Expanding the Interaction Lexicon for 3D Graphics

Jeffrey S. Pierce

CMU-CS-01-160
November 2001

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:

Randy Pausch, Chair
Jim Morris
Scott Hudson
Alan Kay, Viewpoints Research, Inc.

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

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) via the US Air Force Rome Laboratory under Grant No. F30602-97-2-0251, the US Navy under Grant No. F33615-93-1-1330, the US Army under Contract No. DAAD17-99-C-0061, and by a Microsoft Graduate Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, Microsoft Corporation, the US Government, or any other entity.

Expanding the Interaction Lexicon for 3D Graphics

Keywords: human-computer interaction, interaction techniques, 3D interaction, generative method, breaking assumptions, content creation, object manipulation, navigation, 3D painting, interaction surface, interaction maps, voodoo dolls, visible landmarks, place representations

Abstract

Historically user interfaces evolved in a series of rapid paradigm shifts followed by periods of incremental development. Because widely adopted user interfaces are resistant to change, we can potentially have a greater impact on improving interfaces by working on the next, rather than the current, interaction paradigm. While there are several candidate paradigms, I chose to focus on 3D interaction because of its potential to leverage people's natural and learned abilities. While current 3D interfaces are promising, we have only explored a small part of the design space. I believe that the reason is that we are limited by our assumptions when we build 3D interfaces. We all grow up in a fully immersive 3D world, and when we design virtual worlds we often transfer characteristics of the real world without considering other options. We have also been building worlds long enough to start thinking of the current methods as the correct methods, and thus rather than explore new parts of the design space we emulate existing work. I believe that to realize the potential of 3D interaction we need to expand the interaction lexicon for interactive 3D graphics. My hypothesis is that we can create demonstrably useful 3D interaction techniques by identifying and breaking our assumptions about the real world and about existing practice. I present an existence proof for this hypothesis consisting of interaction techniques that I created using this approach. These techniques allow users to manipulate objects using Voodoo Dolls, navigate large virtual worlds using Places and Landmarks, and specify the interaction semantics of 3D models by painting Interaction Surfaces. I also present experimental evaluations of these techniques that demonstrate their utility.

Table of Contents

Evermore thanks, the exchequer of the poor.
William Shakespeare

Acknowledgements

To Kate for helping me survive the process and reminding me that the light at the end of the tunnel was not an oncoming train.

To my parents for teaching me to value education, self-motivation, and self-reliance.

To Randy Pausch for his insight and guidance during my PhD student career, and for his willingness to let me occasionally visit some of the stranger neighborhoods in the design space.

To Matt Conway for his help developing image plane interaction techniques, and for always being willing to discuss my latest interaction technique idea. I lost count of how many times I burst into Matt's office at MSR exclaiming "I've got an crazy idea...". Matt's instincts as an designer continue to amaze me.

To Andy Forsberg for his help developing the image plane interaction techniques that laid the groundwork for the Voodoo Dolls technique.

To Dennis Cosgrove for helping with the implementation of Interaction Surfaces. The fifteen minutes he spent explaining barycentric coordinates to me are emblematic of Dennis' willingness to take time out of his already busy day to help someone with a question.

To Brian Stearns for helping to build the Voodoo Dolls technique. Brian is one of the best sounding boards for ideas that I have worked with, always willing to toss an idea around for awhile and see where it led.

To Kevin Christiansen, a man who could build anything and quite often did.

To Tommy Burnette, who thought that the image plane techniques were cool enough to build support for image plane functions directly into the Alice API.

To Denny Proffitt, for his invaluable critiques of the formal experimental designs in this thesis.

To Steve Audia, Chris Sturgill, Shawn Lawson, David Stern-Gottfried, Sarah Hatton, and Moshe Mahler for creating and painting the models I needed. Without their help the virtual worlds in this thesis would be dull, uninteresting places.

To the members of the Stage 3 Research Group, past and present, for their tireless work developing Alice. A better tool for prototyping interaction techniques would be hard to find.

To Doug Bowman, Ivan Poupyrev, Mark Billinghurst, Mark Mine, and the other members of the 3D User Interface mailing list for their willingness to engage in spirited discussions about 3D interaction.

To George Robertson and the fine folks at Microsoft Research who not only let me spend time working with them, but also supported my work with the Microsoft Fellowship.

To the many researchers, known and unknown, who took the time to read drafts of chapters and conference papers and provide invaluable feedback.

...it is the magical part that is all important and that must be most strongly attended to in the user interface
Alan Kay

CHAPTER 1 *Introduction*

One of the driving forces behind the study of human-computer interaction is the fact that even infinite computing power is useless unless users can effectively communicate their intentions to the computer and understand the results of the computer's computation. This combination of input and output between the user and computer is generally refer to as the computer's *user interface*. Historically these interfaces have evolved in a series of rapid paradigm shifts followed by periods of more incremental development. During the 1940s through 1970s these user interfaces went from batch processing to command line interfaces to WIMP (windows, icons, menus, and pointing) interfaces.

While WIMP interfaces were a significant improvement over command line interfaces, this paradigm has earned the nickname the "WIMP plateau": the power of our computers is increasing exponentially, but the way that we interact with our computers has not altered significantly since the 1970s. This should not be taken as an indication that WIMP interfaces are ideal. Indeed, these interfaces possess some notable drawbacks. They limit users to two input degrees of freedom, and allow applications to present data along only two dimensions at a time. While WIMP interfaces grew out of a growing interest in designing to the user's capabilities, they often require mapping an ability developed in a 3D world to a 2D world. They also suffer from screen real estate limitations: users often have more windows open than they have space available for and so require coping strategies for moving between tasks.

Despite these drawbacks, no new paradigm has yet mounted a serious challenge to WIMP interfaces. This is not due to a lack of candidates: speech, gesture, and interactive 3D (also referred to as virtual reality) are common examples, and the next paradigm will probably be a blend of these technologies. Each of these technologies, however, still presents challenges that must be overcome. Speech recognition still needs to be able to reliably recognize speaker-independent, connected speech. Gestural input needs to reliably recognize handwriting or (in the case of more abstract gestures) determine the most natural set of gestures for people to use. In this thesis I will focus on one of the key challenges for interactive 3D graphics: creating the interaction lexicon.

1.1 What is a 3D[1] Application?

In this thesis I define a 3D application as a program that displays a 3D world to the user and allows the user to interact with that world in real time. I am intentionally using a broad definition because research on 3D applications is not limited to a single combination of display and input devices. Currently the most common display alternatives are desktop 3D, fishtank Virtual Reality (VR), CAVEs (CAVE Automatic Virtual Environments), and head-mounted displays (HMDs). Desktop 3D uses a conventional desktop monitor to display the 3D world, while fishtank VR [Ware93] adds head tracking and a stereo image. CAVEs [Cruz-Neira] project images on large screens that surround the user. HMDs [Sutherland] place displays directly in front of the user's eyes. Researchers typically pair the display device with a six degree of freedom tracking system that provides information about the position and orientation of the user's head (and often hands), although a 3D application may also use input devices with fewer degrees of input. Many 3D games, for example, rely only on a keyboard and mouse for input. Because we are still exploring the advantages and disadvantages of different hardware configurations I do not wish to exclude potentially beneficial 3D applications from consideration simply because they do not use the "correct" hardware. Spreadsheets, for example, were a key breakthrough for early computer interfaces independent of whether the user interacted with them using a mouse or keyboard.

1. When I refer to 3D in this proposal I am specifically referring to *interactive* 3D.

1.2 Why Focus on 3D

3D interfaces have two big advantages over 2D interfaces. First, people spend all their lives in a 3D world and they develop skills for manipulating 3D objects, navigating around the 3D world, and interpreting 3D spatial relationships. While 2D interfaces allow people to use *some* of these skills, they constrain people to use only a proper subset of their skills. High degree of freedom spatial input tasks in particular (e.g. manipulating the atoms in a 3D molecule) do not transfer well to 2D interfaces.

Even those skills that people can use they can often only apply in limited ways. For example, people can use proprioception and kinesthesis to gain a sense of how they are positioning and using their bodies. WIMP interfaces only really allow people to use proprioception to remember where they moved the pointing device, while a 3D interface might allow a person to remember that they hung the hammer from their toolbelt or dropped it behind them [Mine].

Spatial memory is another example. People are very good at indexing memory using locations (demonstrated by the Method of Loci mnemonic technique [Yates], ascribed to Simonides around 500 BC and also referred to as the Memory Palace technique [Spence]). However, 2D interfaces provide a very limited sense of space, limiting the opportunity to apply spatial memory. 3D interfaces, by providing a third dimension and a wider variety of distinctive spaces, can provide more opportunities to use spatial memory (e.g. [Card][Robertson]).

The other advantage of 3D interfaces is that they have a larger design space than 2D interfaces: we can design a wider variety of 3D interfaces than we can 2D interfaces. However, I believe that the view that 3D interfaces subsume 2D interfaces simply because a 2D interface can be applied to any convenient 2D plane in the 3D interface [Lindeman] is too simplistic. 3D interfaces give use new kinds of control: 3D worlds are governed by more complex laws than 2D worlds, and in a virtual world we can define those laws as we see fit. While this larger design space has a correspondingly larger potential for valuable new interfaces, it also means that we may have a harder time finding those interesting points in the design space. This leads us to the challenges for interactive 3D.

1.3 Challenges for Interactive 3D

Despite the potential benefits of 3D applications, there are still two key challenges that we must address to make 3D a viable successor to the WIMP paradigm. The first is the cost of the hardware required to run the 3D application. For the last 30 years interactive 3D graphics was largely confined to research labs, in part because the processing requirements for a system capable of rendering complex scenes at perceptually acceptable rates placed the system out of the price range of the average consumer. This is no longer the case: 3D graphics accelerator cards provide this performance for prices under $200, and inexpensive console game machines like the Playstation 2, X-Box, and Gamecube provide even more rendering power. The price of VR display devices and tracking systems may still be prohibitive, but both of these expenses are steadily decreasing.

The second key challenge is to provide a set of techniques for users to interact with the 3D world. Any interactive 3D application will involve some combination of (at a minimum) selection, manipulation, wayfinding, and navigation, and developers will need techniques that allow users to perform these tasks. If we think about interactive 3D graphics as a new medium that we are exploring, then the set of interaction techniques that we need to create forms part of our *lexicon* for the medium and defines the concepts we can express and the tasks we can perform.

1.3.1 The Lexicon for a New Medium

A new medium typically passes through three stages. In stage one the medium is so new that people regard any use of the medium as unusual and interesting. Thus, to borrow film as an example, people would pay money to watch a movie of waves crashing on a shore. Stage one for 3D graphics was rendering simple wireframe objects and moving around a 3D space.

In stage two people use the new medium to emulate existing media. Thus the next use of the movie camera was to film stage plays. Because I am focusing on interactive 3D graphics, the stage two use of 3D graphics that I am interested in is the use of 3D to emulate the real world. Many applications of 3D graphics are still in this stage.

Stage three is where people take advantage of the unique characteristics of the new medium and create a new lexicon for the medium. The tricks and techniques that take advantage of these characteristics comprise the lexicon of the new medium. The lexicon for film includes techniques like close up, crosscut, flashback, fade, and zoom. Citizen Kane [Welles] is an excellent example of stage three. Orson Welles uses the lexicon to tell a story via film that would be difficult to tell in a stage play. The lexicon of a new medium both shapes and defines that medium: how interesting would a movie be if it could not use any of these techniques? Would you pay to see it?

