Architecture, Authorial Idioms and Early Observations of the Interactive Drama *Façade*

Michael Mateas Andrew Stern

December 5, 2002
CMU-CS-02-198

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15212

Michael Mateas*
Dept. of Computer Science
Carnegie Mellon University
michaelm@cs.cmu.edu

Andrew Stern*
InteractiveStory.net
www.interactivestory.net
andrew@interactivestory.net

*co-authors listed alphabetically

Abstract

Façade is an artificial intelligence-based art/research experiment in electronic narrative – an attempt to move beyond traditional branching or hyper-linked narrative to create a fully-realized, one-act interactive drama. Integrating an interdisciplinary set of artistic practices and artificial intelligence technologies, we are completing a three year collaboration to engineer a novel architecture for supporting emotional, interactive character behavior and drama-managed plot. Within this architecture we are building a dramatically interesting, real-time 3D virtual world inhabited by computer-controlled characters, in which the user experiences a story from a first-person perspective. Façade will be publicly released as a free download in 2003.

Keywords: believable agents, interactive drama, interactive narrative, narrative intelligence

Introduction

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]. 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 [Bates, Loyall and Reilly 1992a, 1992b; Blumberg 1996; Hayes-Roth, van Gent and Huber 1997; Lester and Stone 1997; Stern, Frank, and 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 [Weyrauch 1997; Blumberg and Galyean 1995]. We are currently engaged in a three year collaboration to build an interactive story world, Façade, integrating believable agents and interactive plot. Façade will be publicly released as a free download in 2003.

In Façade, 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. During an evening get-together at their apartment that quickly turns ugly, you become entangled in the high-conflict dissolution of Grace and Trip's marriage. No one is safe as the accusations fly, sides are taken and irreversible decisions are forced to be made. By the end of this intense one-act play you will have 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.

This work is unlike hypertext narrative or "interactive fiction" to date in that the computer characters actively perform the story without waiting for you to click on a link or enter a command. Interaction is seamless as you converse in natural language and move and gesture freely within the first-person 3D world of Grace and Trip's apartment. AI controls Grace and Trip's personality and behavior, including emotive facial expressions, spoken voice and full-body animation. Furthermore, the AI intelligently chooses the next story "beat" based on your moment-by-moment interaction, what story beats have happened so far, and the need to satisfy an overall dramatic arc. Innovative natural language processing allows the system to avoid the "I don't understand" response all too common in text-adventure interactive fiction.

The process of building Façade involves three major research efforts: designing ways to deconstruct a dramatic narrative into a hierarchy of story and behavior pieces, engineering an AI system to reconstruct a real-time dramatic performance from those pieces that integrates the player's moment-by-moment interactions, and understanding how to write an engaging, compelling story within this new organizational framework.

Focus of this paper

As of this writing, the AI infrastructure and animation engine are complete, we have developed a working set of support behaviors and authoring techniques (idioms) within the system, and have finished authoring about 25% of the story content (behaviors, dialog, animation). Getting to this point has involved an iterative process of infrastructure building and throw-away authoring; as we author, we continue refining and adding to the authoring idioms, requiring us to reimplement story content.

The process of developing Façade is yielding lessons about what works and what doesn't in the design and engineering of interactive stories, and is giving us a better understanding of what it will require to create more generative story systems in the future. This mid-project paper limits itself to presenting an overview of the Façade architecture and authoring idioms, including how the architecture's design was motivated by the project design goals. Future papers will address the design lessons learned from the authoring process; however the final section of this paper includes a few early observations of what we already understand about the successes and failures of the system.

Collaboration

The construction of Façade is an equal collaboration between the two authors. We are both intimately involved in the development of the concept including the story and desired player experience. We are both involved in the high-level design of the architecture. We are both authoring 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. Michael is primarily responsible for the detailed (i.e. code-level) design and implementation of the AI architecture. In addition, Michael brings 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, character animation 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, we 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 (Phase I NLP coding), John Rines (additional character animation), and JP Lavin (story consultant).

Review of the project design goals

First we will review our design goals for Façade, originally published in [Mateas and Stern 2000, 2002a], with some thoughts on how they motivate the design of the architecture.

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 very 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 6 or 7 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 which 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. Ultimately, this is the only way to validate the ideas.

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 about 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 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. Ideally 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.

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.

Architecture design motivation

Before starting the tour of the architecture, here are some thoughts on how the system's design is motivated by the project design goals.

To date there have been two general approaches towards creating interactive narrative experiences. One approach is to hand-craft a structure of nodes, often in the form of a graph, network or flowchart, where each node is a finely-crafted chunk of content such as a plot event, information about a character, or a discrete location in an environment. The connections between nodes are often called paths or links. Typically a node connects with a small number of other nodes. The player is given the ability to traverse the graph, and the resulting sequence of nodes constitutes the experience of the narrative. Depending on the complexity of the interconnectedness between the nodes, the range of traversals through the structure can range anywhere from very limited and coherent to very numerous and fragmented, even cyclical and never-ending. Examples include the plot structure of action / adventure games, hypertext fiction, some text-based interactive fiction, and choose-your-own-adventure books.

Another approach is to create a procedural simulation – an open-ended virtual world containing a collection of independent elements, such as objects, environments and (often simplistic) autonomous agents, e.g., NPC's. Each element maintains its own state and has procedures governing its behavior – the different ways it can act upon and react to other elements in the world. The player is just another element in the world. As the simulation runs, all elements run in parallel, allowing many things to happen simultaneously. In its purest form, there is no particular pacing or explicit structure imposed on the experience; the possibilities are only limited by the combinatorics of the range of actions and reactions between the elements in the world. (A slightly more constrained form of simulation offers the player optional "goals" or "missions" to strive for, with a variety of ways to achieve them.) The player experiences a sequence of events over time, akin to how one experiences real life, which may or may not be interpreted as "narrative" by the player, perhaps depending on how closely the sequence of events happens to resembles a narrative structure. When this happens it is called "emergent narrative". Examples include levels in a first person shooter, sim games, "immersive simulation" games, virtual reality and virtual worlds (graphical or text-based, offline or online).

Façade is an attempt to find a capable middle ground between structured narrative and simulation. We want to combine the strengths and minimize the weaknesses of each approach.

The strength of the structured narrative approach, and what is lacking from simulations, is that the system (if so designed) can offer the player a well-formed experience. This means the experience is unified, where all parts of the experience were necessary to contribute to a unified whole with little or no extraneous action, and the experience is efficient and well-paced, where the experience does not take an inordinate amount of time or labor for the player, stays interesting and never lags or gets boring (not everyone wants to spend 40+ hours at the computer to get a complete experience). The tension of the experience may even be made to rise and fall at a pace to match a Aristotelian dramatic arc. These time-tested qualities of unity, efficiency and pacing are part of what makes good narratives so pleasurable, or at least can make them unpleasurable if missing.

The strength of simulations, and what is lacking from structured narratives, is that the player has a high degree of agency and freedom of expression at all times. Many things can happen at any time; the space of possibilities is often an order of magnitude or more larger than what is possible in even a complexly-tangled narrative graph structure. This degree of agency is part of what makes the best simulation games so pleasurable, or unpleasurable when missing.

On moment-by-moment basis, Façade is a simulation. It has a simulated virtual world with objects, the behavior-based autonomous agents Grace and Trip, and the Guest character controlled by the human player. In the moment, the simulation offers a high degree of freedom and local agency to the player, and is where the *character* (personality, emotion, lifelikeness) of the believable agents is experienced first-hand. Beyond what a pure simulation contains, however, is an additional invisible agent called the *drama manager*. The drama manager continuously monitors the simulation and proactively adds and retracts procedures (behaviors) by which Grace and Trip operate. That is, the rules of the simulation are regularly being updated in an attempt to give the player a well-formed overall experience with unity, efficiency and pacing. These simulation updates are organized into story *beats*, each a collection of behaviors tailored to a particular situation or context but still offering a non-trivial simulation space. Beats are annotated by the author with preconditions and effects on the story state, instructing the drama manager when they make sense to use, in the interest of creating an overall dramatic narrative – a *plot*. These preconditions and effects serve to specify a *partial ordering* of beat sequences. So at a high level, Façade's collection of beats and sequencing rules effect a dynamic, flexible version of a structured narrative graph – specifically, a network with more than just a few links between each node, where links can come and go dynamically, and always forward-flowing (no cycles or backtracks).

It is worth noting that the plot structures effected from the dynamic sequencing of a collection of beats is not capturable in a single directed network or flowchart diagram (see [Bernstein 1998] and [Ryan 2001] for examples of such diagrams); perhaps the only way to visualize it is to enumerate all possible orderings, technically in the thousands. The more beats there are for the drama manager to work with, the more possible orderings that emerge, and therefore the more global agency (plot control) the player will experience. See Figure 4 in the Drama Management section of this paper for an illustration of beat sequencing.

How are beats different than, say, levels in a first-person shooter or "immersive simulation" game? Don't game levels also update the rules of the game simulation – also a middle ground between structured narrative and simulation? The differences between Façade beats and game levels have to do with grain size, the number of possible coherent orderings, the degree of update per change in the simulation, and the seamlessness between changes. In Façade, beats are changing every minute or so, are chosen from a pool of ~200 beats, and by design can occur in many different orders while still maintaining narrative coherence. Game levels typically change every 10-15 minutes at most, are chosen from a small pool of levels, and typically cannot occur in many different orders while maintaining narrative coherence. Also, beats change the overall behavior of the simulation to a greater degree than game levels tend to. Each beat has a custom *context* in which saying the same words or doing the same actions as before may now yield a very different result than they did in previous beats. In a typical game level, the environment may have changed from the previous levels, but shooting the same weapon or performing the same action tends to have the same general effect on the world as it did before. Furthermore, beats in Façade are sequenced together seamlessly allowing for continuous, uninterrupted immersion in the narrative, without the break in time or place typical between game levels.

