPENDIX A
SUBJECTIVE AVATARS

Introduction

The goal of the Oz project [Bates 1992] at CMU is to build dramatically interesting virtual worlds inhabited by believable agents – autonomous characters exhibiting rich personalities, emotions and social interactions. In many of these worlds, the player is herself a character in the story, experiencing the world from a first person perspective. Typically, the player's representation within the world – her avatar – is passive. The avatar performs actions as fully specified by the player, and reports events (reporting events can mean rendering a 3D scene or generating descriptive text) in a pseudo-objective manner (*pseudo*-objective because any description encodes the bias of the world author). This paper describes an alternative: a subjective avatar with autonomous interpretations of the world. This appendix first appeared in [Mateas 1997; Mateas 1998].

Why Subjective Avatars?

I want the user to step into the shoes of a character, experiencing a story from an alien perspective. In this manner the user gains an empathic understanding of a character by *being* this character. In non-interactive drama (movies, theater), an audience is able to gain insights into the subjective experience of characters precisely because the experience is non-interactive; the characters in the drama make decisions different than those that audience members might make. For example, consider the movie The Remains of the Day:

> A rule bound head butler's world of manners and decorum in the household he maintains is tested by the arrival of a housekeeper who falls in love with him in post-W.W.I Britain. The possibility of romance and his master's cultivation of ties with the Nazis challenge his carefully maintained veneer of servitude. [Loh 1993]

As an audience, we are able to gain empathy (subjective insight) into the butler because we can watch the butler make decisions within the context of his personal history and social role. However, in a Remains of the Day interactive world, how would a user know how to act like this W.W.I era English butler? What is to keep the user from immediately standing up to the master, thus derailing the story and preventing any empathic insight into this situation? The hope is that if the user's avatar filters and interprets the world in a manner consistent with this W.W.I era butler, the user will begin to feel like this butler, gaining a deeper understanding of the message the author wants to convey.

In addition, if the user really begins to *feel* like this butler, she will implicitly constrain her actions; for example, she won't punch the master in the nose. Such constraints are important: no matter how sophisticated the characters and plot control system might be in some story world, the user must be willing to "play along" in order to

have a high quality story experience (this is the equivalent of suspending disbelief in a play or movie). However, such manipulations of the user's freedom must remain implicit in order to maintain the special vividness interaction lends to a story [Kelso, Weyhrauch & Bates 1993].

Once the avatar is actively interpreting the world, the possibility is open for employing abstract, symbolic or surreal descriptions. Some of these possibilities are explored in [Smith & Bates 1989]. Thus, in addition to improving the user's experience, subjective avatars open up new artistic possibilities in the construction of dramatic worlds.

Fastfood World

I'm currently experimenting with subjective avatars within the Oz text-based system. The specific world I've built within this framework is *Fastfood World*, a McJob [Howe & Strauss 1993] from hell scenario set in a fastfood restaurant.

The Framework

The text-based system provides a framework for building worlds that are rendered in a manner similar to text-based adventure games or multi-user dungeons. This framework includes a non -real-time version of Tok, the Oz agent architecture [Bates, Loyall & Reilly 1992a], an object oriented physical world ontology for building the physical world simulation, a sense ontology for modeling the propagation of sensory information (sensory information can include descriptions of objects, actions and events), and a natural language generator for rendering into text descriptions of objects, actions and events [Kantrowitz & Bates 1992]. Note: the text spoken by characters is canned; it is not generated from semantic primitives.

The Characters

In *Fastfood World*, the user plays a young worker in his early 20s. The worker is frustrated by the dead-end job he finds himself in. He is interested in richly exploring life and thinking about the Big Questions but feels dragged down by his current job, which leaves him too depressed and apathetic to passionately explore his own interests.

Just a quick note on terminology: for the rest of this paper I will refer to the character of the young worker as the player. The player is the user in the role of the young worker. I will also use the masculine pronoun for this character, since, though the user may be of either gender, the role being played is a masculine role.

The player's nemesis is the manager, Barry. Barry, now in his early 30s, has been working at the restaurant since he was 16. He's proud of having worked his way up from clerk to assistant manager. He uses every interaction to assert his superiority over the player. He yells at the player in front of customers, gives long-winded explanations to simple questions, and reminds the player of past screw-ups. A common theme in the harangues he directs at the player is the need to stop dreaming, to stop thinking about the big picture and pay attention to details.

The player also interacts with a customer named Celia. Celia is a garrulous old lady who is more interested in social interaction than in ordering her food. She asks the worker questions about the food (e.g. "How greasy are your fries?"), and tends to break off into reminiscences. A common theme in her conversation is how the small concerns of day-

to-day life tend to fade away as one grows older; looking back on her long life, what was important (and remembered) are themes, feelings, relationships.

The Story

The player's experience in this world is divided into three parts. First, the player comes into work late. He's concerned about being caught. If he can only get to the register and sign in before Barry sees him, he might be OK. At this point, Barry is wandering about the restaurant, glancing at his watch and looking for the worker. If the player starts working on the register before Barry chews him out, Barry makes a mental note to chew the player out later. If Barry catches the player before he's started working, Barry yells at him for being late.

Second, after the player has either escaped being yelled at or not, the player will try to sign into the register. To his horror, the player discovers that he is not able to sign into the register because a previous worker left an incomplete transaction on it. The only person who can clear the register is Barry. The player knows, however, that if he asks Barry, Barry will use this as an opportunity to assert his superiority over the player. If the player continues trying to sign into the register and Barry notices this, he chews the player out for not asking for help (e.g. "we've all got to be a team"), and then "helps" the player. If the player asks for help, Barry chews the player out for not having learned to operate the register yet (e.g. "how many times do I have to show you?") and then "helps" the player. Barry's help consists of a long winded explanation of how you sign into the register (even though the player already knows how to sign in and exactly what the problem is) interspersed with descriptions of Barry's own rise to the glory of assistant manager. It is during this interaction that Barry stresses the importance of focusing on details (a petty focus on details), thus revealing one pole of the tension the player will have to resolve. Eventually Barry clears the register transaction. Incidentally, there is no way for the player to clear the register themselves. Interactive drama is not about solving puzzles; it is about relationships with characters. In this particular case, the player is in a "damned if you do, damned if you don't" situation in which there is no way to avoid having an unpleasant interaction with Barry.

Finally, Celia comes in to order food. The player first perceives her as a batty old woman. Celia seems disorganized and confused. She's more interested in conversation than in ordering food. During her conversation she begins revealing how she's forgotten many of the mundane details in her life; what remains as important are themes, feelings, and relationships. She thus reveals the second pole of the tension facing the player (petty details vs. meaningful experience). During this time, Barry comes out of his office repeatedly, admonishing the player to keep the line moving. If the player succumbs to his fear of Barry and the general bad mood that his job puts him in, the player will continue to perceive Celia as batty and irritating and Barry as dominant and scary. Eventually Celia will finish ordering and the story experience ends (petty details won). However, there is an opportunity for a transformative experience. If the player begins interacting with Celia more deeply, he begins perceiving her as first interesting, and eventually, wise. The player begins perceiving Barry as less scary, and eventually sad and pathetic. The story experience ends with Barry's psychological hold on the player broken (meaningful experience won).

Subjective State

Now that I've described the general motivation for subjective avatars and the specific story within which I'm exploring the concept, it's time to describe how a subjective interpretation of the world can be implemented.

In order for an avatar to provide a subjective interpretation for the player, it must both respond to events in the world by maintaining some sort of state and communicate this state to the player. First I'll describe the mechanisms by which the avatar maintains subjective state. Currently, the avatar's subjective state consists of emotional state (emotional responses to events) and story context.

Emotional State

To maintain emotional state, I make use of Em, the Oz model of emotion [Neal Reilly 1996]. In Em, emotions are generated primarily in response to goal processing events and attitudes. To enable the generation of goal processing emotions, the author of an agent provides annotations for agent's goals such as importance, likelihood-of-success, likelihood-of-failure, failure-responsibility, etc. These annotations allow Em to generate emotions as goals are created, as beliefs change about the likelihood of goals succeeding or failing, and as goals actually succeed or fail. For example, if a goal with non-zero importance fails, distress will be generated. If the goal was annotated with a failure-responsibility function, this function will be called to infer the cause of the failure; anger towards the agent who caused the failure will be generated. If, prior to the failure, there was a non-zero belief that the goal would succeed, disappointment will also be generated. In general, at any given moment, an agent's emotional state will contain several emotions with non-zero values. Over time, Em decays emotions.

In order for the avatar to have goal processing emotions, it must be processing some goals. But, since the avatar never engages in autonomous behaviors that directly result in action (the avatar never moves it's own muscles), what kinds of goals can an avatar have? An avatar can have passive goals, that is, a goal for some event to happen in the world but for which you don't directly take action. Such goals passively wait for some event to occur in the world in order to succeed or fail.

As an example of goal-based emotion processing, consider the avatar goal to avoid being chewed out for being late. The avatar's internal representation of this goal appears in Figure A-1.

```
(with
   (ignore-failure 'except-em)
   (importance 4)
   (compute-lof (lambda () (player-be-chewed-out-lof)))
   (failure-responsibility (lambda () (list $$barry)))
   (subgoal avoid-being-chewed-out-for-late))
```

Figure A-1. Emotion annotations for goal `avoid-being-chewed-out-for-late`

Some behavior is calling the subgoal avoid-being-chewed-out-for-late. If it fails, emotions are generated but the failure does not cause the enclosing behavior to fail. The importance of this goal is 4 on a scale of 1 to 10. Some function called player-be-chewed-

out-lof determines the avatar's belief in the likelihood of failure of this goal. If the goal fails, the agent responsible is Barry.

The avatar's internal representation of the behavior for avoid-being-chewed-out-for-late appears in Figure A-2.

```
(parallel-production avoid-being-chewed-out-for-late ()
   (subgoal succeed-in-avoiding-chewing-out-for-late)
   (with
     (priority-modifier 1)
     effect-only-no-fail
     (subgoal
      fail-in-avoiding-chewing-out-for-late)))
```

Figure A-2. Behavior avoid-being-chewed-out-for-late

Avoid-being-chewed-out-for-late simultaneously (in parallel) examines the world for evidence that the player got to work before Barry could chew him out and for evidence that Barry *is* chewing him out. Both the succeed-in... and fail-in... subgoals are passive goals. They sit and wait for their appropriate conditions to be met. While those two goals are waiting, the avatar can pursue other goals. If the succeed-in... goal succeeds, then avoid-being-chewed-out-for-late succeeds. If the fail-in... subgoal fails, then avoid-being-chewed-out-for-late fails. In either case, the appropriate emotions (joy, or anger and distress) are generated.

In addition to goal processing emotions, the avatar makes use of attitude-based emotions. Attitudes record the fact that the avatar feels a certain way towards certain objects. Every time the avatar senses the appropriate object in its vicinity, an emotion is generated. For example, the avatar fears Barry with a strength of 3 (1 to 10 scale). Every time Barry is in the same location as the avatar, a fear emotion of strength 3 towards Barry is generated. This is in addition to any goal processing emotions the avatar might be having.

Story Context

In addition to emotion processing, the avatar keeps track of where it is in the story (or, at least, the avatar's conception of the story). This is done so as to organize the avatar's goals and simplify the writing of behaviors. At any given moment, there are a set of goals active in the avatar. Some of these are passive goals for emotion generation; others describe specific objects or events to the user or report the avatar's thoughts to the user (described below). The behaviors associated with these goals are watching for certain events or sequences of events to occur in the world. Depending on where the player is in the story experience, the same event may need to trigger different reactions. Alternatively, reacting to a given event may only be appropriate during certain parts of the story. In this case, even if the trigger event never occurs, the goal should be removed from the set of active goals when the story has passed a given point without requiring that the goal succeed or fail. Explicitly maintaining a story context pushes the context information into the set of active goals instead of requiring this information to be included in the test expression of every behavior. For example, the avatar's most abstract understanding of the structure of the story is represented by the behavior shown in Figure A-3.

```
(sequential-production a-day-in-the-store ()
   (subgoal come-into-work-late)
   (subgoal setup-register)
   (subgoal wait-on-customer))
```

Figure A-3. Behavior a-day-in-the-store

A day in the fastfood restaurant consists of three sequential segments: coming into work late, setting up the register, and waiting on a customer. The behaviors associated with each of these subgoals watch for indications that their context has passed (e.g. "I'm done coming into work late."). The goal then succeeds and a behavior associated with the next goal is chosen.