We have taken a few initial steps toward creating the lexicon for 3D graphics. Navigation techniques like teleportation and object manipulation techniques like Worlds in Miniature (WIM) [Stoakley] provide new capabilities not available in the real world. We can display the world in letterbox format (as if the user was watching a movie) to communicate that the user cannot interact with the world at that moment; 3D games have already adopted this convention. We can also use an animated third person to first person transition to communicate the user is playing the part of a specific character in the 3D world. Unfortunately we are still making very slow progress: the research community has been creating and working in 3D worlds for years, but we have only begun to define the interaction lexicon for 3D graphics. I believe that part of the problem is that we are not exploring all of the design space, only those areas that cluster around a 1:1 mapping to the real world. We have essentially cornered ourselves in the design space: we are limited not by the speed of our hardware but by a lack of interesting new ideas.

1.3.2 Cornered in the Design Space

I believe that the most significant reason that we have not explored more of the design space is that when we design a virtual world we are bound by our assumptions. We all grow up in a fully immersive 3D world with fixed rules (notwithstanding the occasional revolution in physics), and when we design a virtual world we tend to transfer characteristics of the real world (e.g. occlusion) without considering other options. Ironically some of these assumptions actually require extra work in a virtual world. For instance, many designers work extremely hard to add realistic gravity to their worlds, yet the need for gravity in a virtual world is not at all clear. When users let go of a tool in a virtual world, it might be much more convenient if the tool remained floating in place instead of falling to the floor out of reach.

Another way of thinking about this is to realize that there is a story associated with every virtual world; we tell this story to the user to explain the rules of the world. That story may be “you’re going to fly a carpet in Agrabah” [Pausch96], but more commonly the story is “it’s just like the real world”. Certainly the latter story is the easiest to convey to the user, but there are other, more interesting stories that we can tell.

We are also bound by the assumptions of established practice. We have been building worlds long enough to start to think of the current methods as the correct methods, and thus rather than explore new parts of the design space we simply emulate existing work. This effect is most visible in desktop systems where the WIMP paradigm dominates designs so completely that other approaches [Perlin] are rarely considered. The tendency of our existing ideas and systems to perpetuate themselves is constraining our thinking.

Some researchers focus on virtual worlds that closely resemble the real world in the hopes of creating a sense of “presence” in the virtual world. Some initial research results do suggest that users feel more “immersed” in a virtual world if the interaction techniques they use are similar to how they would interact in the real world [Usoh]. However, this assumes that the designer’s goal is to immerse the user; if the designer’s goal is to make the user as productive as possible, then more unusual interaction techniques might be appropriate. Emulating the real world is often a useful starting point, but not necessarily the destination. One of the lessons from the Aladdin VR project [Pausch96] was that consistency was more important than realism. Alan Kay wrote [Kay] that “it is the *magical* part that is all important and that must be most strongly attended to in the user interface design.” Just as we do not force users of a word processor to start a new document every time they make a mistake, we should not force users of a virtual world to walk everywhere they go.

The set of assumptions we are fighting is slightly daunting. However, people faced this same problem with previous media and still managed to (eventually) use those media in new and interesting ways. I believe that the key to expanding the design space for 3D interaction techniques is to force ourselves to think “outside of the box,” to work with non-traditional players, and use our creativity to take advantage of the unique characteristics of 3D as a medium.

1.4 Encouraging Creativity: Identifying and Breaking Assumptions

Researchers create most new interaction techniques by focusing on a particular task or human capability. I propose a new method: identifying our assumptions and then imagining what is possible if we break them. Identifying and breaking assumptions is an established technique for encouraging creativity. Most of the interesting existing 3D interaction techniques break specific assumptions about the real world, whether or not this was their creator’s intention. Thinking about the assumptions that a particular technique breaks is an interesting way of classifying the technique. Furthermore, thinking about these assumptions may in turn suggest new techniques or new assumptions that we can break. By focusing on identifying and breaking our assumptions I believe that we can design new interaction techniques in previously unexplored sections of the design space.

1.5 Thesis Statement

Interactive 3D applications have a lot of potential. Before application developers stop developing WIMP applications and start developing interactive 3D applications, however, we need to develop the interaction lexicon for interactive 3D graphics. In short, we need to more comprehensively explore the design space for 3D interaction techniques. To do this I believe that we need to blend creativity for generation and hard science for evaluation. My hypothesis is that:

- We can create new interaction techniques by breaking our assumptions about the real world and about existing practice.
- These new interaction techniques will be demonstrably useful: they will provide new capabilities or make the user more efficient.

In this dissertation I present an existence proof for this hypothesis. This proof consists of interaction techniques that I created by identifying and breaking assumptions. I started by classifying existing interaction techniques by the assumptions they break. I then generated new techniques that break both these initial and new assumptions. I did not fully implement all of the new techniques that I generated; instead, I focused on the three most promising interaction techniques and formally evaluated them to assess their value. These three techniques, and the evaluations that demonstrate their value, form the core of my proof.

1.6 Contributions

This research makes several contributions to 3D interaction and virtual reality:

- The Voodoo Dolls technique, a new technique for manipulating objects in immersive 3D environments where users manipulate handheld copies of objects.
- A formal evaluation comparing Voodoo Dolls and Indirect HOMER (Hand-centered Object Manipulation Extending Ray-casting) [Bowman97], a best practice manipulation technique, that demonstrates that Voodoo Dolls users can position and orient objects more accurately than Indirect HOMER users. On average, across all the experiment tasks, Voodoo Dolls users positioned objects 88.28% more accurately and oriented objects 72.86% more accurately. The results of this study suggest that exploring better methods of providing feedback for interaction techniques may be a valuable avenue for future work.

- A new technique for navigating large virtual worlds using place representations and visible landmarks.
- A practical implementation of a previously hypothetical panning and zooming 3D WIM that incorporated residue and semantic zooming.
- A formal evaluation comparing navigation with Places and Landmarks with navigation using a panning and zooming WIM that demonstrates that users can travel between locations faster when navigating with Places and Landmarks. On average Places and Landmarks users completed the within-place tasks 21.67% faster (29.813 seconds vs. 38.063 seconds) and the between-place tasks 37.77% faster (36.403 seconds vs. 58.495 seconds).
- A new process for specifying a 3D model's interaction semantics where designers paint interaction surfaces onto the model and the system saves the painted surfaces in an interaction map.
- A qualitative evaluation of this process that demonstrates that designers, regardless of artistic ability, can easily specify the interaction semantics of 3D models and assign them interactive behaviors using this process.
- Eighteen ideas that provide enhancements to existing techniques or starting points for further exploration, including creating copies to aid selection, multiple light-weight views of an object, system maintained repositories, and symbolic linking.

Taken together, these contributions form the existence proof for the primary contribution of this dissertation: a new method for creating interaction techniques by identifying and breaking our assumptions. The techniques this dissertation contributes and the evaluations of them demonstrate that this approach can yield valuable interaction techniques. In addition, this dissertation contributes a list of assumptions that I and other researchers have broken, intentionally or not, to generate new techniques. This list may be a fruitful starting point for researchers attempting to create a new technique using this approach.

1.7 Organization

I have organized the rest of this thesis as follows:

In chapter two, Breaking Assumptions, I categorize existing interaction techniques by the assumptions they break, and present a list of assumptions we can productively break.

In chapter three, Voodoo Dolls, I describe the Voodoo Dolls interaction technique, and present a formal study comparing the Voodoo Dolls technique with the HOMER technique. The study demonstrates that the Voodoo Dolls technique allows users to place objects more accurately, perhaps because of the additional feedback the technique provides.

In chapter four, Interaction Surfaces, I describe how designers can specify interaction surfaces by painting interaction maps. This process is a simpler, more flexible method of specifying what parts of a 3D object can respond to user actions. I present a formative evaluation [Hix] of this process that demonstrates that designers can easily learn and use it.

In chapter five, Places and Landmarks, I present a technique for efficiently navigating large virtual worlds using Visible Landmarks and Place Representations. I also describe an alternative technique, navigating using a panning and zooming WIM, based on existing ideas in the literature, and I present a formal study that compares these two techniques for rapid travel in a large virtual world. This study demonstrates the superiority of navigation with places and landmarks for rapid travel.

In chapter six, Other Promising Ideas, I present other ideas for interaction techniques that I generated in the course of this thesis. While these ideas were not promising enough to formally evaluate, they might potentially augment existing interaction techniques or provide a starting point in the future development of new interaction techniques.

In chapter seven, Analysis and Conclusions, I reflect on the lessons learned during this process and summarize my contributions.

Finally, in the appendices I include the data from the formal studies of the Voodoo Dolls and Places and Landmarks techniques.

The advances of the new medium will be defined by seminal visions
of those who are expressing themselves in ways heretofore impossible.
Timothy Oren

You can't make an omelette without breaking a few eggs.
The Joker

CHAPTER 2 *Breaking Assumptions*

Many of the current "best practice" 3D interaction techniques already break one or more of our assumptions about the real world. However, the researchers generating these techniques generally did not explicitly consider breaking their assumptions. My goal in this chapter is to present a set of assumptions that researchers might productively consider breaking. To generate this list I reviewed the existing literature for interaction techniques that take advantage of the novel properties of VR. For each of these techniques I then determined the assumption(s) that it breaks. In this chapter I present these assumptions, along with the existing techniques that break them.

The resulting categorization is not technically a taxonomy. A good taxonomy is not only comprehensive, but the categories are also *exclusive*, and some of the assumptions I list are related or overlap slightly. This is intentional: I am more interested in generating new interaction techniques than classifying existing ones, and by allowing assumptions to overlap I provide different approaches to thinking about new techniques. Bowman's taxonomy [Bowman99a] is an example of a traditional taxonomy; he classifies interaction techniques based on how a user accomplishes the different components of his task.

2.1 Categorizing Existing Techniques

One of our most basic assumptions is that space is linear and continuous. If point B is 10 miles from point A, I will need to traverse 10 miles to reach one point from the other. Similarly, to walk ten miles I will take essentially the same number of steps independent of the direction I am walking. In a virtual world, however, we have full control over space and can create techniques that break this assumption. Teleportation is one example: users can move instantaneously between locations regardless of the distance between them. The destination can be chosen in a variety of ways, including pointing to the desired location on a 2D map [Angus], walking through a portal [Barrus], or returning to a previous *spacemark* (essentially a bookmark in space) [Liang].

Mine created a *head butt zoom* technique [Mine] that allows users to create spatial non-linearities on the fly. Users frame a distant object with their hands on their image plane to create a temporary window frame in space. Sticking their head through this window causes their viewpoint to zoom through space to examine that object up close; pulling their head back returns them to their previous position.

Non-linearities or discontinuities in space can affect the link between users' physical bodies and their virtual bodies. For example, a user's physical hand may be in a different location than his virtual hand. Poupyrev explored non-linear motion with the *Go-Go* interaction technique [Poupyrev95]: when the user reaches out his virtual hand moves at a different rate (e.g. exponentially) than his physical hand. This allows the user to reach out and grab objects that would be out of reach in the physical world. On the other hand, Bowman's *HOMER* technique [Bowman97] creates a discontinuity. When the user selects an object by pointing at it his virtual hand teleports to grasp that object, allowing the user to manipulate that object remotely as if holding it in his hand.

We can also create worlds with built-in non-linear spaces. For example, we can create a world with one-way passages [Barrus]. If the user walks along such a passage from location A to location B, on reaching location B he may discover that the passage has disappeared, leaving no way to return to location A. We can also use non-linear spaces to create buildings that are bigger on the inside than the outside [Barrus]. This allows designers to create large, interesting spaces inside buildings without making those buildings excessively large in the outside space; this can help reduce clutter and crowding in the outside world.

These techniques are breaking another, broader assumption, namely that the visual properties of a space match the physical properties of the space. A space that looks continuous in a virtual world may actually be discontinuous. Existing interaction techniques break this assumption in other ways. An object that is "physically" present in the virtual world may be invisible. Many virtual worlds only draw the user's hands, neglecting to draw the rest of his body. Although many worlds avoid drawing the user's body because they cannot accurately track it, not drawing the user's body has the advantage that the user can still take advantage of the physical presence of his body (by attaching virtual objects to it, for example) without his body obstructing his view. However, not drawing the user's hands is probably a bad idea, because drawing the user's hands helps guide his actions.