The motivation to find a middle ground between structured narrative and simulation manifests itself in Façade in another key way: the architecture offers direct support for coordinated activity between the autonomous agents in the simulation in the form of *joint goals and behaviors*. This allows the author to more readily create sophisticated, lifelike coordinated behavior between dramatic characters than is otherwise possible in a strongly autonomous agent architecture. See [Mateas and Stern 2000, 2002a] for further discussion of this issue.

Finally, we are motivated by the belief that building a *whole system* containing all the required pieces to achieve a complete user experience – a short but "fully-realized" interactive drama – forces us to address issues that otherwise get ignored or swept under the rug when developing only a piece of an architecture. Furthermore, building a complete experience allows us to release the work into the world for people to play with and critique, compelling us to put significant energy into the quality of the writing and story design.

Tour of the architecture and authorial idioms

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 our knowledge this is the first published architecture to integrate all these pieces[1]. Façade's primary architectural contribution, besides achieving the integration itself, is architectural support for authoring dramatic beats, an architectural level which combines aspects of character and story.

High-level overview of the system

The major components of the architecture are (see Figure 1):

- the 3D story world and its first-person user interface
- the computer-controlled believable agents that operate within the world
- the natural language processing system
- the drama (beat) manager and its story memory

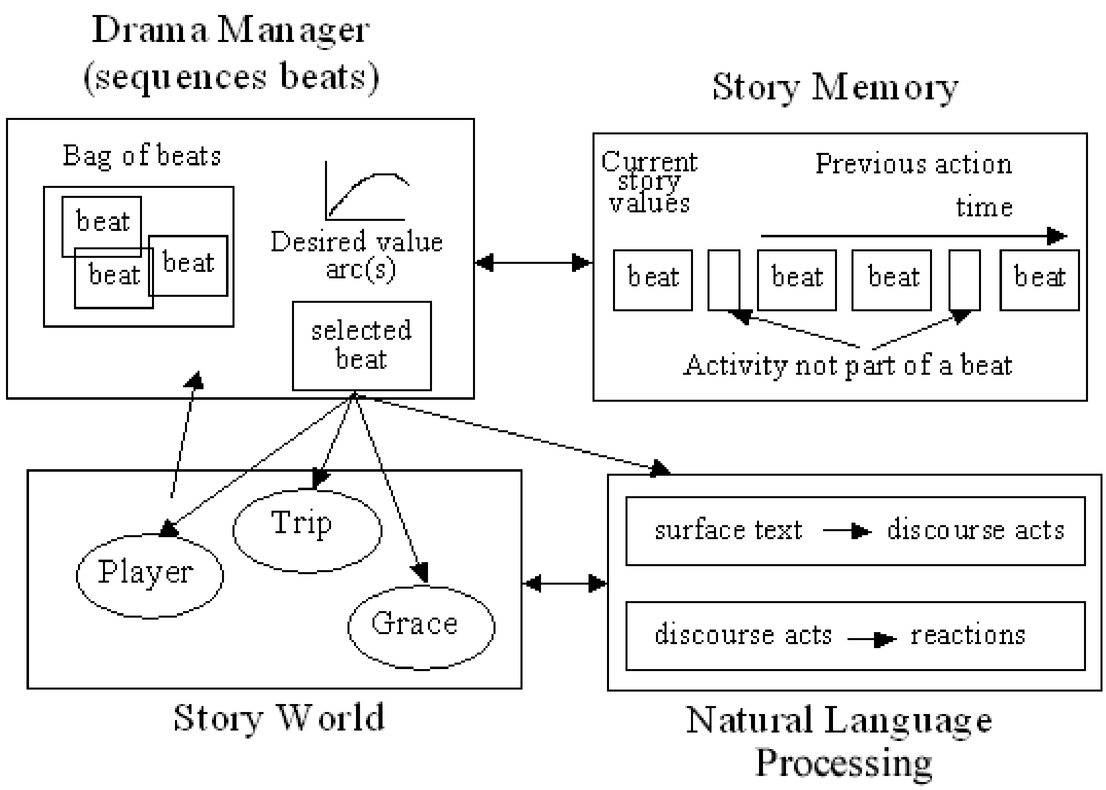

Figure 1. Façade interactive drama architecture

---

[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 reactive musical score, at the 1999 AAAI Fall Symposium on Narrative Intelligence.

Idioms (authoring techniques and behaviors built within the system) include:

- beat goals
- interaction handlers and beat goal mixins
- performance behaviors (dialog, staging, gesture and expression)
- long-term autonomous behaviors
- body resource management

The entire system was implemented from scratch on the Windows XP platform. The following authoring languages were developed:

- A Behavior Language (ABL), a reactive planning language based on Hap [Loyall and Bates 1991, Loyall 1997] that supports sequential and parallel behaviors, including joint behaviors. The compiler was implemented in Java and javacc, and compiles to Java.
- NLU Template language, a forward-chaining template rule language for specifying templates to map player-typed surface text into discourse acts. The compiler was implemented in Java and javacc, and compiles to Jess.
- Reaction Decider language, a forward-chaining rule language for proposing and selecting reactions to discourse acts. This runs on top of Jess with Java support routines.
- Beat Sequencing language, a language that specializes in drama management. The compiler was implemented in Java and javacc, and compiles to Java.

Façade has several threads that run in parallel:

- the non-photorealistic real-time rendered 3D story world, including the environment, objects and characters' bodies
- the believable agent Trip
- the believable agent Grace
- the Player's avatar agent
- natural language processing (NLP)
- the drama manager

The real-time rendered 3D story world is implemented in C++ with OpenGL. Character animation (body and face) is achieved through a mixture of procedural animation and layered keyframe animation data. We implemented a simple scripting language that can sequence together individual frames of keyframe body animation, and annotate pre-recorded lines of audio dialog with lip sync, timing and emphasis cues.

The behaviors for the believable agents Trip and Grace, which modulate the animation of their bodies and faces in the animated story world and sense information about the world, are implemented in ABL. The support code for the Player avatar is also implemented in ABL. The ABL agents communicate to the story world across a Java - C++ API via dll's. The NLP is implemented in a combination of the NLU template language and the Reaction Decider language. ABL agents can add and retract rule sets in the NLP, and invoked it as needed by the ABL agents; the NLP in return deposits messages for the ABL agents and drama manager in a shared memory. The drama manager is implemented in the beat sequencing language, and can add and retract behaviors in the ABL agents.

The following tour of the architecture is organized along the flow of activity through the system, intermixing descriptions of the architecture and idioms. The descriptions of each piece of the system are kept relatively brief; for more detail, including several code examples, refer to Michael's thesis [Mateas 2002].

3D story world and first-person user interface

The 3D story world consists of Grace and Trip's apartment, primarily a large furnished living room where the action of the drama is designed to take place. This room is about the size of a set for a theatrical stage play. The living room contains a couch, paintings and photographs on the walls including a large wedding picture, a bar with drinking glasses, an armoire with decorative objects, and a work table. The apartment also includes windows with a city view, a front door that leads out into a hallway where the Player initially enters from and can exit to, and a sparse (mostly unused) kitchen, bedroom and bathroom.

The interface is first-person in 3D space, in which the player can move forward and back with the up and down arrow keys, and rotate left and right with the left and right arrow keys. (The player cannot tilt their gaze upward or downward.) If the player moves to collide with an object or character, the player's forward motion is slowed to a stop and then is automatically routed to left or right to avoid the collision.

Figure 2. Real-time rendered characters Grace and Trip in Façade, with player-typed text and hand cursor.

The mouse controls a small hand-shaped cursor, an abstraction of the player's actual hand, not to scale. The shape and descriptive text on the cursor changes as the cursor changes context. By moving in space towards objects and clicking on them with the cursor, the player can pick up and put down objects, and knock on and open doors. Held objects become attached to the hand cursor, and in the case of drinks, can be sipped by clicking again when the drink is positioned at the lower center of the screen. The player can click Grace and Trip on their shoulders to comfort or hug them, or click on their lips to kiss them. Comforting, hugging or kissing causes the player to briefly move very close to gestured-to character, as if they are leaning in to perform the chosen gesture. (The gestured-to character will react in context as appropriate.)

To speak dialog, the player types text on the keyboard. The letters appear on the lower part of the screen, like subtitles. Until the player presses enter on their typed text, the dialog is not officially spoken.

The player is free to speak dialog, move around and gesture at any time. Discourse is continuous and real-time, not turn-based. If the player enters text while Grace or Trip is speaking, it tends to interrupt them at the moment the enter key is pressed. (Managing interruption is discussed later in the Player Interaction section of this paper.) However if the player is currently typing, Trip and Grace tend to pause in between lines of dialog of their performance for up to 10 seconds, behaving as if the player is currently speaking, giving the player an "opening" to interject dialog.

There are a few constraints imposed on the input of player dialog. The player is limited to typing one line of dialog at a time, up to a maximum of approximately 10 words at once. This limitation requires the player to speak in relatively simple sentences, increasing the chance that the NLP will be able to understand what was said, while still giving the player a wide range of natural language expression. Also, once the text is entered, the text stays displayed on screen for a few seconds, during which the player is prevented from entering new text. This gives the characters at least a few seconds to respond before the player can speak again.