Figure A-4, showing the behavior come-into-work-late, illustrates how an active behavior defines a story context.

```
(parallel-production come-into-work-late ()
   (with
     (ignore-failure 'except-em)
     (importance 4)
     (compute-lof (lambda () (player-be-chewed-out-lof)))
     (failure-responsibility (lambda () (list $$barry)))
     (subgoal avoid-being-chewed-out-for-late))
   (with
     effect-only
     (subgoal think-about-being-late)))
```

Figure A-4. Behavior come-into-work-late

The first subgoal, avoid-being-chewed-out-for-late was described above. It's job is to generate either positive or negative emotions depending on whether the player is chewed out or not. The second subgoal generates a stream of thought that is reported to the user under appropriate circumstances (described below). The main point here is that this particular opportunity for emotion generation and this particular stream of thought are only appropriate when the player first comes into work. The (ignore-failure 'except-em) and effect-only annotations ensure that the generation of a stream of thought has no effect on terminating this story context, while either the success *or* failure of avoid-being-chewed-out-for-late causes this behavior to terminate with success, resulting in a transition to the next story context.

Narrative Effects

Once the avatar is maintaining a subjective state, that state must influence the user's experience. That is, the subjective state must somehow be expressed. So far I've experimented with two classes of effects: manipulating sensory descriptions and generating a stream of thought.

Sensory Descriptions

The default Oz avatar produces pseudo-objective factual descriptions of the world. Figure A-5 is a short trace of the default avatar in *Fastfood World*.

```
You are in the counter area.
... <some stuff deleted> ...
The deep fat fryer, the cash register and the food tray are in the
counter area.
The three hamburgers are on the food tray.
You are wearing your clothes.
You hear the hot oil sizzling in the fryer.

PLAYER> wait
... <a turn deleted> ...

You wait.
 You hear the hot oil sizzling in the fryer.
Barry is speaking to you.
Barry's voice says "Wait a minute there, buster"

PLAYER> look at hamburgers

You look at the hamburgers.
The three hamburgers are on the food tray.
You hear the hot oil sizzling in the fryer.
Barry goes to the counter area.
Barry is no longer in the window area.
```

Figure A-5. Interaction trace with default avatar

Each turn, sensory information propagates through the physical world graph. Each sensory datum describes an object, action or event. The default avatar constructs intermediate semantic representations called groups to describe the sensory data received during a given clock tick. These groups are then passed to the natural language generator.

In the current implementation, subjective descriptions are generated by intercepting the groups before they are sent to the generator. The groups are passed through a set of rules that, based on the structure of the group and the current emotional state, render the group in natural language. Currently, rules generate descriptions of groups using canned text. If no rule matches, the group is eventually passed to the natural language generator. Figure A-6 shows the typical structure of description rules.

```
(sequential-production describe-something (group)
   (precondition (and  (<group has a given form>)
         (<avatar in a given state>)))

   (<render the group in a given way>))
```

Figure A-6. Typical structure of a subjective avatar description rule

Figure A-7 is a short trace of the subjective avatar in the *Fastfood World*.

```
You are in the counter area.
... <some stuff deleted> ...
The deep fat fryer, the cash register and the food tray are in the
counter area.
The three hamburgers are on the food tray.
You are wearing your clothes.
You hear the hot oil sizzling in the fryer.

PLAYER> wait
... <a turn deleted> ...

You wait.
With a vindictive gleam in his eye, Barry snaps "Wait a minute there,
buster"

PLAYER> look at hamburgers

You look at the hamburgers.
The faceless crowd of hamburgers sits on the food tray.
As if pop rocks had been poured directly on your brain, the hideous
sizzle of hot grease emanates from the fryer.
Barry marches toward you from the drive-up window station.
```

Figure A-7. Interaction trace with subjective avatar

In this trace, several effects are present. In the first turn, the player hears the sizzle of hot grease. In the second turn, this is absent. This occurs because one of the rules checks if this is the first time the player has been behind the counter where he can hear the fryer. If so, the group describing the fryer's sound is passed onto the natural language generator. After the sound has been initially described (thus making the player aware that the fryer does make noise), the sound is not described unless the player is in a particularly fearful state.

In the second turn, the player is not feeling enough fear, so the description of the fryer sound is bypassed. However, the description of Barry speaking to (snapping at) the player is altered. This is due to a moderate amount of fear generated by an increase in the likelihood of failure (the player's belief) of the avoid-being-chewed-out goal failing. This increase in the likelihood of failure is due to hearing Barry speak while the player is still coming in late to work (the player hasn't started working before Barry noticed him).

In the third turn, the description of the hamburgers has changed since the description in the first turn. This is due to a rule that renders any group describing more than two similar objects as "a faceless mass" if the player is in a bad enough mood. Since the matching criterion on the group is fairly general (any description of a multiple number of objects), I hope that such a narrative effect can become a repeated device, a way of communicating a sense of depressing conformity by repeating a key phrase in a variety of contexts. The ability to match on a generic emotion (such as a bad mood) rather than just on specific emotions (e.g. anger at Barry for some specific reason) is facilitated by combination rules in Em that can combine the types and intensities of more specific emotions into an intensity on a more general emotion.

The sizzle of hot grease is back, but this time in a more colorful form. Barry moving into the counter area (same physical space with the player) initiated a new burst of fear from a fear attitude the player holds towards Barry. This new burst of fear was enough to activate the rule that describes the sound of the fryer.

Finally, Barry's movement into the counter area is described by one of a set of rules that watch for any movement Barry makes. Barry's movements are described differently depending on the current emotional state.

Stream of Thought

In addition to manipulating sensory descriptions, I've also experimented with generating a stream of thought. I hope to provide the user with additional orientation to the character they are playing by providing access to the inner life of this character.

The first category of thoughts in this stream of thought is commentary on the current situation. As an example, in the behavior come-into-work-late (described above), one of the subgoals is think-about-being-late. When the user first enters the world, this behavior produces "Damn! I'm late for work. I hope Barry doesn't notice me". If, for several moves in a row, the player makes no progress towards getting to the counter, it then produces "If I can only get to the register and sign in before Barry sees me". When the player first arrives in the counter area, and if the message above wasn't produced (the player didn't take awhile to get to the counter), the behavior produces "If I can just get signed into the register before Barry sees me, I'll be all right". In this example, the commentary takes the form of hints, helping the user to understand the current story situation. In general, such commentary can be less direct, such as thinking "Yeah, right!" when Barry says something like "You need to be a team player".

The recall of memories is a mechanism that could help a user to understand their role more deeply over time. Fragments of events that occurred prior to the story experience could be recalled in appropriate story and emotional contexts. This would give the user access to the prior life of the avatar. I have not yet implemented such memories.

Daydreams may be another mechanism for communicating role information to the user. Mueller's work on daydreaming [Mueller 1990] is particularly relevant. I have not yet implemented any daydreaming.

Related Work

Hayes-Roth's work on improvisational puppets [Hayes-Roth & van Gent 1996] shares a similar goal of helping a user to play a role. Her work, however, focuses on *actions* performed by an avatar rather than on maintaining a complex internal state. In her work, she provides the user with an interface for giving the avatar high level directives. The avatar then translates these directives into concrete actions consistent with the character. In my work, the space of actions the avatar can engage in is not explicitly presented to the user. Instead, I hope to implicitly guide the user in their role by providing access to subjective sensory descriptions and a stream of thought.

Galyean's work [Galyean 1995] employs narrative manipulation of point of view in an immersive, graphical story world. This manipulation guides the user through the story. The narrative manipulation, however, is controlled by a central story manager rather than by the internal state of an avatar. His system also provides no facilities for access to a stream of thought.

Conclusion

In this paper I've described a kind of socially intelligent agent: subjective avatars. Such avatars, employed in story worlds, have autonomous emotional reactions to events in the world and keep track of story context. Much of this context is concerned with the avatar's social relationship with other characters in the world. The emotional state and story context are used to provide subjective descriptions of sensory information and to generate a stream of thought. The purpose of such description is to help a user gain a deeper understanding of the role they are playing in the story world.

APPENDIX B
OFFICE PLANT #1

Introduction

Walking into a typical, high tech office environment one is likely to see, among the snaking network wires, glowing monitors, and clicking keyboards, a plant. What a sad creature it is. Domesticated yet ill adapted to its artificial niche of human design, this generic plant sits on a desk corner under artificial illumination, serving as a placeholder for that which electronic machinery can not offer: personal attachment. Office plants are an expression of a need for undemanding presence in an efficiently impersonal environment. But there are better solutions:

Office Plant #1 (*OP#1*) is an exploration of a technological artifact, adapted to the office ecology, which fills the same social and emotional niche as an office plant. *OP#1* monitors the ambient sound and light level, and, employing text classification techniques, also monitors its owner's email activity. Its robotic body, reminiscent of a plant in form, responds in slow, rhythmic movements to comment on the monitored activity. In addition, it makes its presence and present knowledge known through low, quiet, ambient sound. *OP#1* is a new instantiation of our notion of *intimate technology*, that is, technologies that address human needs and desires as opposed to technologies that meet exclusively functional task specifications. *OP#1* lives in a technological niche and interacts with users through their use of electronic mail. It acts as a companion and commentator on these activities.

A version of this appendix first appeared in [Boehlen & Mateas 1998].

Figure B-1. ***Office Plant #1*** **in a gallery office**

Concepts

In this section we describe the major artistic and technical concepts that underlie the design of *OP#1*. These concepts are: Email Space, Text Classification, Plant Behavior Architecture, and Sculptural Presence. In our practice we simultaneously explore both spaces; artistic and technical constraints and opportunities mutually inform each other. The arrangement of this section exposes this simultaneous exploration.

Email space

Once, social interaction was defined by communal space. The properties of the space delimited the forms of exchange. The local pub, for example, was a space large enough to support a critical threshold of social energy, public enough that a cross-section of the local population was present, yet small enough that one could notice friends and acquaintances. This is the ideal of public intimacy; crowd presence without alienation. Once, letter writing was bound to paper. In this medium, recording ideas required time and effort. This opened a space for contemplative introspection. Something could be learned about the self while writing to the other. This is the ideal of reflective intimacy; private sharing combined with distancing.

Technology, in its usual move of utopian plentitude, offers to satisfy both desires in one convenient package, email. While new forms of computer interaction are continuously created, email is the first computational forum for human social interaction to become ubiquitous.

The lack of public intimacy in the anonymity of the suburb is promised to no longer be a problem. Virtual communities can be formed and conveniently connected by email. With a lowered threshold for message creation and near instantaneous transmission, email is a conversational medium. But the conversants aren't subject to the constraints of real-time response. Given additional time to think, they can engage in the construction of letter writing. But this new medium, while pretending to offer the catch-free satisfaction of two desires, also introduces the watchword of computing into social interaction: efficiency.

Email encourages constant connection. Reflective letter writing may take place in the evening, after the work day is finished. But how inefficient to separate work and personal life. Email encourages continuous multi-tasking between work, play, and social interaction. Sitting constantly at the computer, words can be processed, numbers tabulated, games played, letters answered, all in one undifferentiated flow of activity. Where conversation and letter writing used to require distinct context shifts involving changing mental state as well as physical location, the ease with which the user can switch contexts on the computer belies any distinction between these activities. And the ease with which an email can be sent ensures that all of us will be receiving dozens if not hundreds a day. With the blurring of historical distinctions surrounding concepts such as efficiency, pleasure, conversation and work, this increasing stream of information contains an odd mixture of work related announcements, junk mail, meeting requests, short quips from friends, and occasional heart-felt letters. Offering a seductive outlet for the primal human desire for social contact, email represents the transformation of the alluring familiarity of the letter and conversation by the logic of the machine.

As a new hybrid communication space, email is a fascinating site in which to observe human adaptation to and negotiation within a new medium. *OP#1* is a commentator on this space. It physically responds to the social and emotional content of email messages

received by the user. Unstructured, this email space is not accessible to scrutiny. In order to open this new social sphere for analysis and questioning, we have developed, after reviewing a large body of email, the following categorization scheme.

Text classification

Email messages are sorted into one or more of the following 17 binary classes: private, public, intimate, chatty, meeting, complaint, request, FYI, apology, humor, social event, literature, machine error, problem, response, update, well wishing. Any one email may belong simultaneously belong to several classes, such as a letter from a friend simultaneously being private, chatty, and well wishing.

In order to sort in-coming electronic messages into one of these categories we employ a mixture of naïve bayes [Mitchell 1997: 177-184]and k-nearest neighbor [Mitchell 1997: 230-236] text classification. Both methods make use of training data, in our case ~4000 hand classified email messages, to tune classifier parameters. In the case of naïve bayes, the parameters are the conditional probabilities of a word occurring given the class (e.g. public, social event), obtained by counting word occurrences for every word appearing in the training examples. These probabilities are used to classify each new instance by applying:

$V_{nb} = \text{argmax}\ P(v_j)\ \Pi P(a_i \mid v_j)$

where V_{nb} denotes the target value output by the classifier. $P(v_j)$ is the prior probability of a document class v_j. $P(a_i \mid v_j)$ is the conditional probability of a word a_i appearing given the class v_j.