Virtual worlds may also contain physical constraints without a visual counterpart. A toolbox might follow the user around without being visibly attached to the user [Butterworth]. Breaking this assumption can help guide the user's navigation through the virtual world. A user flying through a 3D version of the solar system might only be able to move along paths that keep the Earth and Moon in view [Gleicher], or the user might be attached to an invisible guide by an invisible spring to keep her moving through the world [Galyean]. A designer can place force fields around objects to help the user avoid bumping into them [Xiao].

Of course, in the real world the visual properties of a space do not always match its physical properties. Consider optical illusions: they occur when appearance does not necessarily reflect reality. However, in a virtual world an optical illusion does not have to be an illusion: we can change reality to reflect appearance if we choose. One of the first systems to break this assumption was the NASA Ames VIEW system [Fisher89]. Users could create a window that looked into a different location; when they resized that window to fill their view the system actually teleported them to that location. Pausch used a similar idea for navigation using a handheld miniature [Pausch95]. To move to a different location the miniature grows and moves so that the user occupies the desired position in the expanding miniature. When the miniature reaches 1:1 scale the system teleports the user to that position in the virtual world.

Pierce breaks this assumption in a different way with image plane techniques that allow users to retrieve distant objects [Pierce97]. When the user frames a distance object between his fingers he creates an optical illusion. The framed object can be interpreted as large and distance or small and near, and the system can alter reality for the user's convenience.

Another implicit assumption is that there is only one world. Obviously we can break this assumption with virtual worlds, because we create new worlds all the time. But we can also give users within one virtual world direct access to other worlds. These worlds might be embedded directly into the current world [Feiner], or they might be completely disconnected from the current world. The user may be able to manipulate objects that represent these different worlds to change their layout or move between them [Stoakley]. If these other worlds or spaces are not normally visible but can be quickly accessed at need the user can store frequently used items in them for easy access while not cluttering his immediate work area [Pierce99b]. We can even create multiple worlds and overlap them using transparency to convey the notion of alternate lines of possibility, and then turn a particular world either opaque (when possibility becomes certainty) or fade it away (when it becomes less likely or impossible) [Conner97][Sung].

Instead of multiple worlds that are completely different, we can think about multiple worlds that are actually the same world at different points in time. For multi-user worlds, we can break the assumption that everyone in the world occupies the same point in time. We can offset users slightly in time to help reduce the likelihood of users performing conflicting actions due to network latency [Sharkey].

We also assume that the world is persistent even when not in view: your home does not vanish into thin air while you are at the office and then reappear when you turn into your driveway. There are different ways of breaking this assumption. To increase speed 3D graphics libraries do not render objects that are out of the user's view, but objects that are not currently in view usually still continue to exist as a mathematical description. Mine's *over the shoulder deletion* technique [Mine] breaks this assumption. Although Mine did not describe the technique from this viewpoint, in essence the world does not exist behind the user's back, so any object the user throws over his shoulder is considered destroyed and no longer part of that world. Cheney explores another variant where objects out of the user's view are neither drawn nor animated; this simulation culling [Cheney] is another method of increasing the frame rate for interactive worlds.

Designers do not always break techniques because they think it is a good idea. Sometimes assumptions are broken simply because it is easier to break them than not to. Most virtual worlds break the assumption that the world is spherical because it is easier to build the world on a plane than on a sphere. Other times the designer would dearly love *not* to have to break a particular assumption. The assumption that gravity exists is a good example. Designers often do not implement gravity (and other real world forces like friction) simply because of the computational complexity. However, designers should consider whether or not virtual worlds really need gravity. Some virtual worlds already take advantage of the lack of gravity by allowing the user to fly or hover. The lack of gravity can also be extremely convenient: users can release objects and have them remain within reach (floating in place) rather than falling to the floor. I suspect that if most virtual worlds were suddenly to start implementing gravity we would find them extremely inconvenient to work in.

In addition to avoiding real world attractive forces like gravity, we can break the assumption that objects do not arbitrarily attract each other by allowing users to create their own attractors in the world. Users can create special attraction objects in the scene and use them to help snap objects together [Bier], or they designate objects already in the scene as attractors that affect the object the user is current moving around [Venolia].

In addition to these assumptions about worlds and spaces, we also assume that the intrinsic (e.g. density, opacity) and extrinsic (e.g. size, shape) properties of objects are inviolate and distinct. Just like gravity, designers often break this assumption without desiring to. Many virtual worlds do not provide collision detection (they do not enforce the density of objects relative to each other) because of the computational expense. This allows objects to occupy the same space, which can be a useful ability. Consider Zhai uses an intangible cube as a *silk cursor* [Zhai] to make it easier for users to select objects in desktop 3D worlds. Objects in virtual worlds are usually weightless as well (unless the system is providing haptic feedback). This makes it much easier to manipulate objects, as a weightless virtual sofa is much easier to arrange than a physical sofa.

Existing interaction techniques also dynamically change the opacity of objects. For example, surgeons can look inside the human body without actually cutting into it [Bajura]. Viega implemented *3D magic lenses* [Viega] that can allow the user to see inside any normally opaque object. These magic lenses can actually be used to selectively modify the properties of objects or the world itself within their volume: time can move faster or slower, gravity can be reversed, or objects can appear inside out.

Turning to extrinsic properties, most virtual worlds keep the size of objects (and the user) constant. Some interactive worlds change the shape and detail of objects as the user gets farther away to increase the world's frame rate [Hoppe]. Other systems (e.g. [Butterworth]) allow the user to change his own size. This allows the user to shrink and work with extremely small objects, or grow and quickly traverse large distances. Alternately, the user can shrink or grow the entire world. Mine shrinks the entire world when the user selects an object [Mine] to bring the object within reach and allow the user to change its position. The user can also change the size of individual objects. As mentioned previously, image plane techniques change the size of objects to make reality reflect appearance [Pierce97].

An object's properties can also depend directly on the viewer. Rademacher implemented view dependent geometry [Rademacher] that changes the shape of an object as the views it from different positions. Objects can also look different to different viewers in multi-user virtual worlds; beauty can indeed be in the eye of the beholder. These subjective views [GSmith] allow an individual's view of the world to be tailored specifically for that user's tasks or interests.

A property of objects that is usually even more taken for granted is that objects are persistent: we cannot create or destroy them on demand. 3D modeling programs (e.g. [Zeleznik][Liang][Butterworth]) break this assertion all the time, but we can break this assumption for other reasons as well. Creating and destroy objects on the fly allows the user to work with a large number of different widgets without cluttering up his environment. With snap-dragging [Bier] users dynamically create jacks that facilitate the manipulation of objects. With Pierce's *Voodoo Dolls* technique

[Pierce99a] the user dynamically creates and destroys miniature copies of objects called dolls for the same reason. The system can also create objects to represent tentative actions, and destroy those objects if the actions are cancelled. Conner creates a transparent copy of a locked object for the user to manipulate while the lock is being obtained [Conner97]; this gives the user a low latency experience, and the system can destroy the copy or the original depending on whether or not the user obtains the lock. Sung extends Conner's approach by showing all the other participants in the multi-user world the transparent copy so that they understand what the user is trying to do [Sung]. Sung also uses transparent copies to help users maintain context. If one user is manipulating a cup on a table and a second user moves the table, Sung creates a transparent copy of the table for the first user to work relative to while he works with the cup. When he releases the cup the transparent table copy fades away, and the cup fades into place on the actual table (where a transparent copy of the cup represented its eventual destination).

Conner and Sung's techniques breaks another assumption, namely that objects can only exist in one location. The object the user is working with temporarily exists in two locations while the system is trying to obtain the lock. The objects represented in a *world in miniature* [Stoakley] also exist in two places at once: in the larger world and in the miniature. Manipulating an object in one location manipulates the object in the other location, and the user can choose whichever is more convenient. To some extent we can think about the dolls in the Voodoo Dolls technique in the same way, although to be precise while the dolls and the larger objects they represent are identical in appearance they do have different behaviors.

From the properties of objects we now turn to the effects of objects and the assumption that "passive" objects cannot affect other objects. For example, in the real world a person can look for the "you are here" spot on a map to determine his location, but moving the "you are here" spot will not change his location. In a virtual world, however, any object can be a widget [Conner92] that combines an appearance with some useful (or even useless) behavior. This makes possible all varieties of useful combinations. Changing the "you are here" spot on a map can indeed change your location [Angus][Pausch95]. The user can sketch a path through the world, and that path can then draw the user along it [Igarashi98]. A user holding a miniature 3D model of a room (a *world in miniature*) can rearrange the furniture in the room by rearranging the furniture in the model [Stoakley]. Users can even practice a little voodoo, manipulating objects at a distance using miniature copies of those object [Pierce99a].

Even for those objects that do affect other objects, people generally assume that objects work the same in all contexts. Scissors, for example, will cut paper equally well whether you use them in your left or right hand. In a virtual world, however, we can change the behavior of objects so that they work differently in different situations. We can leverage the fact that we use our dominant and non-dominant hands differently [Guiard] by making objects work differently depending on the hand holding them. A miniature held copy of an object held in the non-dominant hand might make the object a frame of reference, while the same miniature held in the dominant hand might change the position and orientation of the object it represents [Pierce99a]. We can also change the behavior of a control button based on the location the user presses it, so that when the user presses it above his head he pulls down a menu, while when he presses it while looking at an object he selects that object [Mine]. Another variant is an object that only works at certain orientations. Forsberg presents an aperture selector that only works when aligned with the target object [Forsberg].

Regardless of how objects actually work, in the real world we assume that objects have no built-in knowledge of their properties or functionality. By breaking this assumption, we can create objects that actively assist the user in working with them. For example, we can create maps or bird's eye views that reorient themselves for the user as moves around [Darken93][Fukatsu]. If we encode objects with their default resting plane [Houde], we can create objects that know to move on top of surfaces, along walls, or on the ceiling to make it easier to manipulate objects in a 3D space using a 2D input device [Bukowski]. We can extend the idea of encoding a default resting plane by building offer and binding sites into objects [Gösele]. Objects can then use this information when users manipulate them to determine whether they should move over or under other objects. Instead of acting on their own, objects can also provides hints to users about how interact with them. Kallman explored the idea of encoding descriptions of an object's functionality directly into the object, so that the object can show the user how to activate that functionality [Kallman].

We can also store the changes that happen in a virtual world or to a particular object to break the assumption that objects (and the world) have no high level semantic history or memory. Many systems currently do this to implement an undo and redo mechanism for either the world as a whole or for a specific object, or to allow users to visualize that changes that have occurred recently to an object or the world.

Of course, the user is a special type of object in the scene. This means that another assumption we can break is that the actions the user can perform and their effects are independent of the current context or task. In the real world the fact that I am walking ten feet or ten miles does not change how fast I can walk, despite the fact that if I was really walking ten miles it would be useful to be able to walk at a much faster rate. Point of interest navigation [Mackinlay90] provides this ability: the user covers a fixed percentage of the remaining distance to the target every frame. The user thus automatically moves faster when further from the target and slower when closer to it. Other researchers have experimented with similar ideas. Pierce implemented a navigation technique where the user pulls himself to objects by moving his hand from the point where he specifies the target to his eye [Pierce97]. Because the user always reaches the target when her hand reaches her eye, the user's maximum velocity is determined by the chosen destination. Ware created a technique that scales the user's motion depending on the surrounding context [Ware97]. The user moves faster in wide open spaces, and slower when closely surrounded by other objects.