To an observer watching someone play Façade, the style and pacing of discourse may seem a bit unnatural or undramatic, because Grace and Trip speak with audio voices and the player speaks with (silent) text, and because brief silences ensue if Grace and Trip are pausing while the player is typing. However our guess is that the player will not consider this noticeably unnatural, as she is presumably engrossed in the moment, trying to figure out what she wants to say and do, "hearing" her own words in her mind as if spoken out loud.

Believable agents

The believable agents Trip and Grace are each composed of a large collection of behaviors, written in A Behavior Language (ABL). The behaviors work together to modulate the animation of their bodies and faces in the animated story world, sense information about the world, and plan reactions to the player's actions.

The following is a brief overview of the features of the ABL language, many of which were in the original Hap. The nature of the language and its range of features may give some sense of its authorial power, which is specifically designed for implementing believable agents. Please refer to [Mateas 2002] for more details.

A Behavior Language (ABL)

A key requirement for making lifelike believable agents is to endow them with the ability do several intelligent activities in parallel – for example, to gaze, speak, walk, use objects, gesture with their hands and convey facial expressions, all at the same time.

In ABL, an activity is represented as a goal, and each goal is supplied with one or more behaviors to accomplish its task. An active goal chooses one of its behaviors to try. A behavior is a series of steps, which can occur either sequentially or in parallel. Typically, once a behavior completes all of its steps, it succeeds and goes away. However if any of its steps fail, then the behavior itself fails and the goal is forced to choose a different behavior to accomplish its task, if it has one. Furthermore, a behavior may have subgoaled its own set of goals and behaviors. To keep track of all the active goals and behaviors, and who is subgoaling who, ABL maintains an *active behavior tree* (ABT).

This paradigm of combining sequential and parallel behaviors, success and failure and the ABT are the foundation of the power of ABL as a language for authoring believable agents, versus purely sequential languages such as C++ or Java where parallelism has to be managed manually. ABL is effectively a multi-threaded programming language, making it very easy to author behavior mixing – a powerful feature that can get out of control quickly. In this way, ABL is challenging to program in, even for experienced coders. An important feature of ABL (the primary feature that distinguishes it from Hap) is support for synchronized joint behaviors, which helps the author harness the power of multi-threaded programming.

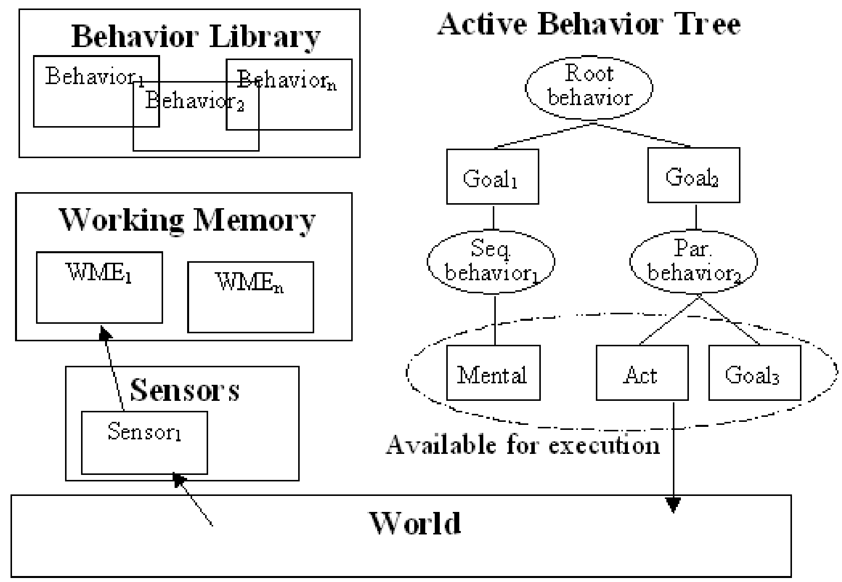

Figure 3. Architecture of a Hap / ABL agent.

Feature highlights of the ABL language:

- WMEs – working memory elements, which are data structures that can hold information (ie, variables), equipped for match tests in conditionals such as preconditions
- preconditions – an optional test attached to a behavior to determine if the behavior is allowed to become active when subgoaled
- context conditions – an optional test that will fail a behavior if the test becomes false
- success tests – an optional test that will succeed a behavior if the test becomes true
- acts – a step in a behavior that takes some sort of action, typically an animated body action such as taking a walk step, gazing at an object, speaking dialog
- mental acts – a step that does some arbitrary internal computation, in raw Java code
- subgoal and spawngoal – start a new behavior as a child of the current behavior, or as a child of a some other specified behavior
- joint with teammembers – when subgoaling, synchronously start the same behavior in multiple believable agents, if possible
- priority – determines which parallel behaviors run before others
- persistence – a way to retry a behavior if it succeeds or fails
- ignore failure – if a behavior fails, do not fail its parent
- effect only – do not require this behavior to succeed for its parent to succeed
- atomic – during this behavior, do not allow any other behaviors to run (shut off parallelism)
- specificity – specify the order in which to attempt different behaviors for the same goal
- number needed for success – how many children of a parallel behavior are required to succeed for the parent to succeed
- sensors – a way to have WMEs get automatically updated information about the 3D story world
- conflicts – a way to prevent certain parallel behaviors from running at the same time, if needed
- demons – a technique of creating a behavior that waits in parallel to all other behaviors for a condition to become true, and then takes action
- meta-abl – a way to reflect upon and affect the currently active behaviors themselves (kept track of in the ABT), to allow a behavior to directly alter, succeed, or fail other behaviors

API between ABL and the 3D story world

Grace and Trip's ABL behaviors sense the 3D story world (e.g., observe where the player is standing) and take action in the world (e.g., animate their bodies) by sending query and action messages to the world. Also they can receive spontaneous event notifications from the world (e.g., the player just said something). The following is a summarized description of the API between the ABL agents and the 3D story world.

- GetObjectPosition
- SetObjectPosition
- GetObjectRotation
- SetObjectRotation
- GetObjectPickupPosition
- GetObjectPickupRotation
- GetObjectStagingPosition
- GetObjectState
- GetGazeTracking
- SetGazeTracking
- SetPerformanceInfo
- DoWalkStep
- AbortWalkStep
- DoDialogAnimation
- Abort DialogAnimation
- DoGestureAnimation
- AbortGestureAnimation
- DoFullExpressionBaseAnimation
- DoFullExpressionMoodAnimation
- DoMiscLittleAction
- GetAllHeldObjects
- SetObjectToHold
- PlaySoundEffect
- PlayMusic
- SetMiscWorldInfo
- EventNotificationAnimationCue
- EventNotificationPlayerGestured
- EventNotificationPlayerTypedDialog
- EventNotificationObjectActivation

In summary, ABL behaviors send simple parameterized action requests to the 3D story world such as "take an angry walk step towards the couch", "look at this object in a coy way", "speak this line of dialog", "do an anxious but smiling facial expression", "do an emphasis hand gesture and nod when I speak", "make my eyes quiver", and so on. ABL behaviors sense the world with queries such as "what is the location of the wedding picture", or "what am I holding in my hand", and receive automatic event notifications such as "you just finished speaking a certain word in your dialog", or "the player just spoke these words", or "the player just picked up a martini glass". The 3D story world is responsible for accomplishing basic performance tasks such as low-level motor control of the body, procedural animation of facial expressions and gaze, lip-sync to dialog, and pathplanning.

Player agent

There is another ABL agent called the Player, which does not take any action on the story world (the human user fully controls the Player character), but instead senses the Player's actions, extracts longer-term meaning from them ("compound sensing"), and supplies this information to the other agents in the system, e.g., Grace, Trip, the drama manager. This includes determining when the Player is making significant movements around the room, and when she has been looking at an object for a significant amount of time.

The beat and dramatic performance

The collection of ABL behaviors Grace and Trip are operating with at a given moment, which effectively determine the scope of the simulation at that moment, are supplied by the current story beat. Only one story beat is active at a time. A beat's collection of behaviors is tailored to focus the activity of the characters in a particular narrative direction, while keeping them broadly reactive to other narrative directions. (Later sections of this paper will describe how and why the system has chosen the current beat to be active; this section describes the internals of an individual beat.)

For example, the "FixDrinks" beat has a collection behaviors whereby Trip and Grace verbally spar as they ask the player what she wants to drink, revealing hints about the conflict in their marriage. This collection of behaviors has a range of ways to perform this activity, and the flexibility to briefly diverge from this activity if the player tries to do something else, with an attempt to coax the player to return to the beat's intended activity. If the player persists on not choosing to participate in a beat's intended activity, the beat may abort, and the system moves on to a different beat, hopefully more aligned to the player's interactions.

Furthermore, beats are often authored to be able to perform their activity in different ways, depending on the story state. The most common variation in performance is dependent on the player's current *affinity* towards Grace or Trip – that is, does the player seem to be favoring Grace or Trip, or neither (neutral).

Beat goals

To achieve the level of flexibility just described, a beat's collection of behaviors are organized into a set of *beat goals*, which potentially can occur in different orders, some of which are optional. In the idioms we've developed for Façade, there are generally 5 types of beat goals in a beat:

- transition-in beat goals – the characters express their intentions for this beat
- body beat goals – the characters pose a dramatic question or situation to the player
- local beat mixin beat goals – the characters react to the player before the beat completes
- wait-with-timeout beat goal – the characters wait for the player's reaction to the situation
- transition-out beat goal – the characters have their final reaction to the player's action (or inaction) for this situation

(There is another type of beat goal that can occur during a beat, *global mixin beat goals*, that come from outside the beat, described later in this paper.)