In the case of k-nearest neighbor, the training step simply consists of storing the training data. Each document in the training set is viewed as a point in vocabulary space, a multi-dimensional space where the number of dimensions is equal to the number of unique vocabulary words appearing across all training documents. Given a new email to classify, the distance is computed between the new email and every stored email from the training set (the distance between two documents is proportional to the number of vocabulary words they share); the new email is assigned the most common class shared by the closest k emails.

The classification method to use for each class was empirically determined.

Plant behavior architecture

The state of the plant is dynamically modeled with a fuzzy cognitive map (FCM) [Kosko 1997: 499-525]. In a FCM, nodes representing actions and variables (states of the world) are connected in a network structure reminiscent of a neural network. FCMs are fuzzy signed digraphs with feedback. Nodes are fuzzy sets or events that occur to some degree. At any point in time, the total state of the system is defined by the vector of node values. In our implementation, the nodes represent actions. The action associated with the action node with the highest value is executed at each point in time. The values of nodes change over time as each node exerts positive and negative influence on the nodes it is connected to. The FCM approach is attractive because it resolves contradictory inputs and maintains sufficient state to exhibit longer-term temporal effects.

Rules map boolean combinations of email classifications into changes in node activation. For example, if an incoming email is both intimate and private, the activation of the bloom node is increased. This makes the plant more likely to bloom, excites the half-bloom state, and inhibits the rest state. Which node will end up with the largest activation, and thus determine the plant's next action, depends on the prior activation

state of the FCM, as well as any additional changes to activation states caused by this email (the set of binary classes assigned to the email may trigger multiple rules) and other simultaneously received emails. Figure B-2 shows the FCM for *OP#1*.

Sculptural Presence

Office Plant #1 is a desktop sculpture, an office machine that serves as a companion. In an environment of button pushing activity, *OP#1*, like a good piece of sculpture, is always on. *OP#1* creates its own kind of variable presence in a user's email space, taking on various attitudes and falling into decided silence. Its reactions serve as a commentary on the events it analyzes. It goes beyond mirroring these events and delivers reactions as if it understood the significance of the exchange. But effectively, *OP#1* is mostly inactive. It has a well defined sense of doing nothing, yet. It is simply there and in that sense a traditional piece of sculpture. Its physicality is as important as its text classifying capabilities.

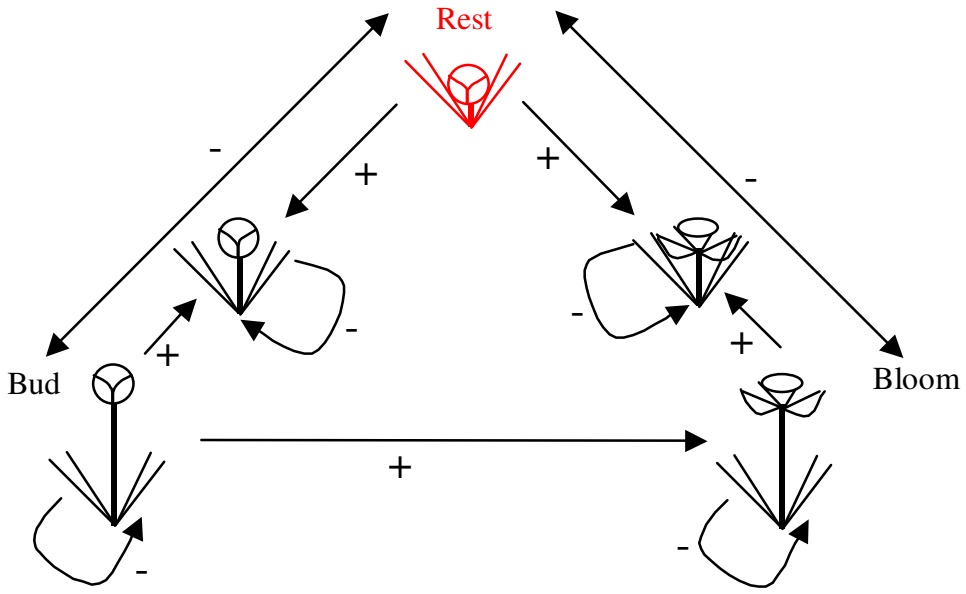

Figure B-2. *Office Plant #1's* fuzzy cognitive map

Physical design

OP#1 consists of a ball/bulb surrounded by metal fronds mounted on a base. The ball, a hammered aluminum sphere, can open and close. Mounted on a stem, it can also rise above the fronds and remain in any intermediate position. The fronds, made of piano wire, sway slowly, moving individually or in synchrony.

A window in the bottom of the base promises to reveal the inner workings of the plant. Rather than revealing gears, motors and electronics, this window opens onto the *datarium,* a scene composed of rocks, sand and a moving counterweight. As the stem raises and lowers, the counterweight moves into the *datarium* or out of view.

In addition to physical movement, *OP#1* has a voice; it produces sound using the speaker in the bulb. These sounds provide the plant with a background presence. The possible sounds include variations on whistle, chant, sing, moan, complain, and drone.

For *OP#1*, the action nodes of the FCM correspond to physical postures. The three primary physical postures of the plant are rest, bud and bloom. Sounds and frond movements are associated with each physical posture. In rest, the bulb is closed and fully lowered. The fronds occasionally move. In bud, the bulb is closed and fully extended. In bloom, the bulb is open and fully extended. In both the bud and bloom states, the plant occasionally performs various frond movements, such as twitching the fronds in a circular pattern. When the node associated with a posture has the most activation energy, the plant performs the action of moving to this posture from its current posture. Activation energy from bud flows towards bloom; budding makes blooming more likely. Rest and bud, and rest and bloom, are mutually inhibitory. Rest and bud both spread their energy to an intermediate posture, and rest and bloom spread their energy to a second intermediate posture. The combination of the mutual inhibition plus the intermediate posture will cause these pairs of states to compromise towards the intermediate posture. Finally, the self-inhibitory links tend to cause values in the system to decay; in the absence of input, the plant will not stay in a posture forever. When all of the nodes are zero, the plant will move towards the rest posture. As email is classified, energy is added to nodes, thus initiating the process of competition and cooperation between the nodes.

Implementing plant movement

Machines excel at performing fast and precise movement. For this task, the requirements are very slow movements. In order to achieve slow linear and rotary motion under space limitations we choose to use micro-stepping stepper motors. This allows both slow and precise movement control [Emerald & Kadrmas 1997; Emerald, Sasaki & Takahashi 1996]. We have tested a variety of actuators for the fronds, amongst them Polyelectrolyte Ion Exchange Membrane Metal Composites (IEMMC) [Mojarrad 1997] and Shape Memory Alloy (SMA) [Dario et. al. 1997]. We choose SMA, as it requires little space, no climate control and provides acceptable reaction times.

Intimate technology

As stated above, *OP#1* is an instantiation of intimate technology. As opposed to traditional machinery that is designed to perform well-defined and economically useful tasks, intimate technology attempts to focus on human desires, in particular desires for contemplation and engaged leisure. Intimate technologies are best situated not in a sterile laboratory setting but in the home or office. Close to people, in bedrooms, kitchens, and as carry on items, intimate technologies act in the niches from which desires have been efficiently eradicated. In our conception of intimate technologies, the device is a mediator between the realm of repeatable machine precision and human instinct. Intimate technologies are an attempt to reclaim the territories colonized by the unquestioned pursuit of efficiency. Intimate technology proposes to reintroduce contemplation into the design space and to build machinery that allows and fosters it. Intimate technology is a form of technology critique, but one that effectively uses what engineering disciplines best offer.

Acknowledgements

We would like to thank the following companies and individuals for their support in this project:

- Allegro Microsystems, who generously supplied electronic components.
- Microkinetics, who discounted their stepper motors for us.
- George Biddle of the Material Science Department of Carnegie Mellon University, who made his machining and materials expertise generously available to us.

APPENDIX C
TERMINAL TIME

A version of this appendix first appeared in [Mateas, Vanouse & Domike 2000]. Other descriptions of *Terminal Time* can be found in [Mateas, Vanouse & Domike 1999; Domike, Mateas & Vanouse 2002].

Terminal Time is a machine that constructs ideologically-biased documentary histories in response to audience feedback. It is a cinematic experience, designed for projection on a large screen in a movie theater setting. At the beginning of the show, and at several points during the show, the audience responds to multiple choice questions reminiscent of marketing polls. The audience interaction in relationship to the viewing experience is depicted in Figure C-1. In the first question period, an initial ideological theme (from the set of gender, race, technology, class, religion) and a narrative arc (e.g. is this a progress or decline narrative) are established.

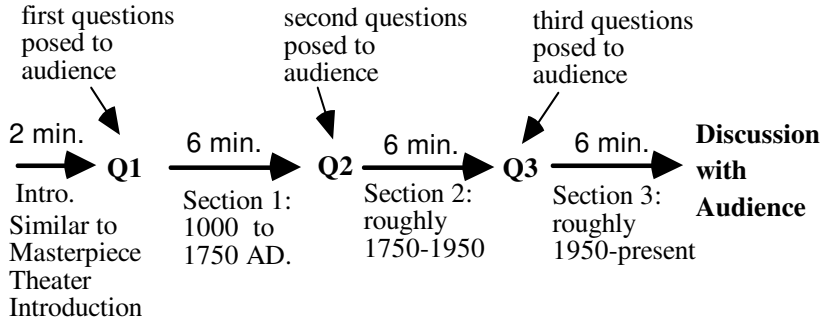

Figure C-1. Audience interaction

The second set of questions refines the ideological theme chosen in the first set, and possibly introduces a sub-theme (e.g. combining race and class, or technology and religion). The third set of questions further refines the theme(s) and introduces the possibility for a reversal (e.g. a decline narrative becoming a progress narrative). An example question (from the first question period) is shown in Figure C-2.

> Which of these phrases do you feel best represents you:
>
> A. Life was better in the time of my ancestors.
> B. Life is good and keeps getting better every day.

Figure C-2. Example question

The audience selects answers to these questions via an applause meter – the answer generating the most applause wins. The answers to these questions allow the system to create historical narratives that attempt to mirror and often exaggerate the audience's biases and desires. By exaggerating the ideological position implied in the audience's

answers, *Terminal Time* produces an uncomfortable history that encourages the audience to reflect on the influence of ideology on historical narratives.

Terminal Time is a collaboration between a computer scientist specializing in AI-based art and entertainment, an interactive media artist, and a documentary filmmaker.

Terminal Time's architecture consists of the following major components: knowledge base, ideological goal trees [Carbonell 1979], rule-based natural language generator, rhetorical devices, and a database of indexed audio/visual elements primarily consisting of short digital movies and sound files containing music. Figure C-3 shows a diagram of the architecture.