As another example, consider turning your head left and right. In the real world this always causes you to look left and right. However, in a virtual world this can have different effects depending on your task. If you are navigating around the world, turning your head left and right may have the normal effect. If you are inspecting an object, then turning your head left and right could cause your viewpoint to rotate around that object [Koller96]. However, designers must be careful when breaking assumptions in this way, because creating a different result to a physical action can break the kinesthetic correspondence principle [Britton] and be difficult for the user to grasp.

The actions available to a user could also depend on the actions of another user in a multi-user world. For example, Hindmarsh shows that a user often has trouble determining exactly what object another user is referring to, especially if he cannot see both the referrer (the pointing user) and the referent (what the user is pointing at) at the same time [Hindmarsh]. A possible solution, Hindmarsh suggests, is to allow the act of one user pointing at an object to make it easy (by providing a new action or shortcut) for other users to look at that object.

When we act in the real world we assume that we control our viewpoint and our actions, yet we can occasionally simplify the user's task by breaking this assumption. This is particularly true in desktop 3D environments where the user realizes that a slightly different viewpoint would useful while performing another task. Rather than interrupt the task, change the viewpoint, and resume the task the user can rely on the system determining the most useful viewpoint. [Phillips][Drucker][He] have all experimented with different techniques for automatically changing the user's viewpoint to provide interesting or more useful views, and some video games control the user's viewpoint to free the user for other tasks. For example, when the user is moving an object around the system can rotate the user's view or move the view forward or backward so that the object remains visible. Breaking this assumption may be less useful in immersive worlds where the user's head controls the view and her hands perform the task.

When we change our viewpoint we assume that we move through the world, the world does not move around us (with the possible exception of earthquakes). In the physical world there are two main methods of locomotion: either we move directly or we move using a vehicle. However, in a virtual world we can expand this to include a third method. Ware terms this the *scene in hand* metaphor [Ware90]: the user can change his position by grabbing the world and moving it around.

In addition to thinking about our actions in different ways, we can think about abstractions and physical objects in different ways. People are taught that abstractions are not represented by physical objects. The abstract idea of space is the lack of physical objects, not a physical object in itself. In a virtual world, however, we can turn abstract objects into physical objects. We can make space a physical object that a user can grasp so that he can move around the scene by grasping space and pulling himself around [Liang][Robinett][SmartScene]. We can also give physical form to a user's field of view or to the act of referring to an object to provide extra information to other users in the world [Hindmarsh]. Another way to break this assumption is to take an amorphous abstract entity like a crowd (how many people are required for a crowd?) and reify it as a specific physical entity. This new entity can be completely different than a group of people, possessing both a different appearance and different physical properties [Benford].

Beyond giving physical form to the abstract, we can break people's assumptions about familiar, everyday objects. We can break the assumption that effects follow causes. Herndon breaks this assumption by allowing users to grab shadows and drag them around to change the position of the object casting the shadow [Herndon]. This can be useful both to constrain the motion of the object (the shadow can only be dragged along the surface it is projected on) and in situations where the shadow is visible but the object is not. Another approach is to allow users to draw in shadows themselves to affect the height or position of objects [Zeleznik]. A virtual world can also allow the user to interact with objects using their shadow. Krueger essentially takes this approach by allowing the user to interact using their silhouette, which can be thought of as the user's shadow projected onto the display screen [Krueger].

In addition to playing with shadows, we can break the assumption that objects work the same in the physical and virtual worlds. We can create mirrors in a virtual world that are not limited to merely reflecting their surroundings. In a virtual world whether or not and how a specific object appears in a mirror can convey information about that object to the user. For example, a mirror may only display those objects in a shared workspace that are also visible to other users [Butz]. We can also create lamps that affect the properties of the objects they illuminate. For example, the user may use a Privacy Lamp [Butz] to indicate that objects illuminated by the lamp are invisible to other users.

2.2 Assumptions We Can Productively Break

Taken as a whole, the literature of existing interaction techniques and the techniques presented in this thesis suggest that we can generate interesting and valuable interaction techniques by breaking the following assumptions:

1. Space is linear and continuous.
2. The visual properties of a space match the physical properties of the space.
3. Appearance does not necessarily reflect reality.
4. There is only one world.
5. Everyone in the world occupies the same point in time.
6. The world is persistent even when not in view.
7. The world is spherical.
8. Gravity exists.
9. Objects do not arbitrarily attract each other.
10. The intrinsic and extrinsic properties of objects are inviolate and distinct.
11. Objects are persistent: we cannot create or destroy them on command.
12. Objects can only exist in one location.
13. “Passive” objects cannot affect other objects.
14. Objects work the same in all contexts.
15. Objects have no built-in knowledge of their properties or functionality.
16. Objects (and the world) have no high level semantic memory or history.
17. The actions the user can perform and their effects are independent of the current context or task.
18. We control our viewpoint and our actions.
19. We move through the world, the world does not move around us.
20. Abstractions are not represented by physical objects.
21. Effects follow causes.
22. Objects work the same in the physical and virtual worlds.

2.3 Breaking Vs. Shaping Assumptions

Beyond the boundaries of computer science, both storytelling and stage magic are intimately connected with assumptions. Storytelling creates assumptions and defines the audience's expectations for an imaginary world. Stories set general expectations by fitting into a genre. For example, in a horror story we expect that spooky music means that something scary is about to happen. Stories also set expectations by creating a particular world. Subsequent stories set in the same world can draw on those established expectations. For example, in any given Road Runner cartoon we expect that the coyote will come up with some ingenious scheme that will inevitably fail, most likely harming himself in the process. We can draw on the expectations engendered by a particular genre or story world by basing a virtual world on an existing story (often called a *backstory*). If we allow our users to step into a Road Runner cartoon they will have a specific set of assumptions about what is and is not possible based on their prior experience with Road Runner cartoons in particular and Saturday morning cartoons in general. These same users will have completely different assumptions if we allow them to step into a scene from Star Wars. We can also try to create a completely new backstory for a virtual world. However, in this case we will need to ensure that users do not spend more time learning the story than they do interacting in the virtual world!

Not only can we learn lessons about how to shape assumptions from storytelling, we can also draw ideas for interaction techniques from existing stories. These techniques can leverage the story they are based on to make the techniques easier to learn. For example, the *Aladdin's Magic Carpet Ride* attraction [Pausch96] at DisneyQuest [DisneyQuest] draws the idea of navigating using a flying carpet (as well as the setting and characters in the world) from Disney's animated feature *Aladdin* (and originally from Scheherazade's *Arabian Nights* tales). Because users are already familiar with the idea of flying carpet, despite their lack of experience with them in the real world, they have little trouble accepting this navigation technique.

We can also learn lessons about how to control or take advantage of assumptions from stage magic. Magicians need to understand and control the audience's assumptions to prevent the audience from noticing any sleight of hand. Tognazzini suggested that we can borrow stage magic's ideas and techniques to create user interfaces with a compelling "user illusion" [Tognazzini]. We might also borrow these techniques when we want to break assumptions about 3D worlds without the user noticing, although our ability to alter the "reality" of our virtual worlds gives us a significant advantage over magicians.

2.4 A Caveat

By arguing that we need to create interaction techniques that break our assumptions I do not intend to suggest that all of our techniques should break assumptions. We have developed significant skills and knowledge interacting with the real world, and there are tasks at which "real world techniques" will probably always be better than techniques that break assumptions. For example, walking is very tough to beat for moving short distances.

The debate on how closely to base computer interfaces on interaction with the real world is not new. Looking back even before 3D interfaces, the use of real world metaphors in 2D interfaces was (and still is) often hotly debated: their proponents argue that the use of metaphors makes user interfaces easier to learn, while their opponents argue that they limit our users thinking and the capabilities that we can provide.

Randy Smith summarized the benefits and drawbacks in a paper exploring the tension between literalism and magic [SmithR]. Although he focused on adherence to an interface's metaphor within the interface itself, I believe that his discussion is also applicable to the choice of whether or not an interface should adhere to the real world. From this perspective, the literal features in an interface work as closely as possible as they would in the real world. Magic features, by contrast, deliberately violate the real world metaphor to provide advanced functionality. The advantage of literalism is learnability, because users can draw on their experience with the real world. The disadvantage is that users are limited by what is possible in the real world. The advantage of magic is power: the ability to go beyond what is possible in the real world. The disadvantage of magic is obscurity, increasing the time required to learn the interface. Because the advantages and disadvantages of literalism and magic are exactly opposite, designers need to carefully choose the balance between them based on the requirements of their application.

Do do that voodoo that you do so well
Cole Porter

CHAPTER 3 *Voodoo Dolls*

In the real world we are constantly manipulating objects: we pick them up, push them, prod them, poke them, and put them back down. This trend seems likely to continue for virtual worlds, as many of the proposed VR applications require manipulating 3D objects. Chemists creating new compounds will need to experiment with different 3D arrangements of molecules, interior designers will need to try different furniture arrangements to assess their impact, and mechanical engineers will need to make sure that all of the parts for a design fit together smoothly. Researchers have thus spent considerable effort developing new techniques for object manipulation in 3D environments.

The simplest object manipulation technique draws heavily on our experience with the real world. Users touch a virtual object, grab it, move their hand to the desired position and orientation, and let go. While very easy for novice users to learn, this technique also has severe drawbacks. The requirement that users touch objects to manipulate them reduces users to only working with objects within arm's reach. To work with other objects a user has to first move to their location. The same requirement restricts placement: to place an object in a location outside his reach the user must first move to that location. In a large environment a user could spend more time moving around than actually manipulating objects. This technique can also make it difficult for users to judge whether a particular placement achieves the desired effect. Consider hanging a picture the virtual world. To manipulate the picture the user has to be standing next to it, but to judge whether the picture is level he needs to look at it from a distance. Finally, this technique makes it difficult to manipulate extremely large objects. For example, an urban planner may need to experiment with different placements for a new skyscraper. However, when he is close enough to manipulate the skyscraper it will effectively block his view of the rest of the scene. To overcome these drawbacks, researchers started looking for new ways to manipulate objects in virtual environments. Although they may not have done so explicitly, they started to break their real world assumptions.

The Worlds in Miniature technique [Stoakley] breaks the assumption that objects can only exist in one place at a time by giving the user a miniature hand-held copy of the world. This miniature copy exactly mirrors the larger world and vice-versa, so that each one reflects any changes to the other. This essentially brings all of the objects depicted in the WIM within reach. To manipulate a distant object the user grabs its copy in the hand-held miniature. Unfortunately, because the WIM needs to be small enough for the user to hold in his hands, it can potentially introduce a very large linear mapping between the WIM and the world. In other words, if the WIM mirrors a very large world then the individual copies it contains will extremely small, and small motions of these copies will result in large motions of the actual objects. The designer can reduce this mapping by including less of the world in the WIM, but then the user is restricted to only manipulating objects contained in the WIM.

Rather than giving the user fixed-scale copy of the world, SmartScene [SmartScene] breaks the assumption that the world is a constant size by allowing the user to grab the world itself and scale it up or down. The drawback to this technique is that, rather than traversing distance to manipulate objects, the user has to traverse scales. The user first scales down the world so that the desired object is within reach, and then scales it up again so that the object is large enough to pick up. The user then again scales down the world to bring the desired location within reach, and scales the world back up so that the world is large enough for him to accurately place the object. The time required to constantly scale the world can be comparable to the time required to traverse the world for the simple technique above.

The World-Scaling technique [Pierce97][Mine] also breaks the assumption that the world is a constant size, but rather than making the user manually scale the world it scales the world instantaneously. The user specifies the object he wants to manipulate using an image plane selection technique [Pierce97]. In response the system instantaneously scales the world down so that the user's hand is touching that object. The user can then place the object anywhere within reach in the scaled world. The ratio of the object's original distance to the distance between the user's hand and his head not only determines how much to scale down the world, it also determines how far away the user can place the object and with what accuracy. The farther the object's initial position, the larger the ratio and the farther into the world the user's arm will reach at full extension. However, the larger the ratio the larger the mapping between small movements of the user's hand and changes in the object's position.