For example, the FixDrinks beat has the following set of beat goals, all tailored towards a particular affinity, such as Player-Grace affinity. (There are two other sibling FixDrink beats, each their own set of beat goals, one tailored for Player-Trip affinity and the other for Player-Neutral affinity.) The names of each beat goal are prefaced with their type.

- TxnIn_BringUpTheIdeaOfDrinks
- TxnIn_AcknowledgeDrinksReferredTo
- TxnIn_AcknowledgeDrinkRequested
- Body_TripSuggestsAFancyDrink
- Body_TripEncouragesRequestedFancyDrink
- Body_TripBragsAboutHisFancyDrinks
- Body_GraceReactsToTripsBrag
- Body_WaitForResponseToTripsSuggestion
- Mixin_TripExcitedByPlayersAcceptance
- Mixin_TripExcitedByPlayersDifferentFancy
- Mixin_TripDismayedByPlayersDecline
- Mixin_TripUnsureAboutPlayersReluctance
- Mixin_GraceSuggestsCounteringAcceptance
- Mixin_GraceSuggestsEncouragingDecline
- Mixin_GraceSuggestsCoaxingReluctant
- Mixin_TripDiscouragesGracesSuggestion
- WaitWithTimeoutForAFinalResponse
- TxnOut_TripExcitedPlayerChoseHisFancyDrink
- TxnOut_TripExcitedPlayerChoseOtherFancyDrink
- TxnOut_GraceExcitedPlayerChoseNonFancyDrink
- TxnOut_PlayerChoosesNeitherCompromise

One of the believable agents in the system (arbitrarily Grace or Trip, designated as the "leader") contains a parallel behavior called BeatGoals at the root of its active behavior tree. BeatGoals is authored to initially subgoal all the transition-in and body beat goals – the beat goals that always need to be tried. Beat goals are marked to conflict with one another, so even though they are subgoaled from the parallel behavior BeatGoals, only one beat goal can be active at a time. Particular mixin and transition-out beat goals will later get added into BeatGoals, by way of spawngoal, in reaction to player action (described later in the Player Interaction section).

Each beat goal has preconditions to prevent it from being chosen if unnecessary or undesirable in the current context, e.g., if the player had not previously requested a specific drink, it makes no sense to choose the "TxnIn_AcknowledgeDrinkRequested" beat goal. Beat goals are optionally given priorities to enforce any needed partial ordering of the beat goals; any beat goals at the same priority will occur in a random order. Finally, beat goals are subgoaled as persistent if they fail, meaning that if a beat goal does not complete for some reason (e.g., it was interrupted), the beat goal will be subgoaled again the next chance it gets until it succeeds.

Inside a beat goal – dramatic performance behaviors

A beat goal has a particular focused dramatic task to accomplish, for example, "Body_TripBragsAboutHisFancyDrinks". A beat goal is typically one to four lines of dialog in length, about 5 to 15 seconds of on-screen performance time. Beat goals may have a variety of dialog alternates for performing its content. It behooves a beat goal to perform its content as quickly as possible, because if interrupted, the beat goal may later have to be repeated, and will need alternate repeat dialog to avoid robotically saying the same dialog over again.

A beat goal is typically implemented as a series of steps in a sequential behavior. Each step is a subgoal to a *joint parallel behavior* for Grace and Trip. It is here where the asynchronous autonomous agents Grace and Trip synchronize their behavior and coordinate their dramatic performances – for example, where one speaks and the other reacts with a look and gesture. A beat goal's series of joint parallel behaviors allows the author to direct Grace and Trip to banter several lines of dialog back and forth in quick succession.

Again, each joint step of a beat goal is a pair of behaviors – one for Grace, one for Trip. Each behavior subgoals one or more of the following *performance behaviors* in parallel:

- Staging (where to walk to, where to face)
- Dialog to speak (one or more pre-recorded phrases concatenated together into a sentence)
- Where and how to gaze
- Arm gestures to perform (e.g., raise and extend the arms to indicate enthusiasm)
- Facial expression to perform (a composite of parameters, e.g., angry smile, or happy skeptical)
- Head and face gestures to perform (e.g., eyes wander and head tilts down while thinking)
- Small arm and head emphasis motions, triggered by timing cues from the dialog (e.g., little head nods, hand flourishes)

Each of these types of performance behaviors are implemented as an extensive library of ever-present, reusable behaviors. These behaviors ultimately perform the low-level acts which send messages to the 3D story world across the ABL-storyworld API, that modulate the character's body and facial animation.

Performance behaviors are often spawngoaled off to the root of the ABT, which allows the beat goal step itself to finish up and move on to the next beat goal step before the performance behavior is actually done. For example, this allows a character to begin the process of walking somewhere (by spawning off a staging behavior), and then continue on with subsequent lines of dialog while the walking is taking place. This is a prime example of how easy it is in ABL to have many behaviors occurring independently, in parallel – essential qualities of lifelikeness. (If the author decided the beat goal step needs the walking to finish before it goes on to the next beat goal step, it would have been written to locally subgoal the staging behavior, instead of spawning it off as a parallel behavior).

Body resource management

Because there are many behaviors running in parallel, some intending to modulate the animation of the character's body, there needs to be a way to prevent behaviors from doing mutually exclusive body manipulations at the same time. For example, a staging behavior may want to direct the character's gaze in the direction the character is walking, while the gaze performance behavior may simultaneously want to direct the gaze to whom the character is currently speaking to, such as the player. This conflict is resolved by an idiom called body resource management, implemented in low-level ABL behaviors. Before sending a gaze animation command to the storyworld, a performance behavior will first request to own the gaze body resource, at a certain priority. If the resource is unowned, it is granted; if the resource is owned by another behavior at an equal or lower priority, it is granted to the new behavior and the previous owner is failed; and if the resource is owned by another behavior at a higher priority, the request is rejected. If rejected the requesting behavior could be coded to do something else, or fail. We established simple conventions of how different types of behaviors tend to request body resources at varying priorities. For example, the gaze performance behavior requests the gaze body resource at a higher priority than staging, because it is typically more dramatically important for a character to look at whom they are speaking to, than look in the direction they are walking.

Player interaction

This section describes how player interaction alters the performance of a beat (local agency), and how the interaction can have longer term effects on future beats (global agency). It starts off by describing how a player's action is interpreted, followed by how a reaction is chosen, finishing with how the reaction gets mixed into the performance of the beat.

Discourse acts

The system attempts to interpret player action into one or more *discourse acts*. A discourse act is a concise representation of the general meaning of the player's action. Any dialog typed by the player, any discrete gesture made by the player, and some patterns of player movement through the environment are interpreted as one of the following discourse acts, most with the option to be directed towards Grace or Trip:

- agree
- disagree
- positive exclamation
- negative exclamation
- express happy
- express laugh
- express sad
- express angry
- maybeUnsure
- dontUnderstand
- thank
- apologizeExcuseMe
- referTo <*topic*>
- praise
- ally
- criticize light
- criticize harsh
- oppose
- flirt
- pacify
- provoke
- greet
- goodbye
- getAttention
- intimate
- judgment
- suggestionAdvice
- misc-custom
- destructiveManipulation
- jokeTestLimits
- inappropriate
- hug
- comfort
- kiss
- physicallyFavor
- wanderAway

This list of discourse acts encapsulates the player's range of expression towards Grace and Trip. (Additionally, Grace and Trip's behaviors are always free to sense the "raw" uninterpreted actions of the player, such as the player's position and rotation, as needed.)

Façade is designed to be able to respond to any of these discourse acts at any time, in a manner appropriate to the current total *context.* A context is represented as a set of rules defining how to react to player action. Multiple contexts are typically active at one time, each at their own priority, creating a *total context*. Each individual context is implemented as a set of forward-chaining mapping rules, described in the NLP section below.

The current beat activates individual contexts that comprise the current total context. Typically this involves a single unique custom context and one more global contexts. Global contexts tend to be reused among beats, so an author usually only has to create one new unique context per beat.

The same discourse act in one total context (beat) may elicit a different response in another total context (beat). The sophistication of the response varies from discourse act to discourse act, from beat to beat. This variation in response to player's action from beat to beat, as well as the varying behavior collections from beat to beat, are the fundamental ways that the Façade simulation changes over time to achieve an interactive narrative.

Broad and Shallow Natural Language Processing (NLP)

Here we briefly describe how the player's typed text ("surface text") and discrete gestures are mapped into discourse acts, and how the discourse acts gets mapped into reactions. 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 processing 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 which human beings make use of in their everyday dealings with the world. While Façade is a micro-domain, a dramaticallyheightened representation of the specific situation of a couple's marriage falling apart, not the whole world, there are still no general theories, techniques or systems which can handle the syntactic, semantic and pragmatic breadth of the language use which occurs in Façade. Instead, Façade makes use of specific (non-general), a-theoretical, author-intensive techniques to understand natural language typed by the player.

NLP is divided into two phases: phase 1 maps surface text into discourse acts, while phase 2 maps discourse acts into one or more character responses. The player avatar agent is responsible for activating the NLP system whenever typed text or discrete gestures occur.