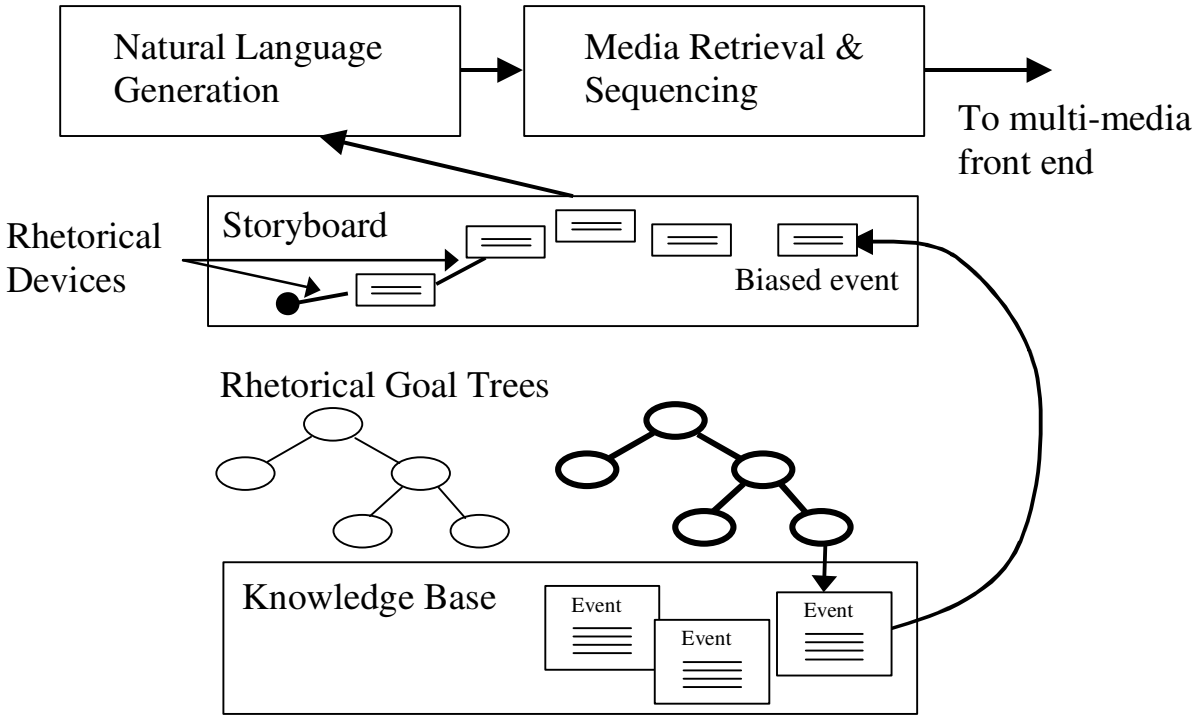

Figure C-3. *Terminal Time* architecture

The knowledge base contains representations of historical events. This is the raw material out of which the ideologically-biased histories are constructed. Examples of historical events are the First Crusades, the invention of Bakelite, and the rise of enlightenment philosophy. Ideological-goal trees represent the current ideological-bias being pursued by the narrator. The goal-trees consist of rhetorical goals ordered by subgoal and importance (to the ideologue) relationships. These goals are used both to select historical events to include in the story and to "spin" the event in an ideologically-consistent manner. The rule-based natural language generator (NLG) generates the narrative text once specific facts have been selected and connected to make a story. The storyboard serves as a working memory for processes that impose a narrative order on event spins created by the goal tree. Rhetorical devices are connecting pieces of text with accompanying constraints on story structure. These devices are used to create narrative connections between historical events. Finally, the multimedia database contains the audio/visual elements for the assembled documentary. Once a narrative track has been constructed, information retrieval techniques are used to match the "best" indexed multimedia elements to the appropriate pieces of text. Once the multimedia elements have been selected, the

resulting documentary is displayed, layering text-to-speech synthesis of the narrative track, and the video and audio elements.

The audience's responses to the questions influence the machine by selecting and editing rhetorical goal trees, selecting a set of rhetorical devices, and placing constraints on the storyboard. In a sense, the audience's responses parameterize the machine. The responses activate structures and processes; the machine then autonomously generates a biased history.

The rest of this paper describes the artistic goals of *Terminal Time*, provides more detail about each of the architectural components, and describes our experiences performing *Terminal Time*.

Artistic Goals

Documentary Form

The popular model of the documentary film in America today, as exemplified, for example, by Ken Burns' "The Civil War", has the familiar structure of Western narrative. The rhetorical structure invariably involves a crisis situation, a climax, and a clear resolution. Generally there is one prevailing narrative, one interpretation of the historical facts presented. The documentary situates itself as an objective retelling of history, making it difficult for the viewer to question the authority of the presented viewpoint.

Terminal Time imitates the model of this "cookie-cutter documentary" with a machine that produces and reproduces it, until the model itself is revealed for the tool of ideological replication that it has become. *Terminal Time* generates endless variations of documentaries that have the "look and feel" of the traditional, authoritative PBS documentary. Yet these generated documentaries are clearly charged with a strong ideological bias derived from the audience's responses. In this way *Terminal Time* invites the viewer to question the implicit authority of documentary presentations of history.

Utopian Navigation

There is a great deal of industry hype surrounding interactive media and computing. Typically such experiences are promoted through a rhetoric of utopian navigation. According to such rhetoric, the computer provides unlimited access to information and experience, a pure source of empowerment that imposes no interpretation on the data that is processed. Other familiar tropes in this rhetoric include: Real-time, Immersion and Virtuality – promising the thrill of reality or hyper-reality, without the effort, right from one's own PC. Microsoft's ads softly beguile us with the question "Where do you want to go today?"

With *Terminal Time*, we play with these notions by building a program that engages in active interpretation and construction of the interactive experience. While the resulting constructed histories clearly respond to audience input, the system has a mind of its own, pushing the story into extremes that the audience did not intend. Thus value-free navigation gives way to a value-laden interpretation. *Terminal Time* is a program that bites back.

Audience Experience

Utilizing indirect questionnaires as a user interface, the system essentially target markets each audience with an appropriate history. Rather than asking audiences what type of history they would like, or how they would like to navigate through history, they are asked questions about their own demographics and psychographics: their work status, what cultural trends they find most disturbing, how well they get along with others, etc. The resulting history is like holding a funhouse mirror to the audience; it reflects an exaggerated and distorted view of the audience's biases.

As the history begins 1000 years ago, the audience should experience a comfortable sense of historical authority engendered by the familiar documentary form and the remoteness of the historical events. As the history unfolds, the effect of the periodic audience polls becomes more and more apparent. The increased bias evident in the history should begin creating a tension with regard to the veridicality of the history (a sense of "wait a minute, this doesn't seem quite right..."). Ideally, this tension should reach a maximum as the piece moves into modern history.

Knowledge Base

Upper Cyc Ontology

The knowledge base consists of higher order predicate statements about historical events, definitions of ontological entities used in the historical event descriptions (individuals and collections), and inference rules. *Terminal Time*'s ontology is based on the Upper Cyc Ontology, the top 3000 most general terms in the Cyc ontology [Lenat 1995]. The Upper Cyc Ontology is available free of charge from Cycorp[41]. It does not include any other components of Cyc (theorem prover, natural language engine, database, etc.). However, the upper ontology provides a useful set of distinctions in terms of which the more specific ontology needed by *Terminal Time* can be defined.

Example Historical Event

Figure C-4 shows the representation of The Giordano Bruno story. Those terms preceded by a $ are defined in the Upper Cyc Ontology. Those terms preceded by % are defined within the *Terminal Time* ontology in terms of the Upper Cyc Ontology. Figure C-4, glossed in English, states:

> The Giordano Bruno story, a historical event occurring in the 16th and 17th century, involved the creation of a new idea system and an execution. The idea system created in this event conflicts with the idea system of medieval Christianity. Both Giordano Bruno and a portion of the Roman Catholic Church were the performers of this event. Giordano Bruno was acted on (he was executed) in this event.

In this particular representation of the story of Giordano Bruno, both the creation of his philosophical writings and his execution by the Roman Catholic Church are treated as a single compound event. If we wanted *Terminal Time* to be able to treat these two events separately (perhaps talking about Bruno's writings without mentioning his eventual execution), they could be represented as two sub-events. In general, events are only

[41] http://www.cyc.com/

represented as deeply as is needed by the rhetorical goal trees, storyboard, and natural language generator.

```
;; Giordano Bruno
($isa %GiordanoBrunoStory %HistoricalEvent)
($isa %GiordanoBrunoStory %IdeaSystemCreationEvent)
($isa %GiordanoBrunoStory %Execution)
(%circa %GiordanoBrunoStory (%DateRangeFn (%CenturyFn 16) (%CenturyFn 17)))
($eventOccursAt %GiordanoBrunoStory $ContinentOfEurope)
($performedBy %GiordanoBrunoStory %GiordanoBruno)
($outputsCreated %GiordanoBrunoStory %GiordanoBrunosIdeas)
($isa %GiordanoBrunosIdeas $PropositionalInformationThing)
($isa %GiordanoBrunosIdeas $SomethingExisting)
(%conflictingMOs %GiordanoBrunosIdeas %MedievalChristianity)
($isa %GiordanoBrunosIdeas %IdeaSystem)
($performedByPart %GiordanoBrunoStory %TheRomanCatholicReligiousOrg)
($objectActedOn %GiordanoBrunoStory %GiordanoBruno)
```

Figure C-4. Example event for the Giordano Bruno story

Inference Engine

The inference engine, implemented in Common Lisp, is based on the interpreter implementing higher-order hereditary Harrop logic described in [Elliott & Pfenning 1991]. Hereditary Harrop logic allows knowledge base entries (the program, thinking in logic programming terms) to consist of Horn clauses, and queries (goals) to consist of all the standard Prolog-like goals (atomic goals, conjunctions, disjunctions, existentials), plus embedded implications (assumptions). The interpreter also includes extra-logical support for operations such as unifying logic variables against a function evaluated by Lisp.

The inference engine is used to answer all queries about historical events. For example, in the discussion below of ideological goal trees, the historical event tests are all made using the inference engine.

Ideological Goal Trees

Terminal Time organizes ideological bias with goal trees, adapted from *Politics* [Carbonell 1979]. In *Politics*, ideology is encoded as a set of goals held by the ideologue. The goals are organized via subgoal links (not corresponding exactly to either the conjunctive or disjunctive notion of subgoal) and relative importance links. The relative importance links place an importance partial order over the subgoals. For example, in *Politics*, the US Conservative ideologue's most important goal is Communist Containment. This goal has a number of subgoals such as Have a Strong Military, Aid Anti-Communist Countries, etc. Though Have a Strong Military and Aid Anti-Communist Countries are sibling subgoals, Have a Strong Military has a higher relative importance. In addition to their own goal tree, an ideologue also possesses beliefs about the goal trees of others. In Carbonell's system, the goal trees were used to organize inferences made by a news story understanding system.

In *Terminal Time*, the goal tree has been modified to represent the goals of an ideological story-teller. Rather than having goals to modify the world, the story-teller has rhetorical goals to show that something is the case. For example, the Anti-Religious

Rationalist possesses the goals show in Figure C-5 during the first segment of the history. The indented goals are subgoals.

The leaf goals in the goal tree are used to organize two kinds of knowledge: a set of tests for recognizing when a historical event is potential fodder for satisfying the rhetorical goal, and a set of plans for actually constructing the description of the event to satisfy the goal (the event spin).

```
show-religion-is-bad
    show-religion-causes-war
    show-religion-causes-crazy-self-sacrifice
    show-religion-causes-oppression
    show-religion-causes-self-abuse
    show-thinkers-persecuted-by-religion
show-halting-rationalist-progress-against-religion
    show-thinkers-opposing-religious-thought
    show-thinkers-persecuted-by-religion
```

Figure C-5. Anti-Religious Rationalist goal tree

Notice that the leaf goal show thinkers persecuted by religion is a subgoal of *two* higher level goals. Satisfying this goal satisfies both higher-level goals.

Tests for Event Applicability

An ideologue needs a way of recognizing when a historical event could be used to satisfy a goal (make an ideological point). For example, the Anti-Religious Rationalist must be able to recognize that the Giordano Bruno story can be used to show-thinkers-persecuted-by-religion. This involves recognizing that a religious organization does something negative to a thinker because of the thinker's thoughts. In the current version of *Terminal Time*, this test determines whether an event involves both the creation of an idea system and an execution, and whether the idea system conflicts with some religious belief system. The formal syntax of this test expressed in the language of the inference engine is shown in Figure C-6.

```
(%and
    ($isa ?event %IdeaSystemCreationEvent)
    ($isa ?event %Execution)
    ($outputsCreated ?event %Execution)
    (%conflictingMOs ?newIdeas ?relBeliefSystem)
    ($isa ?relBeliefSystem $Religion))
```

Figure C-6. Event applicability test for show-thinkers-persecuted-by-religion

This test assumes that the execution must have been performed by the religious organization in response to the creation of the conflicting idea system. Further, it assumes that the only form of persecution is execution. These simplifying assumptions work because, given the current content of the knowledge base, this applicability test is sufficient to locate events that can be slanted as forms of religious persecution. As new events involving religious persecution are added to the knowledge base, the test will most likely have to be broadened.

Plans for Event-level Story Generation

Once an event has been recognized as applicable to a rhetorical goal of the ideologue, additional knowledge is necessary to spin the event in such a way as to satisfy the rhetorical goal. This knowledge is represented as rhetorical plans. These plans put a "spin" on the event (referred to as a spin) by selecting a subset of the event knowledge represented in the KB to place on the storyboard. Rhetorical plans are the mechanism by means of which a rhetorical goal can place its unique view on an event.

The plan language is similar to the rule language for natural language generation, except that the atomic actions for the NLG rule language emit strings while the atomic actions for the plan language add syntactic units to an event spin. See the section on NLG rules for a more detailed description of the logic allowed in rhetorical plans. An outline of an example plan for show-religion-causes-war is shown in Figure 7.