Researchers have also explored techniques that scale the user's reach instead of the world. The Go-Go technique [Poupyrev96] breaks that assumption that the user's virtual hand occupies the same position as his physical hand. When the user moves his physical hand within some fixed radius from the center of his chest, his virtual hand does indeed match the position of his physical hand. When the user reaches beyond this radius, however, his virtual hand reaches out farther than his physical hand along the vector from the center of his chest through his physical hand. The world's developer specifies a mapping function that determines how far the virtual hand moves in response to a movement of the physical hand. This mapping can be linear or non-linear. While this technique allows the user to manipulate objects that are outside his physical reach, it still places bounds on what objects a user can manipulate and where he can place them because even with a non-linear mapping the user can only extend his physical hand a set distance. The designer can extend these bounds by increasing the mapping from physical to virtual, but at the cost of resolution: the user will be unable to accurately position his virtual hand at a distance between small motions of his virtual hand will result in large motions of his virtual hand.

The HOMER technique [Bowman97] is a more general "arm extension" technique that allows users to extend their virtual hand to touch any visible object in the scene. The user points at the object he wants to manipulate and selects it. In response the system casts a ray into the scene to determine what object the user is pointing at, and then instantaneously extends the user's virtual arm so that his virtual hand touches that object. The volume within which the user can subsequently place the object depends the chosen incarnation of the HOMER technique. The first incarnation mimics the Go-Go technique: once the user selects an object the system creates a mapping function that linearly maps the distance from the user's hand to his body to the distance to the selected object. This imposes a maximum distance at which the user can place the object, as well as determining the resolution at which the user can place the object. The farther away the object is initially, the farther away and less accurately the user can place it.

The second incarnation of the HOMER technique uses a 1:1 mapping function. With this incarnation a user can relatively accurately place the object, but only within arm's length of object's initial distance. To overcome the latter problem the technique allows the user to "reel" the object in or out at a constant velocity. Although this incarnation thus allows the user to place objects more accurately, it imposes a minimum placement time that increases linearly with the distance between the object's initial position and its desired position.

None of these techniques are ideal. In order to increase the range within which he can select and place objects, the user of any of these techniques pays either with reduced accuracy or increased placement time. With the exception of the WIM technique, none of these interaction techniques allow users to easily manipulate occluded objects. Those manipulation techniques that rely on ray-casting or image plane selection also suffer from the problem that selecting objects that occlude a small visual angle on the user's image plane (e.g. small, distant objects) is very difficult. By breaking additional assumptions about the real world, however, I can overcome these limitations.

3.1 The Voodoo Dolls Technique

To create the Voodoo Dolls technique I started by breaking the assumption that objects can only exist in one place at the same time. By providing the user with handheld copies of objects I allow the user to simultaneously view the relative positions of objects up close and at a distance. This allows the user, without moving, to verify that the object is accurately positioned and that the position achieves the desired effect.

The WIM technique breaks this same assumption by giving the user a single, persistent copy of the world. This approach imposes the aforementioned trade-off between range and accuracy: making the WIM correspond to enough of the world to allow the user to manipulate any object in the world reduces the accuracy of object placement and makes it difficult to grab and manipulate small objects. I could add controls to change the center of focus and scope of the WIM, but this would replace the trade-off between range and accuracy with the trade-off between range and placement time, because the user would then have to spend time scrolling and scaling the WIM. The key problem is that the WIM is persistent. Changing the WIM's focus and scale requires a continuous transition whose duration is proportional to the size of the change. What we need is a method that allows a discontinuous jump to a more suitable focus and scale. This is, in essence, what the Voodoo Dolls technique provides.

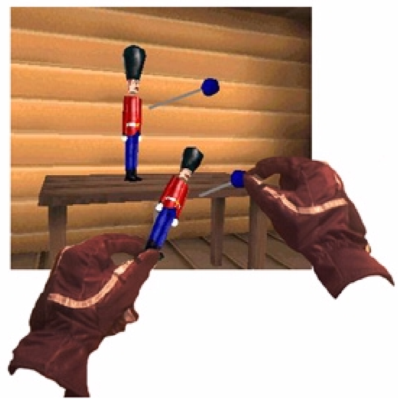

FIGURE 3-1. The Voodoo Dolls technique allows users to manipulate objects through handheld copies of them. Users do not see the gloves; I added them to the images in this chapter to aid exposition.

The Voodoo Dolls technique goes beyond the WIM technique to break two additional assumptions. First, the technique breaks the assumption that objects in the world are persistent by allowing users to dynamically create and destroy *dolls*. Dolls are hand-held, transient objects: users create dolls to represent specific objects, and the dolls vanish when users no longer need them. Visually, dolls are small copies of the objects they represent. Functionally, users employ dolls in pairs, one in each hand, and the dolls serve two different yet complementary purposes depending on the hand holding them. This functional difference breaks the assumption that the behavior of objects is independent of the hand holding them.

In the case of a right-handed user (the user's right hand is the dominant hand, while his left hand the non-dominant hand), the doll held in his right hand determines the position and orientation of the object it represents, while the doll held in his left hand sets the reference frame. In other words, when the user moves the doll in his right hand relative to the doll in his left hand, the object represented by the doll in his right hand moves to the same position and orientation relative to the object represented by the doll in his left hand. Consider a user who wants to put a lamp on a table. The user creates dolls for both objects, and holds the table doll in his left hand and the lamp doll in his right hand. Now, placing the lamp doll on top of the table doll causes the lamp to move on top of the table.

Users can thus quickly change the focus and scale at which they can place an object by just deleting their current reference doll and creating a new one for a different object. A user designing a virtual town can move a house to the other side of town, put a dog next to a doghouse, and place a bee on a flower, all without explicitly changing scale, just by choosing the correct reference doll; the choice of reference object determines both the focus and the scale at which the user can work.

The Voodoo Dolls technique also takes advantage of how people naturally work with their hands. Guiard [Guiard] demonstrated that in the real world the dominant hand works in the reference frame defined by the non-dominant hand. Consider signing a baseball: the baseball is much easier to sign when held in the non-dominant hand than when sitting in a stand on a table. Subsequent studies have confirmed that people use their hands in the same way for human-computer interfaces [Buxton][Hinckley97a] [Hinckley97B].

3.1.1 Creating and Destroying Dolls

The user creates a doll using an image plane technique [Pierce97]: he positions his hand so that the crosshair attached to that hand lies over an object in his view and pinches together the index finger and thumb of that hand (see Figure 3-2). Note that this image, and others in this chapter, only shows the user's hand for pedagogical clarity. In the virtual world the system represents each of the user's hands with a crosshair; a crosshair communicates more effectively the exact selection point, and is small enough not to occlude much of the scene. Selecting an object creates a doll scaled to a convenient working size, positioned at the user's hand, and attached to it. When the user is finished working with a doll he simply lets go of it (stops pinching his fingers together) and the doll vanishes.

FIGURE 3-2. Users create dolls using image plane selection.

3.1.2 Sizing Dolls

Because the user is going to manipulate dolls relative to each other, the dolls need to maintain the same size relative to each other. In addition, dolls should ideally be a comfortable working size regardless of the size of the actual objects they represent. The Voodoo Dolls technique scales dolls so that the doll that representing the larger object in the world is half a meter along its largest dimension and the other doll maintains the correct relative size. This approach does require choosing an appropriate reference object to be most effective. If the user tries to put a bee on a flower by choosing the house next to the flower as the reference object instead of the flower itself, the bee doll will be much smaller and harder to manipulate. This approach also naturally allows users to work more accurately when manipulating smaller objects.

3.1.3 Passing Dolls From Hand to Hand

Users can pass dolls from hand to hand. For example, to pass a doll from his right to his left hand, the user moves the cursor attached to his left hand to the doll and pinches the thumb and forefinger of that hand together; releasing the right hand's pinch now transfers control of the doll to the left hand and changes the doll's mode. The system preserves any changes made to an object's position when the user passes it from hand to hand.

3.1.4 Undo

Because dolls enforce the relative positions of the object they represent as soon as the user is holding two dolls, the user can accidentally move an object by creating a doll for the wrong object with his right hand when his left hand is already holding a doll. For these cases I provide a limited form of undo using Mine's "over the shoulder" deletion technique [Mine]: the user can reset an object's position and orientation by holding the doll that represents it over his shoulder and releasing it. The object will return to the position and orientation it possessed when the user created that doll for it.

3.1.5 Providing Context

When the user holds a doll in his left hand, the system can create dolls for nearby objects to provide some context for the interaction. The system places these new, scaled dolls relative to the doll in the user's left hand and makes them move with it. Consider arranging books on a bookshelf: the task is easier if the user can see the other books on the bookshelf.

The dolls in the context are normal dolls: the user can grab one with his right hand and use it to reposition the object it represents. This allows the user to manipulate objects that are occluded or too small on his image plane to select easily. For example, to move a coffee cup that is hidden behind a lamp on a desk, the user creates a doll for the desk and holds it in his left hand; with his right hand he grabs the coffee cup doll from the context and changes its position. The context disappears when the user releases the doll in his left hand or passes it to his right hand.

I experimented with a number of rules for choosing dolls to add to the context. The rule our users preferred for implicitly specifying context is for the system to create dolls for objects within a specified radius and add them to the context. In the current implementation this radius depends on the size of the reference object: dolls are created and added to the context for all objects that are closer to the reference object than twice the longest dimension of the reference object. Thus if a book on a bookshelf is the reference object a few nearby books will be added to the context, but if the bookshelf is the reference object all the books on it will be added.

Another possible rule is to consider semantic relationships, so that objects on a table would be added to the context, but a chair nearby would not. The authoring tool I used to build Voodoo Dolls, Alice [Conway], implements semantic relationships through a parent-child hierarchy: any translations, rotations, and size changes that affect a parent object will also affect its children. Voodoo Dolls draws on this hierarchy to create the context. When users hold a doll in their left hand, the system recursively creates dolls for all the children of the reference object and adds them to the context.

The user can also explicitly indicate what objects to include in the context. Using both hands the user can frame on the image plane (Figure 3-3) the objects to be included in the context; the user pinches the thumb and middle (as opposed to the index) finger of his right hand together to start framing, and creates the context by pinching together the thumb and index finger of his left hand. The system adds any object that is completely visible in the frame to the context. Alternately, the system could allow the user to explicitly specify a radius around a reference object (Figure 4): the system would then add any object completely within that radius to the context.

The user can change the radius of an existing context by pinching the thumb and middle finger of his right hand together and moving his right hand closer to or farther from his left hand. The distance between his hands at which he releases the pinch determines the new radius of the context.

FIGURE 3-3. A user can explicit specify the context by framing the desired objects on his image plane.

FIGURE 3-4. A user can also explicitly specify a context by specifying the desired center and the radius around it.

3.1.6 Example

The following example illustrates the Voodoo Dolls technique. Imagine a theatrical director using virtual reality techniques to experiment with different layouts of set furniture for a stage play. He is sitting thirty rows back in a virtual theater, and wants to move a telephone onstage from a desk to a coffee table. If the director is right handed, he creates a doll for the desk by “grabbing it” with his left hand. To help provide context (see section 2.3), the system creates dolls for the objects on the desk, which appear on the desk’s doll in the director’s left hand. With his right hand, the director reaches into the context and grabs the doll for the telephone. He releases his left hand’s grip, destroying the desk doll and its context, then creates a doll for the coffee table with his left hand. He places the telephone doll on the coffee table doll, and lets go of both. During this process, whenever both his hands are holding dolls the onstage object represented by the doll in his right hand is positioned relative to the onstage object represented by the doll in his left hand.