Phase I: Surface Text to Discourse Acts

The recognition of discourse acts from surface text is accomplished by rules written in a custom NLU Template Language, which compiles to Jess [Friedman-Hill 1995-2002], a java implementation of the CLIPS rule language [NASA 1985-2002]. This custom rule language looks just like Jess with the addition of an embedded template description language which allows compact descriptions of surface text patterns to appear on the left hand sides of rules. That is, the language provides a set of constructs for detecting patterns in surface text.

Template rules map "islands" of patterns in the surface text into intermediate meanings, which are then chained together to produce the final discourse act(s), capturing the pragmatic meaning of the surface text. 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." Additionally the template language provides support for matching stemmed forms of words, and for matching hierarchical (and recursive) template structures. For more detail on the syntax and operation of the NLU Template Language, and idioms for authoring template rules, refer to [Mateas 2002].

We are hoping to use the same general (large) set of Phase I templates for the entire story. Beats may add and retract small subsets of beat-specific template rules as needed.

The template rules for Façade err on the side of being overly permissive, mapping a large number of inputs, including ungrammatical ones, to discourse acts. (See the "Early Observations" section of this paper for further discussion of this issue.)

If the Phase I template rules cannot match the player's surface text into a discourse act, then it generates a special discourse act called "systemDoesNotUnderstand".

Phase II: Discourse Acts to Reactions

Now that the meaning of the player's surface text has been interpreted into a discourse act, Phase II of the NLP chooses a potential reaction for the current beat to mix in to its ongoing performance. Like the Phase I Template Language, the Reaction Decider language is also built on top of Jess.

Whereas the same general (large) set of Phase I template rules tend to get reused from beat to beat, in Phase II, small sets of custom rules are authored for each beat – a local, beat-specific context per beat, as mentioned earlier. Additionally, the beat typically activates reusable, shared, global rule sets, called global contexts. Each active context has rules to map some or all of the ~40 types of discourse acts to a proposed *reaction*, which if selected, the beat will mix in to its performance. (Local, beat-specific contexts change every beat, but global contexts only change a few times during the entire experience, as the overall tension of the story builds to a climax.)

Custom beat contexts tend to map some subset of the ~40 discourse acts into approximately 5 to 10 types of local, specific reaction proposals. Typical types of local reactions for the beat to mix in are:

- Player is mildly agreeing with the question / situation posed to her in this beat
- Player is strongly agreeing with the question / situation posed to her in this beat
- Player is mildly disagreeing with the question / situation posed to her in this beat
- Player is strongly disagreeing with the question / situation posed to her in this beat
- Player is not giving a direct answer to the question / situation posed to her in this beat

(These rules may consider how far along the beat has progressed in its decision making, as described later in the Mixins and Transition-Out section.)

Note that, for example, the "agree" discourse act is not the only discourse act that can map into a "player is agreeing" reaction. In some situations, the "praise" discourse act may be interpreted as agreement; whereas in other situations, "praise" may be considered "not a direct answer". These Reaction Decider mappings are hand-authored appropriately for the particular situation of the current beat.

If the discourse act falls outside the domain of what the beat's local custom context was listening for, the active global context(s) are there, always ready with a global, general reaction to propose, chosen from a pool of hundreds of reactions. Types of general global reactions include:

- Player has referred to an object in the room
- Player has done an affinity move (e.g., praise, criticize, flirt)
- Player has referred to a related topic (e.g., marriage, divorce, infidelity)
- Player has repetitively pushed on a topic
- Player has referred to the content of a previous beat

If none of the above global reactions were proposed, the system has no choice but to propose a *deflection* reaction, where the characters do their best to cover up the fact that the system did not understand the player's dialog. (More on this in the Story Content section of this paper.)

When a context maps a discourse act to a reaction type, it is *proposing* this as a possible reaction. Because there are usually multiple active contexts, multiple reactions may get proposed. After all the active contexts are done proposing reactions, a *selector* chooses one (possibly two) reactions for the beat to mix in to its performance. The selection is typically based on the priority of the context it was proposed from; for example, local, specific beat reactions tend to win out over global, general reactions.

For more details on the implementation of contexts, proposers and selectors, see [Mateas 2002].

Interaction handlers

Once the NLP system has chosen a reaction to a player action, the NLP's job is done, and it is now up to the current beat to mix in a performance of the reaction. This is accomplished by the beat's *interaction handler* behaviors, which use meta-ABL language features to abort the current beat goal and insert (using spawngoal) a new high priority beat goal, corresponding to the chosen reaction type, into the BeatGoals parallel behavior. (See the "Beat goals" section earlier in this paper for a description of the BeatGoals parallel behavior.) The newly inserted beat goal immediately begins executing, and later on the previous aborted beat goal will re-run if it hadn't already reached its beat goal gist point when it got aborted.

When a beat goal gets inserted into BeatGoals as a reaction to player action, it is considered one of two types: a mixin or a transition-out.

Mixins and Transition-Outs

Each beat keeps track of whether or not its *beat gist* has been reached. The beat gist is set to true when enough of the content of the beat has been performed that it is ready for the player's final interaction regarding the beat's situation. For example, in the "FixDrinks" beat, Grace and Trip are ready to finish up this beat once they've done a sufficient amount of quibbling over what they think the player should drink.

The setting of the beat gist point is author-coded custom logic – typically when all of the body beat goals have each reached their beat goal gist points (and therefore there are no more required beat goals left to do).

Before the beat gist has been reached, the NLP-decided reaction to player action will always be a *mixin* beat goal – a reaction authored to respond in some way to the player's action, but not resolving the situation of the beat. For example, if the player refers to divorce during the "FixDrink" beat, Grace and Trip mix in a short beat goal about their feelings about divorce – and then resume where they left off with their FixDrink beat goals. (More on this in the Managing Interruption section below.) Recall that mixins can either be local, specific beat goals or global, general beat goals.

After the beat gist has been reached, the NLP-decided reaction may now be a *transition-out* beat goal, where the player's action has been interpreted to resolve the outcome of the beat. Only one transition-out beat goal is ever chosen per beat, and it becomes the final beat goal of the beat. For example, if the player has agreed to Grace's drink suggestion after the beat gist was true, then the "TxnOut_GraceExcitedPlayerChoseNonFancyDrink" beat goal is chosen. (Extending the idiom a bit, if a mixin reaction is chosen after the beat gist, the interaction handler can be coded to additionally cause a transition-out – a second reaction following the mixin. In such a case, the beat is reacting sequentially in two ways to the same discourse act, which is sometimes useful.)

Altering story state

Mixins and transition-outs, besides performing a local reactions to player action, can be annotated with side effects on global story state. For example, a mixin that reacts to "praise Grace" may also shift the Player's affinity towards Grace. Or, a reaction to "referTo infidelity" may increase story tension. Altering story state will affect future beat selection, and may even cause the current beat to abort (see the Drama Management section).

Managing interruption

Earlier it was described how interaction handlers, when they receive a chosen reaction to player action from the NLP system, abort the current beat goal and mix in a new beat goal. The previously aborted beat goal, if it hadn't already reached its beat gist point, is to be re-run later after the mixin is done.

This happens slightly differently if the current beat goal has set its "letBeatGoalFinish" flag to true. The behavior author may decide that a certain beat goal is important or intense enough that a chosen reaction to a player action should *not* be immediately responded to, and instead be delayed until the beat goal gist point. For example, if Trip and Grace are yelling at each other, they can believably ignore the player until their yelling is over. (By convention, we try to keep this uninterruptible period as short as possible, no more than 5 seconds long at a stretch.)

So beat goals that turn on their "letBeatGoalFinish" flag are considered *uninterruptible*, until they reach their beat goal gist. Interaction handlers will delay interrupting the current beat goal with a reaction mixin / transition-out until the "letBeatGoalFinish" is false.

If a reaction causes a beat goal to get aborted before it had reached its beat goal gist, it will be re-run later. In such a case, it would be unnatural for the beat goal to simply repeat its original dialog. Therefore any interruptible beat goals need *repeat dialog alternates*. When running its repeat dialog, a beat goal always sets itself to be uninterruptible, so the author does not have to write yet more alternate dialog if the repeat dialog is interrupted. Repeat dialog tends to be a short and efficient version of the original dialog.

Additionally, all beat goals contain a bit of *reestablish* dialog. This dialog gets played at the start of the beat goal if a local or global mixin had just occurred. The idea here is that a mixin may have just taken us off-topic from the intention of the current beat; reestablish dialog serves to refocus the player on where the beat last left off.

Interaction during a mixin

Finally, consider the case where the player acts *during* a mixin. That is, the player had performed an action, the system had chosen a reaction, and before the system was done performing the reaction, the player acts yet again. This is handled by authoring all mixins to have early beat goal gist points, and be uninterruptible until that point. If the player causes a reaction to be chosen during the mixin, the reaction gets delayed until the current mixin is past its beat goal gist point; then the current mixin is aborted and the new mixin beat goal begins. The previous mixin does not need to be repeated, because it had reached its gist point.

Drama management

So far in this tour of the architecture we have focused on how the player experiences local agency in Façade. Local agency means that the player is able to see clear, immediate reactions to her interaction; the characters immediately react to player actions and utterances. The smarts to handle local interaction, and thus create a sense of local agency, reside within beats, as described in previous sections of this paper. The logic of beat goals plus interaction handlers is the framework for local interaction.

The smarts to incorporate interaction into a larger scale story structure, to incorporate not just the most recent 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.

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. While the player is experiencing Façade, 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. After the story is over and the player reflects back upon the experience, global agency means that she could look at the sequence of beats and create an satisfying, believable explanation for why that sequence happened.