Describe the individual who called for the war, mentioning their religious belief Describe the religious goal of the war Describe some event happening during the war Describe the outcome

Figure C-7. Rhetorical plan outline for show-religion-causes-war

In addition to the knowledge elements selected by the rhetorical plan, the name of the rhetorical goal and the names of all of its parents are added to the spin. The goal name(s) tell the rhetorical devices and natural language generator which goal(s) a particular spin is satisfying.

Rhetorical Devices

After the rhetorical goals are done producing event spins, the storyboard now contains an unordered collection of spins. Rhetorical devices connect spins together to form a story. Rhetorical devices consist of an English sentence(s) (actually represented as NLG rules) and accompanying logical tests that can be used to connect spins together. For example, the sentence "Yet progress doesn't always yield satisfaction" can be used to connect several spins describing the positive effects of technological progress and several spins describing social or environmental problems arising from technological progress. The associated tests require that all the spins preceding the rhetorical device must be positive technological, artistic, or industrial progress, and that all the spins following the rhetorical device must be negative effects of progress.

Prescope and Postscope Tests

The prescope of a rhetorical device is the ordered collection of spins preceding the device. The postscope is the ordered collection of spins following the device. The prescope and postscope tests are constraints (interpreted by the inference engine) that the preceding and following spins must satisfy in order for the rhetorical device to be applicable (that is, able to glue the spins together). Scope tests can either require that all the spins in the scope satisfy the test or that at least one spin in the scope satisfies the test. In addition, the scope range and length can be specified. The scope length is the number of spins to include in the scope; the default is 1 (that is, only the preceding or following spin must satisfy the test). The scope range specifies the range of spins that can be

searched for a satisfying scope; the default is (1 1) (the range consists of only the immediately preceding or following spin).

Rhetorical Device NLG Rule

Associated with each rhetorical device is an NLG rule for generating the English sentence associated with the device. For some rhetorical devices, this may be a simple rule generating a single canned sentence. For other (more flexible) rhetorical devices, the rule may be passed arguments (which were bound by the scope tests) that influence the generated sentence.

Example Rhetorical Device

An example rhetorical device is shown in Figure C-8.

```
(def-rhetdev :name :non-western-religious-faith
   :prescope-length 2
   :prescope-test (:all-events-satisfy (%and
     ($isa ?event %HistoricalSituation)
     (:kb ($eventOccursAt ?event %FirstWorld))
     (%rhet-goal :show-religion-is-good)))
   :postscope-test (:some-event-satisfies ?spin (%and
     ($isa ?event %HistoricalSituation)
     (:kb ($eventOccursAt ?event %NonFirstWorld))
     (%rhet-goal :show-religion-is-good)))
   :nlg-rule :generate
   :nlg-context-path (:non-western-religious-faith))
```

Figure C-8. Example rhetorical device for the Pro-religious Supporter

This rhetorical device, employed by the Pro-religious Supporter, is used to connect a couple of spins describing Western religious faith, with an example of non-Western religious faith. The prescope-length is 2; since no prescope-range is specified, it defaults to the preceding two events. Thus the prescope test must be satisfied by the immediately preceding two spins. The test requires that the event occurred in the First World (represented in the ontology as a collection of geographical regions that includes regions such as Europe) and that it satisfied the rhetorical goal show-religion-is-good. The %rhet-goal term was added to the event spin during rhetorical plan processing (when the rhetorical goal sticks the spin on the blackboard). Most rhetorical devices test the satisfied rhetorical goal in their scope tests; these goal labels indicate how an event has been slanted in order to create a specific even spin. The postscope test similarly tests whether the immediately following event spin satisfies the goal show-religion-is-good in the *non*-First World. In the event that the constraints are satisfied, text will be generated by the NLG rule. In this case the NLG rule produces the text "The call of faith was answered just as ardently in non-western societies."

Story Generation

Once a collection of event spins has been placed on the storyboard, a historical story can be generated. For each of the three periods of the documentary, each ideologue has a list of storyboard constraints. The storyboard constraint for section 1 of the Anti-Religious Rationalist is shown in Figure C-9.

```
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-religion-is-bad)
(%rhet-goal :show-halting-rationalist-progress)
```

Figure C-9. Anti-Religious Rationalist storyboard constraint

The length of the constraint list determines how many event spins will be included in the story section. In this example, six spins will be included. Each test in the list constrains the spins that can appear in each position in the story. Typically these are constraints on the rhetorical goals that were satisfied to create the spin. In addition to the storyboard constraints, there is also an implicit temporal constraint that requires that spins appear in roughly chronological order.

To generate a story, the space of all sequences of event spins satisfying the storyboard constraints is searched for a sequence that can be satisfied by the current set of rhetorical devices. A sequence is satisfied if a rhetorical device with satisfied scope tests can be placed between every spin in the sequence. The resulting sequence of interleaved spins and rhetorical devices is a story.

NLG Rules

The natural language generation (NLG) system generates English text for both event spins and rhetorical devices. NLG is accomplished using rules that map pieces of knowledge representation onto English. There is no deep theory of lexical choice or text planning. The goal of the NLG system is to produce high quality text for the stories generated on the storyboard. The rule language provides a framework in which a human author can write text ranging in granularity from canned paragraphs down to individual words and phrases and provide the logic to map these varying chunks onto pieces of knowledge representation.

NLG Rule Syntax

Figure C-10 provides an abstract example of a rule. This example makes use of most of the features supported by the rule language. This language is similar to the language used for rhetorical plans. All tests mentioned in the NLG rules are interpreted by the inference engine.

An NLG rule is identified by a name and a rule context. The rule context provides a name space in which NLG rule names are recognized. Each context defines a set of rules that are specialists in handling some particular NLG task. Typically, a separate rule context is used for each historical event found in the knowledge base. When generation is initiated, a rule name, arguments (list of knowledge representation elements for which English should be generated) and a context list are given. The context list provides a set of contexts (in order from most specific to most general) in which to search for rules matching the rule name.

When a rule with matching rule name is found, the test is evaluated against the arguments to determine whether to use that instance of the rule for generation. Within a context, there may be multiple rules with the same name (corresponding to different ways

to accomplish the same generation task); the test is used to determine which of these rules should be applied given specific arguments.

```
(def-nlgrule
   :name :rule-name
   :context :some-context
   :test test over the rule arguments
   :body (:seq
     (:terminal
       (:string "string 1")
       (:keywords k1 k2 k3))
     (:cond
       ((test1 over rule arguments) (:terminal…))
       ((test 2 over rule arguments) (:bag-one-of step1…stepn)))
     (:rule subrule args (context1 … contextn))
     (:opt (:if (another test) (:seq…)))))
```

Figure C-10. NLG rule syntax

Once a rule is found whose test evaluates to true, the rule body is interpreted. In the example rule, the rule body is a sequence of steps. Terminals are the atomic steps that emit language. In addition to emitting an English string, terminals emit keywords associated with the English string. These keywords are used by the multimedia retrieval subsystem to associate a video clip with the sentence.

The conditional step (:cond) allows generation to branch depending on tests over the rule arguments. The branches may either by individual terminals or an entire rule body. If none of the tests in a conditional succeeds, then the rule fails; the system will try to find other applicable rules for generation. The bag-one-of body in the second branch of the conditional chooses one of the steps at random to execute. This can be used to add some random variation to generation.

A rule may contain a call to another generation rule.

The :opt construct allows the insertion of an optional conditional step. If the conditional is satisfied, the conditional branch is executed. If the conditional fails, execution of the rule body continues after the optional step.

Video Sequencing

After natural language generation, the event spins and rhetorical devices have been rendered as English text. Video clips from the database of keyword-annotated clips must be sequenced against the text (which forms the narrative track) to create the complete documentary. Video sequencing takes place in two steps. First, the keywords associated with each sentence (the keywords emitted by terminals in NLG rules) are used to retrieve keyword annotated video clips using TF/IDF term-based retrieval [Salton & Buckley 1988]. This creates a list of top-scoring video clips for each sentence (typically the top 4 or 5 are taken). Then a greedy forward and backward search through the narrative track is performed to try and maximize clip continuity while minimizing clip reuse. If a pair of consecutive sentences shares clips among their top scoring clips, this greedy search will play the top-scoring shared clip across both sentences.

Current Status

Currently *Terminal Time* contains 134 historical events and 1568 knowledge base assertions (in addition to the assertions in the Upper Cyc Ontology). Nine major ideologues are represented using a total of 222 rhetorical goals, 281 rhetorical devices, and 578 NLG rules. The video database contains 352 annotated 30 second clips. *Terminal Time* has been performed in 14 venues, including the Carnegie Museum of Art, the Warhol Museum, and as the Walker Museum's entry in Sonic Circuits. Work continues in adding new events, goals, devices, NLG rules and video clips.

Performance Experiences

One way to evaluate an AI-based interactive art work is to evaluate the audience response to the system, to examine whether the AI architecture supports an audience interaction that successfully conveys the artistic intentions of the piece. Our knowledge of the audience reaction to *Terminal Time* comes both from observing audiences during a performance and from the audience discussion period we always hold after a performance.

During performances, the audience is highly engaged with the piece. During the interactive polls, segments of the audience sometimes compete for control, clapping and shouting to make their choice the winner. At other times, the audience laughs when a choice meets with silence (no one wants to vote for it). Sometimes the applause grows into a groundswell of whistling and clapping as it becomes clear that certain choices are nearly unanimous. As the audience watches the constructed histories, there is often laughter, and sometimes groans and gasps. These reactions tend to grow as the documentary proceeds, indicating that the ideological bias is indeed becoming stronger and more visible as the history proceeds.

The discussion period tends to be quite animated, with the audience offering many questions and comments. Common topics of discussion include the role of ideology in the construction of history, the nature of certain specific biases, and the experience of being unable to completely control the machine. From both the audience reactions during the performance and the nature of the post-performance discussion period, *Terminal Time* is successfully creating an engaging audience experience in accord with our artistic intentions.

Related Work

Two of the earliest computational models of ideology are Abelson's *Goldwater Machine* [Abelson & Carroll 1965] and Carbonell's *Politics* [Carbonell 1979]. The *Goldwater Machine* mimics the responses of conservative presidential candidate Barry Goldwater to questions about the Cold War. The *Goldwater Machine's* responses are driven by a Cold War masterscript describing typical roles and event sequences during the Cold War. *Politics* represents rhetorical goals in order to guide biased understanding of news stories. In contrast to both the *Goldwater Machine* and *Politics*, *Terminal Time* generates biased historical stories composed of multiple events, rather than answering individual questions.

Pauline [Hovy 1987] generates natural language text for news events subject to the pragmatic constraints of rhetorical goals. Rhetorical goals include goals of opinion (e.g. show that our side has good goals or takes good actions) and goals of style (level of formality, level of simplicity). *Pauline* knows about 3 events, but is able to produce 100 different descriptions of an event. Where Pauline has a deep architecture for generating descriptions of individual events, *Terminal Time* selects and connects multiple events to satisfy an ideological position.

Spindoctor [Sack 2000a] uses statistical techniques to classify bias in news stories. This system determines the ideological point-of-view expressed in stories about the Nicaraguan Contras. While the use of statistical techniques makes *Spindoctor* robust, it is concerned with classification where *Terminal Time* is concerned with generation.

Some computer based documentaries support the user in navigating through documentary materials (e.g. [Davenport & Murtaugh 1995; Schiffer 1999]). As a user interacts with the system, implicit queries retrieve and play annotated video clips. Where these systems support a user in exploring a documentary space through immediate navigation, *Terminal Time* autonomously generates biased documentaries in response to punctuated audience feedback.

Finally, *Terminal Time* differs from all these systems in self-consciously being a work of art. *Terminal Time* is a piece of interactive performance art designed to create an experience for an audience.

BIBLIOGRAPHY

Aarseth, E. 1997. *Cybertext: Perspectives on Ergodic Literature*. Baltimore: The Johns Hopkins University Press.

Abelson, R. P. 1981. Constraint, Construal, and Cognitive Science. In *Proceedings of the 3rd Annual Conference of the Cognitive Science Society*. Berkeley, CA. pp. 1-9.

Abelson, R. and Carroll, J. 1965. Computer Simulation of Individual Belief Systems. *American Behavioral Scientist*, 8, 24-30.

Adam, A. 1998. *Artificial Knowing: Gender and the Thinking Machine*. London: Routledge.

Adams, E. 1999a. Three Problems for Interactive Storytellers. *Designer's Notebook Column, Gamasutra*, Dec. 29, 1999. http://www.gamasutra.com/features/designers_notebook/19991229.htm.

Adams, E. 1999b. It's Time to Bring Back Adventure Games. *Designer's Notebook Column, Gamasutra*, November 9, 1999. http://www.gamasutra.com/features/designers_notebook/19991109.htm.