3.1.7 Creating Dolls: Special Cases

Occasionally the user will want to move an object somewhere where there is nothing nearby to serve as a reference object. For example, the user might want to place a sofa in an empty corner of a room. The user could choose a reference doll with a sufficiently large context to make this placement. For our example the user could create a reference doll for the room itself. The drawback to this approach is that choosing a doll with a large enough context may reduce the available placement accuracy to an unacceptable level.

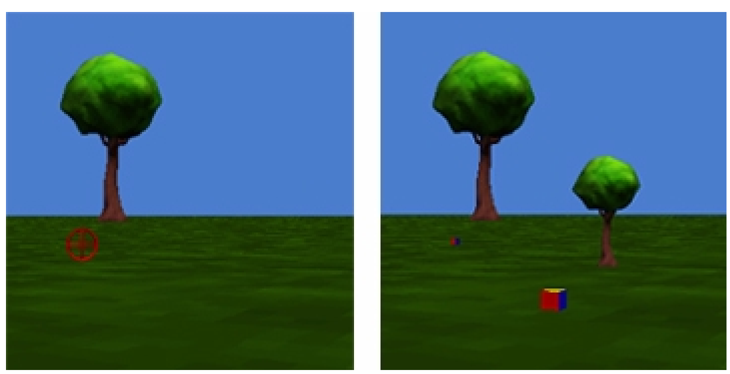

FIGURE 3-5. If the user tries to create a reference doll for a non-existent object, the system creates a temporary reference object for him.

If the user wants to merely adjust the position of an isolated object slightly, he can instead create two dolls for the same object. The doll in the user’s left hand represents the object’s initial position and makes it the frame of reference. The doll in the user's right hand thus sets the object's new position relative to its initial position. If the user instead wants to move the object to another location, however, this approach will not help.

To address the latter case I allow the user to simultaneously create a temporary object and its doll by selecting a point on the ground or floor where he wants the object to appear. The temporary object appears as a transparent colored cube half a meter long on each side, and disappears as soon as the user destroys the doll representing it. I provide a context for the doll whose size depends on the distance of the temporary object to the user; the farther the temporary object, the larger the context around it.

3.2 Implementation

This technique differs from some previous two-handed techniques [Hinckley94a][Stoakley] by using gloves, rather than hand held physical *props*. Props provide affordances for interaction, and are most useful when the user works with objects that resemble the props in some way. Because the user can manipulate any object with this technique, there is no "generic prop" that provides affordances for every case. Instead, users wear FakeSpace PinchGloves that detect contact between the tips of the user's fingers [Mapes].

As with other techniques that use image plane selection, the user must designate his dominant eye prior to using the system if the system displays the world in stereo. The system can then use the image displayed to that eye to determine what object the crosshair is over when creating dolls. The current implementation avoids this difficulty because it renders a bi-ocular image that is presented to both eyes (in our lab, and some others, bi-ocular rendering is standard because of the higher frame rate). Note that image plane techniques do not necessarily have to entail ambiguity when selecting objects using a stereo display. The system can render the crosshair in a slightly different position for each eye so that the crosshair appears over the same object for both eyes. With my implementation the user must still designate whether he is right or left-handed because dolls created with the right or left hand have different characteristics. I could have built the system to infer this information (e.g. asking users to grab an object and recording which hand they use), but under formal experimental conditions asking users and setting handedness is both easy and certain.

3.3 Evaluation

The Voodoo Dolls technique works for a broad range of object sizes, allows the user to manipulate both nearby and distant objects, and allows the user to accurately position objects when the target position is nearby or distant. To evaluate this technique, therefore, I wanted to compare it against an equally versatile technique. The four existing best practice general manipulation techniques for VR are Go-Go [Poupyrev95], World-scaling [Mine], Worlds in Miniature [Stoakley], and HOMER [Bowman97]. I analyzed each of these techniques to determine the best comparison technique.

Go-Go. The Go-Go technique works best for nearby objects. For more distant objects, the designer is limited to a fundamental trade-off between range and resolution. Because the user can only extend their arm so far, to allow the user to reach farther the designer has to increase the non-linearity of the mapping between the physical and virtual hands. This increases the user's range, but at a cost to accuracy. I chose not to compare against this technique because of the lack of accuracy when manipulating distant objects.

World-scaling. World-scaling works best when the target's initial position is not too distant from the user, and when the final position is closer to the user than the initial position. When the target object is too far from the user the world must scale down so much that the user loses accuracy: small movements of his hand result in large changes in position. In addition, the size the world shrinks to is set when the user selects an object, so that the user cannot place an object any farther than he can reach in the scaled world. I chose not to compare against this technique because of the loss of accuracy when manipulating distant objects, and because of the restriction on placement after selection.

SmartScene. SmartScene works best when the user works with objects that are approximately the same size and located close together. If the objects vary widely in size and position then the user spends much of his time scaling the world up and down and pulling himself through the space. I chose not to compare against this technique because of the time required by these continuous transitions between location and scale, and because users usually have to change their position to accurately manipulate objects.

WIM. A WIM works best when the size of the objects to manipulate is appropriate for the size of the WIM. For example, given a WIM for a room manipulating the furniture in the room is extremely easy. However, the WIM also has a trade-off between size and accuracy. Because the user can only manipulate objects in the WIM the WIM has to show every object that the user must manipulate. However, the more the WIM contains the larger the mapping between WIM space and "real" space. With a large mapping users will have difficulty selecting small objects because they become miniature in the WIM, and small motions in the WIM will result in large motions in the world, making accurate placement difficult. I chose not to compare against this technique because of the loss of accuracy when manipulating and difficulty in selecting small objects in a large world.

HOMER. I could compare Voodoo Dolls to either variant of the HOMER technique. In the first, Direct HOMER, the system linearly maps the distance between the user's body and his physical hand to the distance between his body and his virtual hand after he selects an object. This variant suffers the same accuracy problem as the previous techniques: when the user selects a distant object the size of the mapping makes it hard to accurately position the object. However, the second variant, Indirect HOMER, preserves a 1:1 mapping between the motion of the user's physical and virtual hands, and allows the user to "reel" the object in or out by pushing a button.This incarnation allows users to accurately manipulate both nearby and distant objects. A drawback to the HOMER technique in general is that selecting small, distant objects can be difficult; however, all of the techniques also suffer from this problem. A drawback to Indirect HOMER in particular is that the target object must travel the entire distance between its initial and final positions. While the designer can increase the travel velocity, this can result in users overshooting the final position and needing to recover. Despite the latter drawback, I chose to compare against Indirect HOMER because it alone of these techniques allows the user to accurately position objects both nearby and far away.

3.3.1 Task Descriptions

Researchers have developed a number of testbeds for evaluating VR interaction techniques, including the VEPAB testbed [Lampton], the VRMAT testbed [Poupyrev97], and VR-SUITE testbed [Bowman99b]. Unfortunately, because these testbeds are built around existing interaction techniques they often do not provide tests that adequately compare new techniques with existing techniques. For example, with existing techniques manipulating small, far away objects is extremely difficult, so these testbeds do not compare how existing techniques fare at that task. The VEPAB testbed in particular assumes that only objects within reach can be manipulated. The tasks in these testbeds also tend to entail manipulation of abstract shapes instead of real world objects. The manipulation tasks in the VRMAT testbed all consist of placing one cylinder atop another, and the manipulation tasks in the VR-SUITE testbed all entail the manipulation of cubes. In addition, the VR-SUITE manipulation tasks focus on testing placement speed instead of placement accuracy.

I believe that these testbeds suffer from a common failing: their developers chose their tasks so that they could compare techniques while changing a single variable at a time. There is a general tension [Brooks88] between comparing the usability of interaction techniques for specific situations versus possible combinations of input variables (size, shape, initial position, and initial orientation of the target object, final position and orientation of the target, how crowded the world is, whether the initial or final positions are occluded, etc.). The advantage of choosing specific situations is that the designer can choose representative tasks (e.g. arranging furniture in a room), but the disadvantage is that the technique's performance will not necessarily generalize to other situations. The advantage of varying a single variable at a time is that the designer can, provided he examines enough cases, argue that his results are generalizable. The disadvantage is that the tasks, as in the above testbeds, usually consist of moving platonic objects in a featureless environment, and may not be at all representative of how users actually employ the techniques. Given a choice, I prefer the former, so for this experiment I designed the tasks to be representative of typical manipulation tasks while focusing on key difference between the two interaction techniques:

- The HOMER technique allows users to manipulate objects as accurately as if the user was actually holding them in his hand, while the Voodoo Dolls technique scales the effects of the user's hand motions based on the size of the manipulated and reference objects. The larger the objects the user manipulates, the less accurately he can position them.
- The HOMER technique forces users to work at a distance, while the Voodoo Dolls technique allow users to simultaneously work up close and at a distance.

To compare the Voodoo Dolls and HOMER techniques I created one practice condition and five different task conditions divided among indoor and outdoor locales. Each task condition is composed of 3 placement tasks, for a total of 15 tasks. Some of these tasks are designed so that there is no nearby object to serve as a Voodoo Dolls reference object.

3.3.1.1 Practice

The practice world initially contains only a ground plane. Within this world users complete five practice tasks. When the user starts a practice task the system displays the target object to move, the target position, and any reference objects for that task. The practice tasks consisted of moving a chair in front of a desk, placing a television set on a desk, repositioning a toy on a table, moving a distant cement mixer, and moving a distant biplane. All users encountered the practice tasks in that order.

3.3.1.2 Indoor Tasks

The indoor tasks take place in a living room. The user stands in the middle of the room, surrounded on all sides by the room's furnishings. The dimensions of the furnishings and the placement distances for this task, and for the outdoors task, are contained in the appendices.

FIGURE 3-6. A top-down view of the living room for the study.

Furniture (medium: 1 - 2 meters in size)

This task group studies how well the techniques allow users to manipulate medium size objects when both the initial and final distances to the target object are within 10 meters. Users have to move three different pieces of furniture (a desk, a comfy chair, and an end table) from their initial positions to their target positions. My hypothesis was that HOMER would perform slightly better than Voodoo Dolls on this task that because the objects are close and large enough for HOMER users to judge the accuracy of their placements as well as Voodoo Dolls users without imposing the penalty of working in a scaled space.

FIGURE 3-7. The furniture users manipulated in the study.

Bookshelf (small: 0.2 - 0.5 meters in size)

This task group studies how well the techniques allow users to manipulate small, nearby objects when the difference between the initial and final position is small. Users have to rearrange a book and two bookends on a nearby bookshelf. My hypothesis was that the Voodoo Dolls and HOMER techniques would fare equally well on this task. HOMER users would not have to work in a scaled space, but Voodoo Dolls users would be better at evaluating placement accuracy.

FIGURE 3-8. The small objects users manipulated on the bookshelf.

3.3.1.3 Outdoor Tasks

The outdoor tasks take place in an amusement park. The user stands in the middle of the park, surrounded on all sides by the rides, food carts, and game booths.

FIGURE 3-9. A bird’s eye view of the outdoor scene for the study.

Rides (large: 15 - 140 meters in size)

This task group studies how well the techniques allow users to manipulate large, far away objects when the target position is far away from the initial position. Users have to move three amusement rides from their initial positions to their target positions. My hypothesis was that the Voodoo Dolls technique would be slightly more accurate for this task because, despite the size of the objects, users would be placing them far enough away that Voodoo Dolls users will be better able to evaluate object placements.

FIGURE 3-10. The amusement park rides users manipulated in the study.

Booths & Carts (medium: 1.5 - 5 meters in size)

This task group studies how well the techniques allow users to manipulate medium size, far away objects when the target position is far away from the initial position. Users have to move one game booths and two food carts from their initial positions to their target positions. I expected that the Voodoo Dolls technique will perform better here because, as the target objects get smaller, the ability to verify placement accuracy becomes more and more important.

FIGURE 3-11. The carts and booth users manipulated in the study.