Structuring the story as a bag of beats

The nature of Façade's drama – the story of a married couple inviting a friend over for drinks, ostensibly discussing how great their lives are while dodging and weaving around the fact their marriage is falling apart, as the tension builds to a breaking point a la *Who's Afraid of Virginia Woolf* – was chosen because it can be successfully broken apart into story beats that can be coherently resequenced in many different orders. Façade is purposely designed as a somewhat open-ended psychological situation with an array of topics to be discussed, secrets to be revealed and head games to be played, in which it is acceptable for only a subset of topics, secrets and head games to occur in any one run-through of the drama. Contrast this to a tightly-plotted drama such as *Casablanca*, where the order of events is carefully crafted and really cannot be altered without ruining the integrity of the narrative. It is safe to say that certain types of stories, such as character-oriented kitchen sink dramas, lend themselves better to interactivity than plot-oriented action dramas.

Beat Sequencing Language

In the Beat Sequencing Language developed for Facade, the author can annotate each beat with selection knowledge, can define actions which are performed at various stages in the beat selection process, and can define beat variables which are accessible by all tests and actions within a beat.

Selection knowledge

There are six types of selection knowledge which influence the selection of a beat:

- precondition {<wme test>} – similar to an ABL behavior precondition – an optional test on story memory variables (WMEs) if this beat is allowed to run

- 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.
- weigh_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, if the weight test is true the associated weight overrides the static weight. If multiple weight tests are true, the test with the largest associated weighting factor is used.
- 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 then the beat is given the specified priority. If a beat defines both a static priority and a priority test, if the priority test is true, the associated priority 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. 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. 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.

Actions

Beats can define actions which are performed at various stages in the beat selection process:

- 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.
- select_action is executed if a beat is selected for sequencing. Select actions are typically used to activate the beat behaviors (beat goals, interaction handlers and NLP contexts) and set any WME state which the beat behaviors will need.
- succeed_action is executed if a beat succeeds.
- 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, similar to mental acts in ABL behaviors.

Beat-scope variables

Beats can define beat variables which are accessible by all tests and actions within a beat. The beat environment is persistent – beat variables maintain their values across beat selection cycles.

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 directly 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.

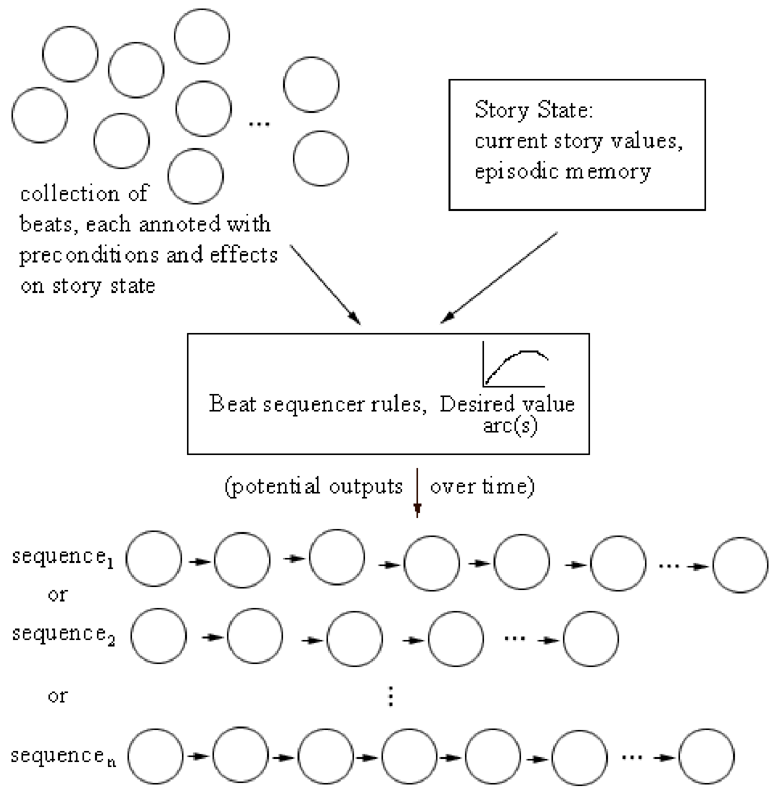

Figure 4. Illustration enumerating the potential outputs of beat sequencing.

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 which may play a role in beat selection.

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 which is the max of the priorities associated with the satisfied priority tests. Collect the beats in **Satisfied** which 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 which 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.

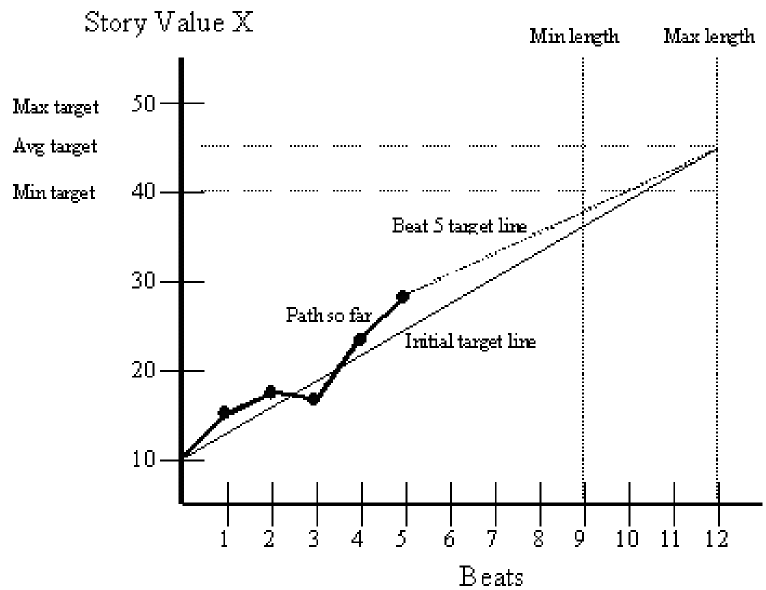

Figure 5. Example beat scoring situation with a linear story arc.

Figure 5 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.

The beat scoring algorithm makes use of the definitions in Table 1.

At the beginning of the plot unit, 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 $deltaX_{beat}$ is closest to $slope_{initial}$, 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 after the first beat has been selected, for example, the scoring for selecting beat 6 (beat 5 has just finished playing out, so the situation looks like Figure 5). 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.

Definition	*Expression*
Initial value of X	$\mathbf{X_{initial}}$
Average target value of X	$\mathbf{X_{avg}}$
Value of X after beat n	$\mathbf{X_n}$
Optimal value of X given by initial target line	$\mathbf{X_{nopt}}$
Maximum number of beats	$\mathbf{beat_{max}}$
Minimum number of beats	$\mathbf{beat_{min}}$
Slope of the initial target line	$\mathbf{slope_{initial} = (X_{avg} - X_{initial}) / beat_{max}}$
Slope of the adjusted target line for choosing the n+1 beat	$\mathbf{slope_{n+1} = (X_{avg} - X_n)/ (beat_{max} - n)}$
The delta value change for story value X caused by a candidate beat	$\mathbf{deltaX_{beat}}$
Candidate beat score	$\mathbf{score_{beat} = 1 / e^{\vert slope_{n+1} - deltaX_{beat} \vert}}$

Table 1. Values employed by the beat scoring algorithm.

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 step? 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. However, 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 $slope_{initial}$ 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[2] 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 which allows it to achieve the specified targets given the player's interaction. Authors are essentially giving the

[2] The beat manager keeps track of $error_n = sqrt(\sum_{i=1 \text{ to } n} (A_i - A_{iopt})^2)$ (the standard square root of sum of squares error), though any error term could be used.

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.

Automatically maintained drama management state

The beat manager automatically maintains some state which is useful in preconditions, priority tests and weight tests:

- 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.

Beat behavior parameters

When a beat is selected and it instantiates is collection of behaviors in Grace and Trip, it may pass some parameters to the beat goals:

- transition-in type – specific information about which transition-in beat goal makes sense to perform
- affinity switch – letting the beat behaviors know if we had just aborted a beat in order to switch to a different affinity-version of the same type of beat
- spin – if the transition-out of the beat goals should try to spin the content positively or negatively, if applicable

Long-term autonomous behaviors

There is a special breed of performance behaviors in the library of ever-present reusable behaviors, called *long-term autonomous behaviors.* These behaviors that are initially spawned by beat goals at the root of the ABT, and execute beyond the life of the beat that spawned them, for up to several minutes, across several beats. They may lie dormant for a while, and then attempt to grab body resources in order to perform a short bit of activity. For example, this may be as simple as a character sipping their drink from time to time, or as complex as making drinks and carrying them to the player. If a long-term behavior needs to speak some dialog, it spawns its own beat goal mixins. Long-term behaviors are designed to mix well with any beat. This level of arbitrary behavior mixing greatly enhances the illusion of life of Grace and Trip.

Story content organization

The following is a brief overview of the particular way we have organized Façade story content into story values, topics, beats, mixins, and so on. A different story implemented in the same architecture would have its own particular organization of content, perhaps very different than this organization.

Spoiler warning: some high level details about the story content of Façade are discussed here; please skip this section to avoid learning details you would rather not know until you get a chance to play the finished experience itself!

Story values

Façade keeps track of two primary story values, which have strong influences on beat sequencing:

- tension – an integer value ranging from 0 to 5
- player affinity – an integer value ranging from -2 to 2, where -2 means the player has strong affinity for Grace, 2 means the player has strong affinity for Trip, and 0 means the player is neutral. (Affinity is a zero-sum game in Façade.)