Agre, P. 1997a. *Computation and Human Experience*. Cambridge, UK: Cambridge University Press.

Agre, P. 1997b. Toward a critical technical practice: Lessons learned in trying to reform AI. In Geoffrey C. Bowker, Susan Leigh Star, William Turner, and Les Gasser (Eds.), *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide*. Erlbaum.

Agre, P. 1988. *The Dynamic Structure of Everyday Life*. Ph.D. Dissertation. Department of Electrical Engineering and Computer Science, MIT.

Allen, J.F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11), 832-843.

Allen, J., Byron, D., Dzikovska, M, Ferguson, G., Galescu, L., and Stent, A. 1998. An Architecture for a Generic Dialog Shell. *Natural Language Engineering* 6 (3).

Allen, J. F., Schubert, L. K., Ferguson, G., Heeman, P., Hwang, C. H., Kato, T., Light, M., Martin, N. G., Miller, B. W., Poesio, M., and Traum, D. R. 1995. The TRAINS project: A case study in designing a conversational planning agent. *Journal of Experimental and Theoretical AI*. 7, pp. 7 - 48.

Anderson, P. B., Holmqvist, B., Jensen, J. F. 1993. *The Computer as Medium*. Cambridge: The Cambridge University Press.

Andre, E., Rist, T., Mueller, J. 1998. Integrating Reactive and Scripted Behaviors in a Life-Like Presentation Agent. *Proc. of the Second International Conference on Autonomous Agents* (Agents '98), pp. 261-268.

Aristotle, 330 BC. *The Poetics*. Mineola, New York: Dover, 1997.

Assanie, M. 2002. Directable Synthetic Characters. In Forbus K. & Seif El-Nasr M. (Eds.), *Working Notes of the AAAI-2002 Spring Symposium on Artificial Intelligence and Interactive Entertainment*. AAAI Press, pp. 1-7.

Avedon, E., and Sutton-Smith, B. 1971. *The Study of Games*. New York: Wiley.

Aylett, R. 1999. Narrative In Virtual Environments: Towards Emergent Narrative. *Working notes of the Narrative Intelligence Symposium*, AAAI Spring Symposium Series. Menlo Park: Calif.: AAAI Press.

Bailey, P. 1999. Searching for Storiness: Story Generation from a Readers Perspective. *Working notes of the Narrative Intelligence Symposium*, AAAI Spring Symposium Series. Menlo Park: Calif.: AAAI Press.

Barthes, R. 1972. *Mythologies*. (Trans: Annette Lavers). New York: Hill & Wang. Translation of *Mythologies*, first published in 1957.

Barthes, R. 1967. *Elements of Semiology*. (Trans: Annette Lavers & Colin Smith). New York: Hill & Wang. Translation of *Eléments de Sémiologie*, first published 1964.

Bates, J. 1994. The Role of Emotion in Believable Agents. *Communications of the ACM*. 7 (37): 122-125.

Bates, J. 1992. Virtual Reality, Art, and Entertainment. *Presence: The Journal of Teleoperators and Virtual Environments* 1(1): 133-138.

Bates, J. 1990. Computational Drama in Oz. In *Working Notes of the AAAI-90 Workshop on Interactive Fiction and Synthetic Realities*. Boston, MA.

Bates, J., Loyall, A. B., and Reilly, W. S. 1992a. Integrating Reactivity, Goals, and Emotion in a Broad Agent. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, Indiana, July 1992.

Bates, J., Loyall, A. B., and Reilly, W. S. 1992b. An Architecture for action, emotion, and Social Behavior. In Cristiano Castelfranchi and Eric Werner (Eds.), *Lecture note in Artificial Intelligence, Artificial Social Systems: 4th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, MAAMAW '92. New York, NY: Springer Verlag.

Beer, R. D. 1995. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72, 173-215.

Bernstein, M. 2003 (forthcoming). Card Shark and Thespis. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Bledsoe, W. 1961. The use of biological concepts in the analytical study of systems. In *Proceedings of the ORSA-TIMS National Meeting*, San Francisco.

Blumberg, B. 1996. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. Ph.D. Dissertation. MIT Media Lab.

Blumberg, B. and Galyean, T. 1995. Multi-level Direction of Autonomous Creatures for Real-Time Virtual Environments. *In Proceedings of SIGGRAPH 95*.

Boal, A. 1985. *Theater of the Oppressed*. New York: Theater Communications Group.

Boehlen, M., and Mateas, M. 1998. Office Plant #1: Intimate space and contemplative entertainment. *Leonardo*, Volume 31 Number 5: 345-348.

Bringsjord, S., and Ferrucci, D. 2000. *Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, a Storytelling Machine*. Mahwah, NJ: Lawrence Erlbaum.

Brooks, R. 1991. *Intelligence Without Reason*, A.I. Memo 1293. Artificial Intelligence Lab. MIT.

Brooks, R. 1990. Elephants Don't Play Chess. *Robotics and Autonomous Systems* 6: 3-15.

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1).

Burke, R., Isla, D., Downie, M., Ivanov, Y., and Blumberg, B. 2001. CreatureSmarts: The Art and Architecture of a Virtual Brain. In the *Proceedings of the Game Developers Conference*, pp. 147-166, San Jose, CA.

Burnham, J. 1968. *Beyond Modern Sculpture: The Effects of Science and Technology on the Sculpture of This Century*. New York: G. Braziller.

Callaway J., and Lester J. 1995. Robust Natural Language Generation from Large-Scale Knowledge Bases. In *Proceedings of the Fourth Bar-Ilan Symposium on Foundations of Artificial Intelligence*, pp. 96-105, Jerusalem, Israel.

Carbonell, J. 1979. *Subjective understanding: Computer models of belief systems*. Ph.D. Dissertation, Computer Science Department, Yale University.

Cassell, J., Vilhjálmsson, H., Chang, K., Bickmore, T., Campbell, L. and Yan, H. 1999. Requirements for an Architecture for Embodied Conversational Characters. In Thalmann, D. and Thalmann, N. (eds.), *Computer Animation and Simulation '99*. Vienna, Austria: Springer Verlag.

Cavazza, M., Charles, F., and Mead, S. J. 2002. Sex, Lies and Videogames: an Interactive Storytelling Prototype. In Forbus, K., and El-Nasr, M. S. (Eds.), *Proceedings of the AAAI 2002 Symposium on Artificial Intelligence and Interactive Entertainment*, 13-17.

Chalmers, D. 1996. *The Conscious Mind: In Search of a Fundamental Theory*. New York and Oxford: Oxford University Press.

Chapman, D. 1990. *Vision, Instruction, and Action*. Ph.D. Dissertation, MIT Artificial Intelligence Lab. Technical report 1204.

CogSci. 1993. Special Issue on Situated Cognition. *Cognitive Science* 17 (1993).

Cohen, P. and Levesque, H. 1991. Teamwork. *Nous*, 35.

Colby, B. N. 1973. A partial grammar of Eskimo folktales. *American Anthropologist* 75:645–662.

Colby, K. M. 1975. *Artificial Paranoia: A Computer Simulation of Paranoid Processes*. New York: Pergamon Press.

Cope, D. 1996. *Experiments in Musical Intelligence*. Madison, WI: A-R Editions.

Crawford, C. 2002. Assumptions underlying the Erasmatron storytelling system. Forthcoming in M. Mateas and P. Sengers (Eds.), *Narrative Intelligence*. Amsterdam: John Benjamins.

Crawford, C. 1993. Fundamentals of Interactivity. *Interactive Entertainment Design*, Vol. 7, http://www.erasmatazz.com/library/JCGD_Volume_7/Fundamentals.html.

Crawford, C. 1992. A Better Metaphor for Game Design: Conversation. *The Journal of Computer Game Design,* Vol. 6, http://www.erasmatazz.com/library/JCGD_Volume_6/Conversational_Metaphor.html.

Cullingford, R. (1981). SAM. In Roger Schank and Christopher Riesbeck (Eds.), *Inside Computer Understanding: Five Programs Plus Miniatures* (75-119). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Dario, P., Paggetti, C., Troisfontaine, N., Papa, E., Ciucci, T., Carrozza, M.C., & Marcacci, M. A. 1997. Miniature Steerable End-Effector for Application in an Integrated System for Computer-Assisted Arthroscopy. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, NM., April 1997: 1573-1579.

Davenport, G., Murtaugh, M. 1995. ConText: Towards the Evolving Documentary. In *ACM Multimedia '95*, November.

Dennett, D. 1998a. The Logical Geography of Computational Approaches: A View from the East Pole. In *Brainchildren: Essays on Designing Minds*, 215-234. Cambridge, MA: The MIT Press (A Bradford Book). Originally appeared in Brand, M. and Harnish, M. (Eds.), *The Representation of Knowledge and Belief*. Tuscon: University of Arizona Press, 1986.

Dennet, D. 1998b. When Philosophers Encounter Artificial Intelligence. In *Brainchildren: Essays on Designing Minds*, 265-276. Cambridge, MA: The MIT Press (A Bradford Book). Originally appeared in *Daedalus: Proceedings of the American Academy of Arts and Sciences* 117 (1), Winter 1988.

Dennett, D. 1997. Cog as a Thought Experiment. *Robotics and Autonomous Systems*, 20(2-4), August 1997: 251-256.

Domike, S.; Mateas, M.; and Vanouse, P. 2002. The recombinant history apparatus presents: Terminal Time. Forthcoming in M. Mateas and P. Sengers (Eds.), *Narrative Intelligence*. Amsterdam: John Benjamins.

Douglas, J. Y. 2000. *The End of Books — Or Books Without End?: Reading Interactive Narratives.* Ann Arbor: University of Michigan Press.

Doyle, P. 2002. Believability through context: Using "knowledge in the world" to create intelligent characters. In *Proceedings of the International Joint Conference on Agents and Multi-Agent Systems (AAMAS 2002)*, 342-349, ACM Press, Bologna, Italy, July 2002.

Doyle, P., and Hayes-Roth, B. 1998. Agents in Annotated Worlds. In the *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN, May 1998.

Dreyfus, H. 1999. *What Computer Still Can't Do: A Critique of Artificial Reason*. MIT Press, original edition published in 1972.

Dyer, M. 1983. *In Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*. Cambridge, MA: MIT Press.

Egri, L. 1946. *The Art of Dramatic Writing: Its Basis in the Creative Interpretation of Human Motives*. Simon and Schuster.

Elliott and Pfenning. 1991. A semi-functional implementation of a higher-order logic programming language. In Peter Lee (Ed.), *Topics in Advanced Language Implementation*, pages 289-325. Cambridge MA: MIT Press.

Emerald, P., and Kadrmas, K. 1997. Controller IC and Power Multichip Module Yield Smart, Compact Stepper Controller. *PCIM*, April, 1997: 40-55.

Emerald, P., Sasaki, M., & Takahashi, H. 1996. CMOS Step Motor IC and Power Multi-Chip Module Combine to Integrate Versatile, Multi-Mode PWM Operation and Microstepping. In *Proceedings of Powersystems World 96*, Las Vegas, NV., Sept. 1996.

Eskelinen, M. 2001. Towards Computer Game Studies. In *Proceedings of SIGGRAPH 2001, Art Gallery, Art and Culture Papers*: 83-87.

Fellbaum, C. (Ed.). 1998. Wordnet: An Electronic Lexical Database. MIT Press.

Ferguson, G., Allen, J. F., Miller, B. W., and Ringger, E. K. 1996. *The design and implementation of the TRAINS-96 system: A prototype mixed-initiative planning assistant*. TRAINS Technical Note 96-5, Department of Computer Science, University of Rochester, Rochester, NY.

Forgy, C. L. 1982. Rete: A Fast Algorithm for the Many Pattern/ Many Object Pattern Match Problem. *Artificial Intelligence* 19, 17-37.

Frasca, G. 2003 (forthcoming). Frasca response to Mateas. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Frasca, G. 2001. *Videogames of the Oppressed: Videogames as a Means for Critical Thinking and Debate*. Masters Thesis, Interactive Design and Technology Program, Georgia Institute of Technology. Available at: www.ludology.org.

Friedman-Hill, E. 1995-2003. Jess, the Rule Engine for Java. Sandia National Labs. http://herzberg.ca.sandia.gov/jess/.

Galyean, T. 1995. *Narrative Guidance of Interactivity*. Ph.D. Dissertation, MIT Media Lab, MIT.

Gibson, J. 1979. *The ecological approach to human perception*. Boston: Houghton Mifflin.

Gibson, J. 1977. The theory of affordances. In R. E. Shaw & J. Bransford (Eds.), Perceiving, acting, and knowing. Hillsdale, NJ: Erlbaum Associates.