Prizes (small: 0.5 - 1 meters in size)

This task group studies how well the techniques allow users to manipulate small, far away objects when the target position is far away from the initial position. Users have to move three prizes (stuffed animals) from one game booth to another game booth. I expected that the Voodoo Dolls technique would perform significantly better here, because the combination of small targets and distant locations would make verification the predominant factor.

FIGURE 3-12. The prizes users manipulated in the study.

3.3.1.4 Task composition

Before the start of each task I presented the target object and target position (indicated by a transparent copy of the target object) to the user. I guided the user verbally until he located the target object in the world, and then repeated the process until he located the target location. Guiding the user until he located both the target object and location allowed me to locate the target object and target position so that they were not necessarily simultaneously visible. When the user had located both the target object and location, I asked the user to announce when he was ready, and then told him to begin. Figure 3-13 shows what a user might see while completing the outdoor task involving the Ferris Wheel.

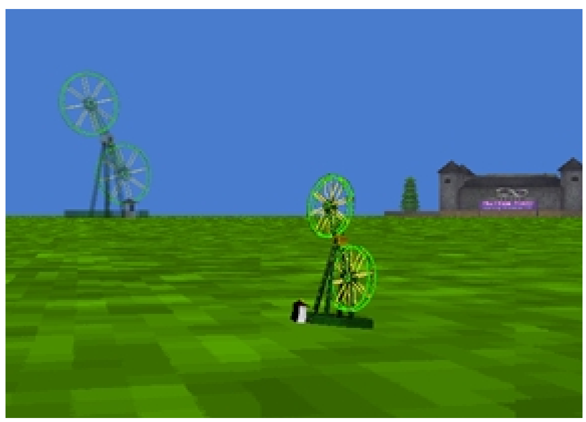

FIGURE 3-13. A user holds a doll a doll for the Ferris Wheel in his dominant hand, while in the background the transparent copy of the Ferris Wheel indicates the target location.

The user's goal for each task was to manipulate the target object to match the copy's orientation and position as closely as possible. To avoid discriminating against the HOMER technique I did not allow the user to directly create a doll for the target copy with the Voodoo Dolls technique. Users could create a doll for the target copy indirectly by creating a reference doll that contained the target copy in its context. Users completed each task by destroying their dolls and announcing "Done" when they were satisfied with the placement of the target object.

3.3.2 Method

Twelve undergraduate and graduate students, ten male and two female, participated in this experiment. Half of the users used the Voodoo Dolls technique, while the other half used the HOMER technique. I chose a between-subjects design rather than a within-subjects design primarily to reduce the amount of time users had to spend in the head-mounted display (HMD). I balanced the locale order (indoors and outdoors) and the task group order within a locale between users, and randomized the task order within a task group for each user. Before starting the experiment users completed a simple questionnaire to determine their gender, age (within a 5 year range), handedness, computer experience, and previous experiences with virtual reality.

I demonstrated the relevant interaction technique for each user by performing two example placement tasks in the practice world. For the HOMER technique I demonstrated how to use ray-casting to select an object, how to position the object around its initial position, and how to reel an object in or out. For the Voodoo Dolls technique I demonstrated how to create a doll, how to grab a doll from the reference doll's context, and how to create a reference doll if there was no useful reference object near the target position.

Each user then completed all five practice tasks. I instructed users to focus on placing the objects as accurately as possible, and after each practice task I provided users with both numerical and visual feedback on their accuracy for that task. While I read off the distance error (in centimeters) and orientation error (in degrees), the system shrank the target object and the translucent copy, moved them in front of the user (maintaining their position and orientation relative to each other), and slowly rotated them over three seconds. The system then moved the target object to the copy over one second, rotated the target object to match the correct orientation over another second, and then scaled the target object and copy back up and returned them to their initial positions.

Before each task group I explained in general terms what the tasks entailed and reminded the user to concentrate on accuracy. I started tasks when users announced they were ready, and completed them when users had dropped the target object or dolls they were holding and announced that they were satisfied with their placement. I did not provide any feedback during the experiment tasks. Between each task I reset the objects back to their original positions.

In order to protect users from prolonged exposure to the virtual environments, I allowed users to remove the HMD and take a short break between the practice and actual tasks, and between the indoor and outdoor locales. These breaks meant that users were exposed to the virtual world for at most twenty minutes at a time. I also allowed users to take a break at any time if they started to feel dizzy or nauseous. One user did briefly feel dizzy, but recovered after a short break.

After completing all the task groups users completed a short questionnaire to determine what was easy and what was hard with the interaction technique they used, and whether they had experienced any discernible arm fatigue, dizziness, or nausea.

3.3.2.1 Apparatus

I implemented the virtual worlds in Alice 99 [Conway]. For the experiment I ran the worlds on a Pentium III PC. All users viewed a bi-ocular version of the world in a Virtuality Visette Pro HMD (640 x 480 resolution, 60 degrees x 46.8 degrees field of view), while the input device they used depended on the interaction technique. For the Voodoo Dolls technique users wore FakeSpace PinchGloves, while for the HOMER technique users held a three button joystick. The system tracked the HMD and input devices using an Ascension SpacePad. All of the worlds ran at a minimum of 30 frames per second.

3.3.2.2 Performance measures

For each interaction technique I measured the position error and the orientation error for each placement task. I defined the position error as the distance between the target's insertion point and its copy's insertion point. To measure the orientation error I calculated the axis of rotation and amount of rotation between the target and copy orientations, and used the amount of rotation as the orientation error. While this does not allow me to draw any conclusions about the direction of orientation error, it does allow me to compare the magnitude of the error between tasks using a single value.

3.3.3 Results

I conducted a multivariate repeated measures analysis of variance on the results with the interaction technique as the between-subjects variable and the five task groups as the within-subject variables. Overall Voodoo Dolls users manipulated objects more accurately than HOMER users for both distance ($F_{1, 10} = 17.953$, $p < 0.0025$) and orientation ($F_{1,10} = 35.260$, $p < 0.001$).

For the task groups, the difference in accuracy for Voodoo Dolls and HOMER users was not statistically significant for either distance or orientation for the indoor, medium group. For the indoor, small group Voodoo Dolls users were significantly more accurate for both distance ($F_{1, 10} = 21.347$, $p < 0.001$) and orientation ($F_{1, 10} = 6.236$, $p < 0.05$). These results reinforce the importance of feedback: the techniques were comparable for medium-size, nearby objects, but Voodoo Dolls users were more accurate when placing smaller objects at a distance.

Voodoo Dolls users were also significantly more accurate at the small, outdoor tasks for both distance ($F_{1, 10} = 7.689$, $p < 0.025$) and orientation ($F_{1, 10} = 62.874$, $p < 0.001$). Similarly for the large, outdoor tasks Voodoo Dolls users were significantly more accurate for both distance ($F_{1, 10} = 38.235$, $p < 0.001$) and orientation ($F_{1, 10} = 20.183$, $p < 0.001$). However, while Voodoo Dolls users were significantly more accurate at the medium, outdoor tasks for distance ($F_{1, 10} = 8.912$, $p < 0.025$), the difference in orientation accuracy was not statistically significant ($F_{1, 10} = 3.693$, $p = 0.084$).

Quantifying exactly how much more accurate Voodoo Dolls users were than HOMER users is difficult because the accuracy difference appears to depend on how large the manipulated object was and how far users had to place it. However, I can quantify the accuracy difference in general by comparing the average accuracy, across all the study tasks, for both techniques. On average Voodoo Dolls users positioned objects 88.28% more accurately (0.207 m off vs. 1.767 m off) and oriented objects 72.86% more accurately (5.149 degrees off vs. 18.968 degrees off).

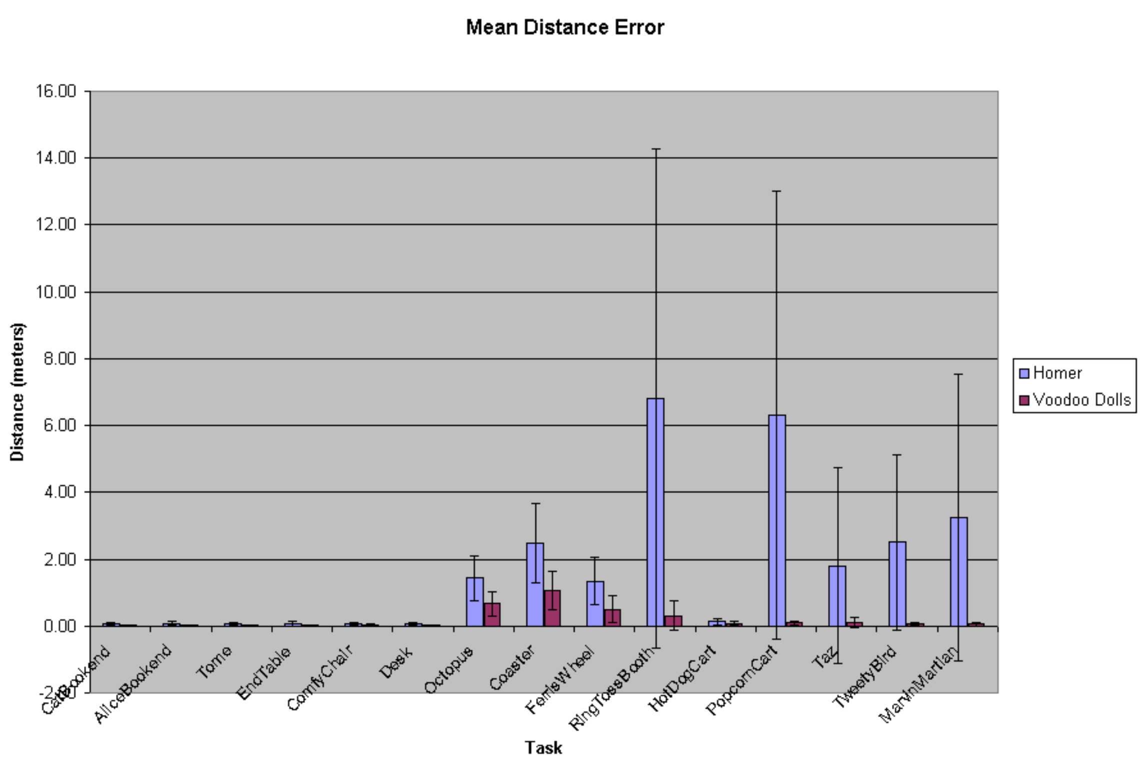

FIGURE 3-14. Mean distance error for each placement task for the HOMER and Voodoo Dolls techniques.