Value arc

The following diagram illustrates the ideal pacing of the rise and fall of tension in Façade, resembling an Aristotelian dramatic arc:

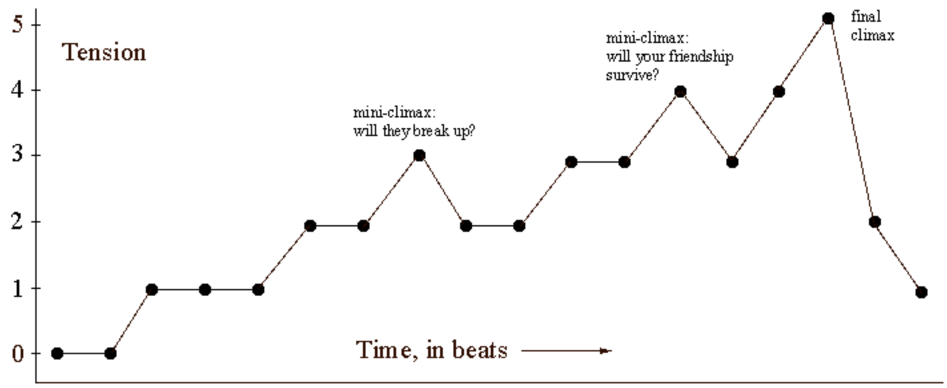

Figure 6. Ideal tension value arc for Façade.

Story topics

There a four primary story topics in Façade. Any one beat tends to focus the conflict and reveal information on one of these topics:

- ArtistAdvertising (AA) – concerned with how Grace originally wanted to be an artist, but instead Trip pressured her to pursue a more lucrative career in advertising.
- Façade (F) – concerned with how Trip and Grace portray a veneer of success and happiness to their friends and each other, when actually they are each deeply unsatisfied with their lives.
- RockyMarriage (RM) – concerned with how their marriage began on a faulty premise and developed over time into an unsustainable, dysfunctional relationship.
- TripsAffairs (TA) – concerned with Trip's infidelities, his reaction to the unhappiness in their marriage.

Bag of beats

As of this writing, the have specified only a subset of the total number of beats the system will contain in its finished form (~200). The bag of beats so far includes:

- PBehindDoor
- TGreetsP
- TFetchesG
- GGreetsP
- ExplainDatingAnniv
- Decorating AA tension1 affinity-Neutral
- Decorating AA tension1 affinity-Grace
- Decorating AA tension1 affinity-Trip
- WorkAsOutlet AA tension1 affinity-Neutral
- WorkAsOutlet AA tension1 affinity-Grace
- WorkAsOutlet AA tension1 affinity-Trip
- FixDrinks F tension1 affinity-Neutral / Trip
- FixDrinks F tension1 affinity-Grace
- GracesParents F tension1 affinity-Neutral
- GracesParents F tension1 affinity-Trip
- GracesParents F tension1 affinity-Grace
- Italy RM tension1 affinity-Neutral
- Italy RM tension1 affinity-Grace
- Italy RM tension1 affinity-Trip
- TripsParents RM tension1 affinity-Neutral
- TripsParents RM tension1 affinity-Grace
- TripsParents RM tension1 affinity-Trip
- WorkTable TA tension1 affinity-Neutral
- WorkTable TA tension1 affinity-Grace

- WorkTable TA tension1 affinity-Trip
- AloneG
- ForgotCheese
- TReturns
- AloneT
- Veronica
- GReturns
- Decorating AA tension2 affinity-Grace
- Decorating AA tension2 affinity-Trip
- WorkAsOutlet AA tension2 affinity-Grace
- WorkAsOutlet AA tension2 affinity-Trip
- FixDrinks F tension2 affinity-Trip
- FixDrinks F tension2 affinity-Grace
- GracesParents F tension2 affinity-Trip
- GracesParents F tension2 affinity-Grace
- Italy RM tension2 affinity-Grace
- Italy RM tension2 affinity-Trip
- TripsParents RM tension2 affinity-Grace
- TripsParents RM tension2 affinity-Trip
- WorkTable TA tension2 affinity-Grace
- WorkTable TA tension2 affinity-Trip
- RomanticProposal
- MoveWeddingPic
- InappropriateRecovery
- MiniClimax

Global mixins

Beyond what beat goals offer, the player can experience significant amounts of the story content from global mixin beat goals, which are short reactions mixed into beat performance (described earlier in this paper). The number of global mixin beat goals are too numerous to fully list here (~500 in the entire story). These are the categories of global mixins, each of which contains several variations at each tension level of the story:

- object mixins – contentful responses to references to the various objects in the world
- affinity DA mixins – affinity-game style reactions to praise, criticism, flirt, etc.
- satellite topic mixins – contentful responses to references to related topics such as marriage, divorce, infidelity, sex, therapy, money, career, family, children and happiness.
- pushed too far mixins – dramatic responses to when the player has repeatedly dwelled on a story or satellite topic, to a breaking point.
- post-beat mixins – customized deflect-style responses when the player refers to the specific content of a previous beat.
- pattern mixins – short reactions that comment on noticeable patterns in the player's interactions.
- deflect mixins – an array of general responses for deflecting or ignoring the player when the NLP system did not understand her typed dialog.

Long-term autonomous behaviors

Examples includes:

- fixing drinks
- sipping drinks
- personality moves (e.g., Trip's eightball)
- miscellaneous commentary

Early Observations

How well does this architecture work? That is, to what extent does it offer the player the experience of an interactive story? Although the authoring of Façade is still underway, this section includes a few early observations of what we already believe we understand about the successes and failures of the system.

The success of an interactive story involves the combination of many factors, some of which of course are subjective and can only be measured empirically, if at all. However, we can try to quantify a few characteristics of Façade that we believe have a significant influence on the quality of the experience as a whole.

Player agency

We believe it is valuable to try to measure the degree of agency the player has in an interactive story, both local agency and global agency. Three ways to measure local agency include:

- on average, how many distinctly different expressions / actions can the player input to the system at any moment;
- on average, how many distinctly different immediate responses can the system express back to the player, in reaction to her action;
- on average, how often does the context of the narrative change, such that repeating the same action gives a different immediate response.

For all three of these, we believe Façade performs well relative to the state-of-the-art for interactive narratives (e.g., critically-acclaimed examples of hypertext fiction, interactive fiction, adventure games and immersive simulation games.) In Façade, the player is always free to express any of the ~40 types of discourse acts, mapped from millions of combinations of raw natural language surface text, some of which have a range of parameters, e.g., topics that can be referred to. The NLP system maps the player's input to one of anywhere from 3 to 10 different context-specific local responses (local beat goals), and/or one of least 30 different generic global responses (the currently accessible global mixin beat goals from a pool of ~500). The context of the narrative changes frequently in Façade, with small updates on every player interaction, and larger updates once or twice a beat, about every 30 to 60 seconds. Furthermore, the system never repeats a reaction (except for generic deflection reactions) – as the player interacts, she works her way through a broad database of content, and can only see up to about 25% of the total content in any one session.

One way to measure global agency is to count the number of possible orderings of meaningful story events, assuming the player has strong influence on which ordering occurs. (How meaningfully different the orderings are from one another is a more subjective, empirical question, strongly influenced by the particular meaning of each event, which this analysis does not address.) Façade has a two-tiered hierarchy of chunks of meaningful content, as previously described: beat goals (sentence-level chunks) and beats (plot-event-level chunks). Here we are concerned with the number of possible orderings of beat goals within any beat, and the number of possible orderings of beats over the entire story. A typical beat chooses from 10 local and 30+ global beat goals, and is about 6 beat goals in duration. Not all permutations are possible of course, because of the partial ordering imposed by beat goal preconditions and the particular selection logic of the NLP rules. We can safely say that a typical beat has hundreds, if not thousands, of coherent possible orderings of its beat goals (phrases and sentences). Moving up the hierarchy from beat goals to beats, the beat sequencer will have ~200 beats at its disposal, with a full story about 18 beats in duration. Again, preconditions and effects specify a partial ordering; we estimate there will be thousands of distinctly different possible beat orderings (plot-events) in Façade. Just as with local content, the player can only see a minority of the total number of beats in any one session, hopefully motivating her to replay the story to see more, inevitably with a different ordering.

Division of responsibility between the author, the system and the player

The Façade architecture offers the author a framework for organizing the content of a dramatic story into a hierarchy of pieces: beats, each containing beat goals, each containing behaviors that the system dynamically organizes into an active behavior tree. Each piece in the hierarchy is annotated with preconditions on global story state and/or local simulation state, and each piece can potentially cause effects on such state. Pieces can also be annotated with priorities, context-conditions, success-tests, and in the case of beats, scoring weights. It is the author's job to define the conditions, tests, priorities, scoring weights, content meaning, and effects of each piece. When the system runs and the player contributes her own continuous effects on the simulation and story state through her actions, the resulting sequencing of the content – the construction of the narrative – occurs.

In Façade, it is to the extent of the evaluation of this author-annotated knowledge about behaviors and beat goals and beats that the system is "reasoning" over the story content and "generating" a story. The system's role is that of an editor, assembling a story from an array of story fragments, where each fragment was created by a human author, who presumably possessed some original intent on how the fragments could ultimately fit together. In this way, the degree of player agency – how "interactive" the story is – is proportional to the number of possible coherent orderings, which is proportional to the number of pieces of story content. Therefore the potential "goodness" of the story is still very much in the author's hands, encoded in the quality of the content meaning of each piece on its own, and the narrative quality of the potential ordering of the pieces, encoded in the preconditions and effects. (However, once the grain size gets very small and the number of pieces gets very large, the line between editing (sequencing) content and writing (generating) content becomes blurry.)