Grand, S. 2001. *Creation: Life and How to Make It*. Cambridge, MA: Harvard University Press.

Gratch, J., and Marsella, S. 2001. Tears and Fears: Modeling emotions and emotional behaviors in synthetic agents. In *Proceedings of the 5th International Conference on Autonomous Agents*. Montreal, Canada, June 2001.

Grosz, B. and Kraus, S. 1996. Collaborative plans for complex group actions. *Artificial Intelligence*, 86, 269 - 358.

Hayes-Roth, B., van Gent, R. and Huber, D. 1997. Acting in character. In R. Trappl and P. Petta (Eds.), *Creating Personalities for Synthetic Actors*. Berlin, New York: Springer.

Hayes-Roth, B., and van Gent, R. 1996. *Story Making with Improvisational Puppets and Actors*, Technical Report, KSL-96-05, Stanford Knowledge Systems Laboratory, Stanford Univ.

Hayles, N. K. 1999. *How We Become Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics*. Chicago, IL: The University of Chicago Press.

Haugeland, J. 1985. *Artificial Intelligence: The Very Idea*. Cambridge, MA: The MIT Press (A Bradford Book).

Herold, C. 2002. A Streak of Glamour but a Lack of Lifeblood. *New York Times, Game Theory Column*, Sept. 5, 2002.

Hjelmslev, L. 1961. *Prolegomena to a Theory of Language* (trans. Francis J Whitfield). Madison: University of Wisconsin Press. Translation of *Omkring Sprogteoriens Grundlæggelse,* first published in 1943.

Holland, J. 1962. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 3, 297-314.

Hovy. 1987. *Generating Natural Language Under Pragmatic Constraints*. Ph.D. Dissertation, Computer Science Department, Yale University, Research Report #521.

Howe, N., and Strauss B. 1993. *13th Gen: Abort, Retry, Ignore, Fail?*. New York, NY: Vintage Books.

Huhtamo, E. 1998. Silicon Remembers Ideology, or David Rokeby's Meta-Interactive Art. Catalog essay for *The Giver of Names* exhibit at McDonald-Stewart Art Center. Available online at: http://www.interlog.com/~drokeby/erkki.html.

Ihnatowicz, E. 1986. *Cybernetic Art: A Personal Statement*. Self-published. http://members.lycos.co.uk/zivanovic/senster/ihnatowicz brochure.pdf.

Jenkins, H. 2003 (forthcoming). Game Design as Narrative Architecture. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Johnson, M. P., Wilson, A., Kline, C., Blumberg, B., & Bobick, A. 1999. Sympathetic Interfaces: Using a Plush Toy to Direct Synthetic Characters. In *Proceedings of SIGCHI 1999*.

Jones, C. 1989. *Chuck Amuck: The Life and Times of an Animated Cartoonist*. Farrar, Straus and Giroux.

Kahn, K. 1979. *Creation of computer animation from story descriptions*. Ph.D. Dissertation. MIT. AI technical report 540.

Kantrowitz, M., and Bates, J. 1992. Integrated Natural Language Generation Systems. In R. Dale et al. (Eds.), *Lecture Notes in Artificial Intelligence #587, Aspects of Automated Natural Language Generation (Proceedings of the Sixth International Workshop on Natural Language Generation)*. New York: Springer-Verlag.

Kelso, M., Weyhrauch, P., Bates, J. 1993. Dramatic Presence. *Presence: The Journal of Teleoperators and Virtual Environments*, Vol. 2 No. 1, MIT Press.

Kirsch, J. L., and Kirsch, R. A. 1988. The Anatomy of Painting Style: Description with Computer Rules. *Leonardo* 21(4): 437-444.

Kolodner, J. 1984. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Kosko, B. 1997. *Fuzzy Engineering.* New York: Simon & Schuster.

Lamport, L. 1978. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*. 21(7):558-565, July 1978.

Landow, G. 1992. *Hypertext: The convergence of contemporary critical theory and technology*. Baltimore, MD: John Hopkins University Press.

Lang, R. 2002. Story Grammars: Return of a Theory. Forthcoming in M. Mateas and P. Sengers (Eds.), *Narrative Intelligence*. Amsterdam: John Benjamins.

Lang, R. 1999. A Declarative Model for Simple Narratives. In M. Mateas and P. Sengers (Eds.), *Working notes of the Narrative Intelligence Symposium*, AAAI Spring Symposium Series. Menlo Park: Calif.: AAAI Press.

Laurel, B. 2003 (forthcoming). Laurel response to Mateas. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Laurel, B. 1991. *Computers as Theatre*. Reading, MA: Addison-Wesley.

Laurel, B. 1986. *Towards the Design of a Computer-Based Interactive Fantasy System*. Ph.D. Dissertation., The Ohio State University.

Lauzzana, R. G., and L. Pocock-Williams. 1988. A Rule System for Analysis in the Visual Arts. Leonardo 21(4): 445-452.

Lavie, A. and Tomita, M. 1996. GLR* - An Efficient Noise-Skipping Parsing Algorithm for Context-Free Grammars. In H. Bunt and M. Tomita (Eds.), *Recent Advances in Parsing Technology*, Text Speech and Language Technology series (vol. 1). Kluwer Academic Press.

Lebowitz, M. 1985. Story Telling as Planning and Learning. *Poetics* 14, pp. 483-502.

Lebowitz, M. 1984. Creating Characters in a Story-Telling Universe. *Poetics* 13, pp. 171-194.

Lenat, D. 1995. Cyc: A Large-Scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38, no. 11, November.

Lesh, N., Rich, C., and Sidner, C. 1999 Using Plan Recognition in Human-Computer Collaboration. In *Proceedings of the Seventh International Conference on User Modeling*. Banff, Canada.

Lester, J., Stone, B. 1997. Increasing Believability in Animated Pedagogical Agents. *Proceedings of the First International Conference on Autonomous Agents*. Marina del Rey, CA, USA, 16-21.

Lester, J., Voerman, J., Towns, S., and Callaway, C. 1999. Deictic Believability: Coordinating Gesture, Locomotion, and Speech in Lifelike Pedagogical Agents. *Applied Artificial Intelligence*. 13(4-5), pp. 383-414.

Loh, K. 1993. Plot summary for *Remains of the Day*, Internet Movie Database, http://us.imdb.com/.

Loyall, A. B. 1997. *Believable Agents*. Ph.D. Dissertation, Tech report CMU-CS-97-123, Carnegie Mellon University.

Loyall, A. B. and Bates, J. 1997. Personality-Rich Believable Agents That Use Language. In *Proceedings of the First International Conference on Autonomous Agents*. Marina del Rey, CA.

Loyall, A. B. and Bates, J. 1993. Real-time Control of Animated Broad Agents. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*. Boulder, CO.

Loyall, A. B. and Bates, J. 1991. Hap: A Reactive, Adaptive Architecture for Agents. Technical Report CMU-CS-91-147. Department of Computer Science. Carnegie Mellon University.

Magnuson, W. 1995-2002. English Idioms, Sayings, and Slang. http://home.t-online.de/home/toni.goeller/idiom_wm/.

Manurung, H. M., Ritchie, G., Thompson, H. 2000. Towards a Computational Model of Poetry Generation. In *Proceedings of AISB Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*, 79-86, Birmingham, April 2000.

Marsella, S. 2000. Pedagogical Soap. In K. Dautenhahn (Ed.), *Working Notes of the Socially Intelligent Agents Symposium: The Human in the Loop*. AAAI Fall Symposium Series. Menlo Park, CA: AAAI Press.

Marsella S., Johnson W. L., LaBore, C. 2000. Interactive Pedagogical Drama. In *Proceedings of Autonomous Agents 2000*. Barcelona, Spain.

Masterson, M. 1971. Computerized haiku. In J. Reichardt (Ed.), Cybernetics, Art and Ideas. Greenwich, CT: New York Graphic Society Ltd. 175-183.

Mateas, M. 2003a (forthcoming). A Preliminary Poetics for Interactive Drama and Games. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Mateas, M. 2003b (forthcoming). Mateas response to Frasca. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Mateas, M. 2003c (forthcoming). Mateas response to Laurel. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Mateas, M. 2001a. Expressive AI. *Leonardo: Journal of the International Society for Arts, Sciences, and Technology*, 34 (2), 147-153.

Mateas, M. 2001b. A preliminary poetics for interactive drama and games. In *Proceedings of SIGGRAPH 2001, Art Gallery, Art and Culture Papers*, 51-58.

Mateas, M. 2000a. Expressive AI. In *Electronic Art and Animation Catalog, Art and Culture Papers, SigGraph 2000*. New Orleans, LA.

Mateas, M. 2000b. A Neo-Aristotelian Theory of Interactive Drama. In *Working notes of the AI and Interactive Entertainment Symposium*, AAAI Spring Symposium Series. Menlo Park, CA.: AAAI Press.

Mateas, M. 1999a. Not your Grandmother's Game: AI-Based Art and Entertainment. *Working notes of the AI and Computer Games Symposium*, AAAI Spring Symposium Series. Menlo Park: Calif.: AAAI Press.

Mateas, M. 1999b. An Oz-Centric Review of Interactive Drama and Believable Agents. In M. Wooldridge and M. Veloso, (Eds.), *AI Today: Recent Trends and Developments*. Lecture Notes in AI 1600. Berlin, New York: Springer. First appeared in 1997 as Technical report CMU-CS-97-156. Computer Science Department, Carnegie Mellon University.

Mateas, M. 1998. Subjective Avatars (poster). In *Proceedings of the Second International Conference on Autonomous Agents*, pp. 461-462.

Mateas, M. 1997. Computational Subjectivity in Virtual World Avatars. *Working notes of the Socially Intelligent Agents Symposium, AAAI Fall Symposium Series*. Menlo Park: Calif.: AAAI Press.

Mateas, M and Sengers, P (Eds.). 2002 (Forthcoming). *Narrative Intelligence*. Amsterdam: John Benjamins.

Mateas, M and Stern, A. 2002. Towards Integrating Plot and Character for Interactive Drama. In K. Dautenhahn, A. Bond, L. Canamero, and B. Edmonds (Eds.), *Socially Intelligent Agents: Creating Relationships with Computers and Robots*. Norwall, MA: Kluwer Academic Publishers.

Mateas, M. and Stern, A. 2000. Towards Integrating Plot and Character for Interactive Drama. In *Working notes of the Social Intelligent Agents: The Human in the Loop Symposium*. AAAI Fall Symposium Series. Menlo Park, CA.: AAAI Press.

Mateas, M., Vanouse, P., and Domike S. 2000. Generation of Ideologically-Biased Historical Documentaries. In *Proceedings of AAAI 2000*. Austin, TX, pp. 236-242.

Mateas, M., Vanouse, P., Domike S. 1999. Terminal Time: An Ideologically-biased History Machine. *AISB Quarterly, Special Issue on Creativity in the Arts and Sciences*, Summer/Autumn 1999, No. 102, 36-43.

McCarthy, J. 1968. Programs with Common Sense. In Marvin Minsky (Ed.), *Semantic Information Processing*, pp. 403-418. Cambridge, MA: The MIT Press.

McCarthy, J. and Hayes, P. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie (Eds.), *Machine Intelligence*, Vol. 4. Edinburgh: Edinburgh University Press.

McCorduck, P. 1991. *Aaron's Code: Meta-art, Artificial Intelligence, and the Work of Harold Cohen*. New York, NY: W. H. Freeman and Co.

McKee, R. 1997. *Story: Substance, Structure, Style, and the Principles of Screenwriting*. New York, NY: HarperCollins.

Meehan, J. 1976. *The Metanovel: Writing Stories by Computer*. Ph.D. Dissertation. Yale University.

Mitchell, T. 1997. *Machine Learning*. New York: McGraw-Hill.

Mojarrad. 1997. Biomimetic Robotic Propulsion Using Polymeric Artificial Muscles. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Albuquerque, NM., April 1997: 2152-2157.

Montfort, N. 2003 (forthcoming). *Twisty Little Passages: An Approach to Interactive Fiction*. Cambridge, MA: MIT Press.

Moreno, R., Mayer, R., Lester, J. 2000. Life-Like Pedagogical Agents in Constructivist Multimedia Environments: Cognitive Consequences of Their Interaction. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia, and Telecommunications (ED-MEDIA)*, pp. 741-746, Montreal.

Mueller, E. 1990. *Daydreaming in Humans and Machines*. Norwood, NJ: Ablex.

Murray, J. 1998. *Hamlet on the Holodeck*. Cambridge, MA: MIT Press.

NASA. 1985-2002. C Language Integrated Production Systems (CLIPS). Originally developed at NASA's Johnson Space Center. http://www.ghg.net/clips/WhatIsCLIPS.html.

Neal Reilly, W. S. 1996. *Believable Social and Emotional Agents*. Ph.D. Dissertation., School of Computer Science, Carnegie Mellon University.

Newell, A. 1990. *Unified Theories of Cognition: The William James Lectures 1987*. Cambridge MA: Harvard University Press.

Newell, A. 1982. The Knowledge Level. *Artificial Intelligence* 18: 87-127.

Newell, A., Rosenbloom, P., & Laird, J. 1989. Symbolic architectures for cognition. In M. Posner (ed.), *Foundations of Cognitive Science*. Cambridge, MA: MIT Press, Bradford Books.

Newell, A. and Simon, H. 1976. Computer science as empirical enquiry: symbols and search. *Communications of the ACM*, 19:113--126.

Norman, D. 1988. *The Design of Everyday Things*. New York: Doubleday.

Oliver, D. 1995-2002. The ESL Idiom Page. http://www.eslcafe.com/idioms/id-mngs.html.

OMM 2001. Who Killed Adventure Games? *Old Man Murray*. http://web.archive.org/web/20010417025123/www.oldmanmurray.com/features/doa/page1.shtml.

Ortony, A., Clore, A., and Collins, G. 1988. *The Cognitive Structure of Emotions*. Cambridge, England: Cambridge University Press.

Pask. G. 1971. A comment, a case history, and a plan. In J. Reichardt (Ed.), Cybernetics, Art and Ideas. Greenwich, CT: New York Graphic Society Ltd. 76-99.

Penny, S. 2000. Agents as Artworks and Agent Design as Artistic Practice. In K. Dautenhahn (Ed.), *Human Cognition and Social Agent Technology*. Amsterdam: John Benjamins.

Penny, S. 1999. Systems Aesthetics and Cyborg Art: The Legacy of Jack Burnham. *Sculpture* 18(1), January/February 1999.

Penny, S. 1997. Embodied Cultural Agents at the Intersection of Robotics, Cognitive Science and Interactive Art. In K. Dautenhahn (Ed)., *Working notes of the Socially Intelligent Agents Symposium*. AAAI Fall Symposium Series. Menlo Park: Calif.: AAAI Press.

Penny, S. 1995. Consumer Culture and the Technological Imperative: The Artist in Dataspace. In S. Penny (Ed.), *Critical Issues in Electronic Media*, SUNY Press.

PhraseFinder. 2002. Phrases, Sayings, Quotes and Cliches at The Phrase Finder. http://phrases.shu.ac.uk/index.html.

PhraseThesaurus. 2002. Phrase Thesaurus. http://www.phrasefinder.co.uk/index.html.

Pinhanez, C. 1997. Interval Scripts: a Design Paradigm for Story-Based Interactive Systems. In *Proceedings of CHI97*. Atlanta, GA, pp. 287-294.

Poesio, M., and Traum, D. 1997. Conversational Actions and Discourse Situations. *Computational Intelligence*. Vol. 13, No. 3.

Porter, M. 1980. An algorithm for suffix stripping. *Program*, 14(3) :130-137.

Rich, C., and Sidner, C. 1998. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction*. Vol. 8, No. 3/4, pp. 315-350.

Rickel, J. and Johnson, L. 1998. Animated agents for procedural training in virtual reality: perception, cognition, and motor control. *Applied Artificial Intelligence* (13), 343-382.

Rickman, B. 2002 (forthcoming). The Dr. K – Project. In M. Mateas and P. Sengers (Eds.), *Narrative Intelligence*. Amsterdam: John Benjamins.

Roads, C. (1985). Improvisation with George Lewis. In C. Roads (Ed.), *Composers and the Computer*, 75-88. William Kaufmann, Inc.

Rosenberg, J. 1998. Locus Looks at the Turing Play: Hypertextuality vs. Full Programmability. In *Hypertext 98: The Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia*, ACM, New York, 152-160.

Rosenblatt, F. 1959. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386-408.

Rousseau, D. and Hayes-Roth, B. 1998. A Social-Psychological Model for Synthetic Actors. In the *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN, May 1998.

Rumelhart, D. E. 1975. Notes on a Schema for Stories. In Bobrow, D. G., and Collins, A., (Eds.), *Representation and Understanding*. Academic Press. 211–236.

Ryan, M. 2001. Beyond Myth and Metaphor – The Case of Narrative in Digital Media. *Game Studies: The International Journal of Computer Game Research*, Vol. 1, Issue 1. Available at: http://www.gamestudies.org/0101/.

Sack, W. 2002 (forthcoming). Stories and Social Networks. In M. Mateas and P. Sengers (Eds.), *Narrative Intelligence*. Amsterdam: John Benjamins.

Sack, W. 2000a. Actor-Role Analysis: Ideology, Point of View and the News, in *Narrative Perspectives: Cognition and Emotion*, Chatman S., Van Peer, W. (editors), New York: SUNY Press.

Sack, W. 2000b. *Design for Very Large-Scale Conversations*. Ph.D. Dissertation, MIT Media Lab.

Saffiotti A., Konolige, K. & Ruspini, E. H. 1995. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence*. Vol. 76, No. 1-2. pp. 481-526.

Salton, G., Buckley C. 1988. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, Vol. 24 No. 5: 513-523.

Samuel. A. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3, 211-229.

Saussure, F. 1974. *Course in General Linguistics*. London: Fontana. Translation of *Cours de linguistique generale*, first published in 1916.

Schank, R. and Reisbeck, C. (Eds.). 1981. *Inside Computer Understanding: Five Programs Plus Miniatures*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Schiffer. S. 1999. The Rise and Fall of Black Velvet Flag: An "Intelligent" System for Youth Culture Documentary. In M. Mateas and P. Sengers (Eds.), *Working Notes*

of the AAAI Fall Symposium on Narrative Intelligence. AAAI Fall Symposium Series. Menlo Park, CA: AAAI Press..

Sengers, P. 2000. Panel: Fiction 2001, SIGGraph 2000.

Sengers, P. 1999a. Designing Comprehensible Agents. In *Proceedings of the Sixteenth Annual International Joint Conference of Artificial Intelligence*. Stockholm.

Sengers, P. 1999b. Cultural Informatics: Artificial Intelligence and the Humanities. In *Surfaces: Special Issue on Humanities and Computing - Who's Driving?*. Volume 8. Available online at http://www.pum.umontreal.ca/revues/surfaces/vol8/vol8TdM.html.

Sengers, P. 1998a. *Anti-Boxology: Agent Design in Cultural Context*. Ph.D. Dissertation. School of Computer Science, Carnegie Mellon University.

Sengers, P. 1998b. Do the Thing Right: An Architecture for Action-Expression. In *Proceedings of the Second International Conference on Autonomous Agents*. pp. 24-31.

Sgouros, N. M. 1999. *Dynamic generation, management and resolution of interactive plots*. Artificial Intelligence 107 (1999): 29-62.

Sleator D., and Temperley, D. 1993. Parsing English with a Link Grammar. In *Third International Workshop on Parsing Technologies*.

Smith, J. H. 2000. The Dragon in the Attic – On the Limits of Interactive Fiction. http://www.game-research.com/art_dragon_in_the_attic.asp.

Smith, S., and Bates, J. 1989. *Towards a Theory of Narrative for Interactive Fiction*, Technical Report, CMU-CS-89-121, Dept. of Computer Science, Carnegie Mellon Univ.

Smolensky, P. 1998. Connectionism, Constituency, and the Language of Thought. In A. Clark and J. Toribio (Eds.), *Cognitive Architectures in Artificial Intelligence: The Evolution of Research Programs*. New York & London: Garland Publishing Inc.

Stern, A. 2003 (forthcoming). Stern response to Bernstein. In N. Wardrip-Fruin and P. Harrigan. (Eds.), *First Person: New Media as Story, Performance and Game*. Cambridge MA: The MIT Press.

Stern, A. 2002 (forthcoming). Virtual Babyz: Believable Agents with Narrative Intelligence. Forthcoming in M. Mateas and P. Sengers (Eds.), *Narrative Intelligence*. Amsterdam: John Benjamins.

Stern, A. 2001. Deeper conversations with interactive art, or why artists must program. *Convergence*, Vol. 7, No. 1, Spring 2001.

Stern, A.; Frank, A.; and Resner, B. 1998. Virtual Petz: A hybrid approach to creating autonomous, lifelike Dogz and Catz. In *Proceedings of the Second International Conference on Autonomous Agents*, 334-335. Menlo Park, Calif.: AAAI Press.

Stern, A. 1999. Virtual Babyz: Believable Agents with Narrative Intelligence. In M. Mateas and P. Sengers (Eds.), *Working Notes of the 1999 AAAI Spring Symposium on Narrative Intelligence*. AAAI Press.

Swartout, W., Hill, R., Gratch, J., Johnson, W. L., Kryakakis, C., LaBore, C., Lindheim, R., Marsella, S., Miraglia, D., Moore, B., Morie, J., Rickel, J., Thiebaux, M., Tuch, L., Whitney, R., Douglas, J. 2001. Toward the Holodeck: Integrating Graphics, Sound, Character, and Story. In *Proceedings of the 5th International Conference on Autonomous Agents*.

Tambe, M. 1997. Towards Flexible Teamwork. *Journal of Artificial Intelligence Research* (7) 83-124.

Tesauro, G. J. 1995. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38:58-68.

Thorison, K. 1996. *Communicative Humanoids: A Computational Model of Psychosocial Dialogue Skills*. Ph.D. Dissertation. MIT Media Laboratory.

Thom, B. 2001. *BoB: An Improvisation Music Companion*. Ph.D. Dissertation, Department of Computer Science, Carnegie Mellon University.

Thomas, F. and Johnston, O. 1981. *The Illusion of Life: Disney Animation*. Hyperion.

Tosa, N. 1993. Neuro Baby. *Siggraph '93 Visual Proceedings*: 167.

Traum, D. 1999a. Speech Acts for Dialog Agents. In M. Wooldride and A. Rao (Eds.), *Foundations of Rational Agency*. Kluwer. 169-201.

Traum, D. 1999b. 20 Questions for Dialog Act Taxonomies. In *Amstelogue'99 Workshop on the Semantics and Pragmatics of Dialogue*.

Traum, D. and Hinkelman, E. 1992. Conversation Acts in Task-Oriented Spoken Dialog. *Computational Intelligence: Special Issue: Computational Approaches to Non-literal Language* vol. 8, no. 3.

Turing, A. 1950a. Computing machinery and intelligence. *Mind* 59:433-60.

Turner, S. R. 1994. *The Creative Process: A Computer Model of Storytelling and Creativity*. Lawrence Erlbaum Associates.

Turner, S. R. 1991. A case-based model of creativity. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*. Chicago, Illinois.

Veloso, M., Stone, P., & Han, K. 1998. The CMUnited-97 Robotic Soccer Team: Perception and Multiagent Control. In Sycara, K., and Wooldridge, M. (Eds.), *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pp 78-85.

Varela, F., Thompson, E., Rosch, E. 1999. *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press.

Vere, S. 1975. Induction of concepts in the predicate calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pp. 351-356.

Weizenbaum, J. 1976. *Computer Power and Human Reason: From Judgment to Calculation*. W.H.Freeman.

Weizenbaum, J. 1966. Eliza – A computer program for the study of natural language communication between man and machine. *Communications of the ACM* 9(1):36-45.

Weyhrauch, P. 1997. *Guiding Interactive Drama*. Ph.D. Dissertation, Tech report CMU-CS-97-109, Carnegie Mellon University.

Wilensky, R. PAM. In Roger Schank and Christopher Riesbeck (Eds.), *Inside Computer Understanding: Five Programs Plus Miniatures* (136-179). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Wilson, S. 2002. *Information Arts: Intersections of Art, Science and Technology*. Cambridge, MA: The MIT Press.

Wilson 1995. Artificial Intelligence Research as Art. *Stanford Humanities Review*, special issue on *Constructions of the Mind: Artificial Intelligence and the Humanities*, 4(2). Available at http://www.stanford.edu/group/SHR/4-2/text/wilson.html.

Winograd, T., and Flores, F. 1986. *Understand Computers and Cognition: A New Foundation for Design*. Addison-Wesley Publishing.

Winston, P. 1970. *Learning Structural Descriptions from Examples*. Ph.D. Dissertation, MIT. MIT technical report AI-TR-231.

Yoon, S.Y., Blumberg, B., & Schneider, G. 2000. Motivation Driven Learning for Interactive Synthetic Characters. In *Proceedings of Autonomous Agents 2000*.

Young, R. M. 2001. An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment. In *The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*.