We should not forget the contribution of the player's expressions themselves to the quality of the story as a whole. By virtue of the open-ended natural language input in the interface, the player can type any dialog they want at any time, and it gets always performed (displayed) on-screen. In this way, the player is doing more than influencing the plot; the player is also interacting to produce dialog to be included in the text of the story overall. In a stageplay trace of an experience of playing Façade, the player's language would be equally intermixed with the system's language. Furthermore, the system always tries to maintain the player's suspension of disbelief of the experience as a "real" dramatic play; the characters will never utter "illegal command", "system does not understand", or "does not compute", even when the system does not actually understand. Assuming that the player isn't purposely being absurd, a trace of the experience should be coherent, containing at worst the occasional non-sequitur.

Failures of the system

Natural language understanding. Given the kind of story we want to tell, an adult domestic drama, the use of language in the story is unavoidable. But just as the open-ended natural language input offers the player a true-to-form way of participating in a drama, this interface will inevitably be the least effective and most aesthetically failure-prone part of the system. We expect three types of failures in Façade regarding natural language input: non-understood utterances, false positives, and an asymmetrical range of expression.

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 which occurs in Façade. Instead, Façade makes use of specific (non-general), a-theoretical, author-intensive techniques to understand natural language typed by the player. But even with thousands of hand-coded template rules, and the limiting of the number of words in any one utterance, players will be able to (accidentally or not) input text that the system cannot understand. When this occurs, the best the system can do is *fail gracefully*. Non-understood utterances trigger behaviors that employ an array of deflection-type responses, such as "hold on, hold on" or "well, what I was trying to say was...", or by abruptly changing the subject, as if they have something urgent they need to say and can believably ignore the utterance. As mentioned earlier, the system never outright rejects an utterance with an error message, which would unnecessarily break the player's sense of immersion in the drama and suspension of disbelief that Grace and Trip are intelligent adults.

The template rules for Façade err on the side of being overly permissive, mapping a large number of inputs, including ungrammatical ones, to discourse acts. 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. While players will sometimes try to break the 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. This permissiveness will inevitably result in false positives, where the system thinks the player meant something that she did not. This could potentially be very confusing to the player, forcing her to try to understand why the characters reacted the way they did.

Finally, although Façade achieves some symmetry in the interface beyond what it is typically offered in interactive narratives – that both the characters and the player speak in natural language – the player may notice an asymmetry between the range of expressions she can perform and the range of responses the system can perform in return. With open-ended natural language input, at any moment the player can express millions of different meanings valid within the domain of the story. However the system has, at best, only hundreds of different meanings it can express back at any moment. If noticeable, this incongruity will further impair the player's suspension of disbelief.

Authorial labor. While the architecture affords the author an expressive framework for creating a character-oriented interactive drama, in practice it is time consuming to create all of the behavior to achieve what we consider a satisfying level of lifelikeness, responsiveness and player agency. A future research direction to pursue is one where the system can help generate beats, beat goals and behaviors themselves, helping relieve the authorial burden of creating enough content that achieves a satisfying level of agency for the player. If the system is to do more sophisticated reasoning and therefore generation, this would probably require, for example, annotations on story content more descriptive than preconditions and effects.

Authorial complexity. As described earlier, the languages developed for this architecture, particularly ABL with its direct support for parallel behaviors and behavior mixing, offer the author the expressive power to achieve lifelike behavior in fewer lines of code than traditional sequential languages such as C++ and Java. However this power can be difficult to harness, and one can quickly write programs that are unusually difficult to debug. The learning curve for taking full advantage of the authorial power of an architecture like Façade may be steeper than we would like.

Sufficient degree of global agency. While the Façade architecture can support a non-trivial degree of story agency for the player, achieving this requires creating a critical mass of story content for the system to work with. As previously described, Façade's idioms for authoring real-time interactive lifelike dramatic performance put much of the burden of content creation (e.g., behavior coding) on the human author, versus having the system itself share more of the load by being more generative. Even with our plan to spend upwards of two man-years on authoring on Façade, it is unclear whether we will achieve the critical mass of story content that allows for the plot to feel as mutable as we originally hoped for.

Player as protagonist. The player in Façade is more than an interactive observer, but is not the story's protagonist. The story of Façade is primarily about the dissolution of Grace and Trip's marriage; the player plays a central role in the outcome of this dissolution, but is not the "main character". At most the player shares an equal role with Grace and Trip (which as authors we are happy with, and we hope players will be satisfied with as well). Originally we had intended to author additional story content that would have put the player more front-and-center in the drama, but due to time constraints have had to cut that aspect of the potential story.

Scalability. This architecture is designed to support *intimate* interactive experiences – that is, it supports relatively rich and varied expression for a single player among a small number of complex characters in a small environment, versus relatively shallow expression between multiple players and large number of simple characters in a large environment. It is not clear how well this architecture would scale up to the epic scope of a typical contemporary game, if desired.

Conclusion

This mid-project paper described Façade's project design goals, discussed the motivations underlying the design of the architecture, provided a tour of the architecture, and finished with some early observations about what we already believe we understand about the successes and failures of the system. We look forward to completing the authoring process in 2003 and releasing Façade to the public to play.

The proof is in the pudding. Future papers will detail the lessons learned from the authoring process, and attempt to evaluate how well Façade achieves a middle ground between open-ended simulation and structured narrative.

References and Bibliography

Aarseth, E. 1997. Cybertext: Perspectives on Ergodic Literature. Baltimore: The Johns Hopkins University Press.

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.

Bates, J. 1992. Virtual Reality, Art, and Entertainment. Presence: The Journal of Teleoperators and Virtual Environments 1(1): 133-138.

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.

Berne, E. 1968. Games People Play: The Psychology of Human Relationships. New York, NY: Random House.

Bernstein, M. 1998. Patterns of Hypertext. In Shipman, F., Mylonas, E. and Groenback, K. (eds.), Proceedings of Hypertext '98. New York: ACM.

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.

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.

Colby, K., Weber, S. and Hilf, F. 1971. Artificial Paranoia (Parry). Artificial Intelligence, 2:1, 1-25.

Crawford, C. 2002. Assumptions underlying the Erasmatron storytelling system. Forthcoming in M. Mateas and P. Sengers (Eds.), Narrative Intelligence. Amsterdam: John Benjamins.

Fellbaum, C. (Ed.). 1998. Wordnet: An Electronic Lexical Database. MIT Press.

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.

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.

Landow, G. 1992. Hypertext: The convergence of contemporary critical theory and technology. Baltimore, MD: John Hopkins University 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.

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.

Loyall, A. B. 1997. Believable Agents. Ph.D. Dissertation, Tech report CMU-CS-97-123, Carnegie Mellon University.

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.

Mateas, M. 2003 (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: MIT Press.

Mateas, M. 2002 (forthcoming). Interactive Drama, Art and Artificial Intelligence. Ph.D. Dissertation, Carnegie Mellon University.

Mateas, M. 2000. Expressive AI. In Electronic Art and Animation Catalog, Art and Culture Papers, SigGraph 2000. New Orleans, LA.

Mateas, M. 1999. 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 and Sengers, P (Eds.). 2002 (Forthcoming). Narrative Intelligence. Amsterdam: John Benjamins.

Mateas, M and Stern, A. 2002a. 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. 2002b. A Behavior Language for Story-Based Believable Agents. In Working notes of the Artificial Intelligence and Interactive Entertainment Symposium. AAAI Spring Symposium Series. Menlo Park, CA.: AAAI Press. An alternate version appears in IEEE Intelligent Systems, July 2002.

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.

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.

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.

Pinhanez, C. 1997. Interval Scripts: a Design Paradigm for Story-Based Interactive Systems. In Proceedings of CHI97. Atlanta, GA, pp. 287-294.

Reilly, S. 1996. Believable Social and Emotional Agents. Ph.D. Dissertation., School of Computer Science, Carnegie Mellon University.

Ryan, M. 2001. Narrative as Virtual Reality. Baltimore, MD: Johns Hopkins University Press.

Sengers, P. 1998a. Anti-Boxology: Agent Design in Cultural Context. Ph.D. Dissertation. School of Computer Science, Carnegie Mellon University.

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: MIT Press.

Stern, A. 2003 (forthcoming). Creating Emotional Relationships with Virtual Characters. In R. Trappl, P. Petta and S. Payr (Eds.), Emotions in Humans and in Artifacts. Cambridge MA: MIT Press.

Stern, A. 2001. Deeper conversations with interactive art, or why artists must program. Convergence, 7:1, 17-24.

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. Forthcoming in M. Mateas and P. Sengers (Eds.), Narrative Intelligence. Amsterdam: John Benjamins, 2002.

Stern, A. 1998. Interactive Fiction: The Story is Just Beginning. IEEE Intelligent Systems, 13:5, 16-18.

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.

Tambe, M. 1997. Towards Flexible Teamwork. Journal of Artificial Intelligence Research (7) 83-124.

Thomas, F. and Johnston, O. 1981. The Illusion of Life: Disney Animation. Hyperion.

Weizenbaum, J. 1966. Eliza. Communications of the ACM, 9, 36-45.

Weyhrauch, P. 1997. Guiding Interactive Drama. Ph.D. Dissertation, Tech report CMU-CS-97-109, Carnegie Mellon University.