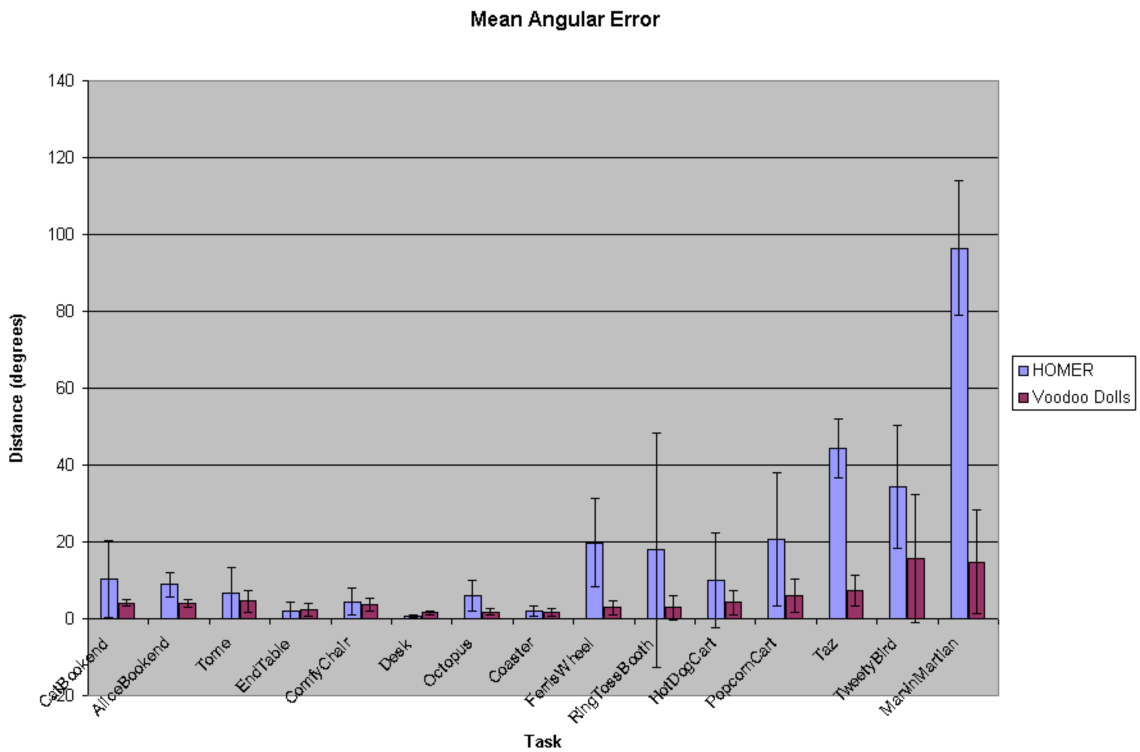

FIGURE 3-15. Mean angle error for each placement task for the HOMER and Voodoo Dolls techniques.

3.3.4 Discussion

While the HOMER technique should, in theory, allow users to manipulate objects more accurately than the Voodoo Dolls technique, the Voodoo Dolls technique in practice allowed users to position objects more accurately in terms of both distance and orientation. This suggests that providing accurate manipulation alone is not enough. In particular, techniques need to provide sufficient feedback in addition to sufficient accuracy so that users can verify and adjust while manipulating objects.

The hypothesis that feedback is the key factor in the difference between the techniques is reinforced by the fact that the standard deviation for tasks performed with the HOMER technique is almost zero for placement tasks near the user but is much larger for placements tasks involving small or distant objects. The standard deviation for tasks performed with the Voodoo Dolls technique is fairly consistent across tasks. This difference may be because placements with HOMER involved more of a guess as to whether or not the placement was accurate.

Overall the measurements of distance and orientation error also support this hypothesis, but there are two cases that deserve closer examination. First, HOMER users were much more accurate placing the Hot Dog Cart than they were placing the Popcorn Cart or the Ring Toss Booth (all medium, outdoor tasks). The difference is most likely due to the fact that for the Hot Dog Cart task users had to move the cart from a far position to a near position. Thus HOMER users had better feedback for this task than for the other two outdoor, medium tasks.

The second case concerns the placement of the small, outdoor objects. HOMER users were more accurate when positioning the small, outdoor objects than the medium, outdoor objects. This result might appear to contradict the importance of feedback, but in fact HOMER users had better feedback when manipulating the small, outdoor objects than when manipulating the medium, outdoor objects. Most HOMER users adopted a “silk cursor” [Zhai] strategy when placing objects: they would move the object away until the translucent copy appeared in front of it, and would then pull the object closer until it once again appeared in front. When placing the small, outdoor objects users could use the booth they were placing the objects on for additional feedback: if the object disappeared behind the booth, it was too far away.

Users comments about the techniques also supported the importance of feedback. Five out of six HOMER users mentioned the lack of feedback as one of the “hardest three things about using the technique”. By contrast, four out of six Voodoo Dolls users mentioned feedback as one of the “easiest three things about using the technique”.

The other most frequently mentioned “hardest three things” about the HOMER technique were the difficulty selecting distant objects (five of six users) and the time required to reel objects in or out (five of six users). The most frequently mentioned “easiest three things” were selecting large, nearby objects (five of six users) and moving objects at a constant distance (three of six users).

The most frequently mentioned “hardest three things” about the Voodoo Dolls technique were selecting the correct context for distant objects (four of six users) and rotating the context in the non-dominant hand (three of six users). The other most frequently mentioned “easiest three things” were fast, rough placement (four of six users), selecting objects in general (three of six users), and selecting small objects by retrieving them from the context (three of six users).

I also made a number of qualitative observations during the user study. While I had been concerned about the learnability of Voodoo Dolls, no users had any trouble learning either the HOMER or Voodoo Dolls techniques. Indeed, I had initially provided 15 practice tasks, but during an initial pilot study discovered that users picked up the assigned technique after performing one or two placements. As a result, in the study reported here I shortened the number of practice tasks to 5.

The primary difficulty I observed with the Voodoo Dolls technique was that some users seemed to expect that looking very closely at one corner or side of a doll and manipulating it would make the opposite corner or side act as the doll’s pivot. In other words, they wanted to make a minor adjustment to that corner or side without changing the doll’s position at the other end. In practice this did not work; the point where the user grasps acts as the pivot, so the user often completed their fine manipulation and looked at the rest of the doll to discover that they had messed up the placement on the other side. While these users were still able to make accurate placements, this does suggest a possible improvement to the Voodoo Dolls technique. If the user moves his head so that he can only see one corner or side of an object, the system could actually make the opposite side or corner act as the pivot. This would allow users to make incremental adjustments to a doll’s position (alternating between sides) that could slowly converge to an exact placement.

Another difficulty I observed with the Voodoo Dolls technique is that on two occasions users temporarily got confused as to which of the objects in their view were the dolls, and which were the original objects. Simply waving their hands did not necessarily help, because the original object would usually move as well. Both times users were able to quickly overcome the problem by rotating their body so that only the hand-held dolls were in view and then rotating back. Presenting the worlds in stereo might eliminate this problem, or the system could make dolls visually distinct from the original objects (e.g. black and white, or less saturated colors).

I saw no evidence that the Voodoo Dolls technique caused more fatigue than the HOMER technique. While the Voodoo Dolls technique requires users to raise their arms when creating dolls, it also allows them to move their arms to a comfortable working area after creating dolls. The HOMER technique, by contrast, does allow users to select objects from a more comfortable position (assuming users do not need to sight down the ray to select an object), but may require users to work with their arms extended in front of them in order to correctly position an object. In the post-survey questionnaires, user self-reports of fatigue on a scale from 1 (no fatigue) to 4 (extreme fatigue) were actually slightly higher for the HOMER technique than for the Voodoo Dolls technique (means of 2.5 and 2.0 respectively). Note that this result does not contradict Bowman’s work [2] suggesting that image plane selection may cause more arm fatigue than ray-casting. Image plane selection may indeed cause more arm fatigue than ray-casting, but because users spent more time manipulating objects than selecting them the overall reports of arm fatigue were not affected.

3.4 Enhancements to Voodoo Dolls

The previous discussion of the Voodoo Dolls technique describes the simplest version of that technique, the version I used in the formal user study. In the course of this thesis I also developed a number of optional enhancements to the technique. Although I discuss these enhancements in the context of the Voodoo Dolls technique, some of them are also useful in combination with other interaction techniques.

3.4.1 Moding By Image Plane Position

Interaction technique developers often worry about overloading users with commands to activate different techniques. For example, a user wearing PinchGloves might create dolls with by pinching his index finger and thumb, image plane navigate by pinching his middle finger and thumb, and pull themselves through space by pinching his ring finger and thumb. Instead of forcing users to remember which pinch activates a particular command, I can instead overload a single pinch so that the activated command depends on what the user is trying to do. For example, while developing the Voodoo Dolls technique I observed that users position their hands differently when image plane selection than when pulling themselves through space. Specifically, users about to image plane select an object tend to position their hands toward the center of their image plane, while users pulling themselves through space tend to position their hands at the edges of the image plane or out of view entirely. Considering this fact I can implicitly mode the user's pinch by where the user's hand lies on the image plane. If the user's hand is in the middle of his image plane the system executes an image plane select, otherwise the user's hand grabs the world and he can pull himself around. I can communicate where the regions are on the user's image plane by lightly coloring the different regions, or I could outline those regions on the overlay plane.

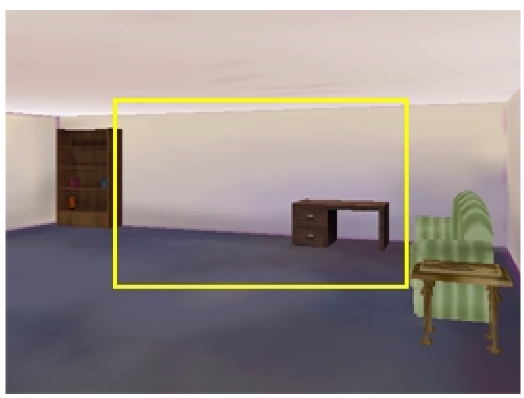

FIGURE 3-16. If the user pinches his fingers while his hand appears within the outline on his image plane he will create a doll; otherwise he will grab space and can pull himself around.

3.4.2 Improving Selection Feedback

When a user is employing the Voodoo Dolls technique to manipulate objects there is a cost associated with creating a doll for the wrong object when the user's non-dominant hand is already holding an object: the user has to undo the creation by disposing of the doll over his shoulder or he will have accidentally moved that object. I can try to reduce this cost by improving selection feedback to reduce the likelihood that user's will select the wrong object. One of the ways I can do this is to create a transparent doll in the user's hand to show what doll the user will create if he performs a selection. If the user wants to create a doll for that object he pinches his fingers together and the doll turns opaque.

If the transparent doll is not the one the user wants, he adjusts the position of his hand to move the crosshair on his image plane, and the system updates the transparent doll to reflect the new potential selection target.

Ideally the system would only provide the transparent doll for feedback when the user is actually trying to create a doll. Instead of assuming that the user is constantly about to select an object or forcing the user to issue an “about to select” command I can take advantage of the previous enhancement and assume that the user only wants to create a doll when the user’s hand is within the middle region of his image plane. This approach also gives the user additional feedback as to what will happen (creating a doll vs. grabbing space) when he pinches his fingers.

FIGURE 3-17. When the user positions the crosshair over an object the system provides feedback by creating a transparent doll for the object. The doll turns opaque when the user pinches his fingers together.

I can extend this idea to communicate to the user the context that the system will provide for different reference objects. When the user is preparing to create a doll with his non-dominant hand the system can create transparent context dolls around the transparent candidate doll in the user’s hand.

3.4.3 Zooming for Selection

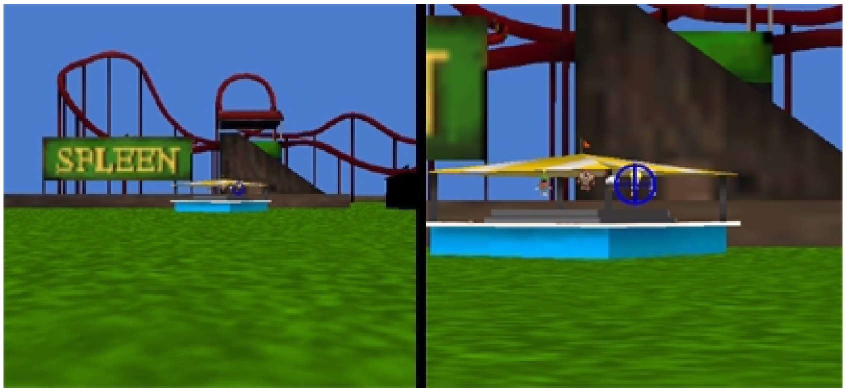

FIGURE 3-18. If the user is having trouble positioning the crosshair over a distant object, he can lean forward to zoom in around the crosshair.

Even with feedback communicating what doll the user will create with a pinch, the user may still have trouble creating dolls for objects that occlude a very small visual angle. Although the user might be able to pull a doll for this object out of a context around his non-dominant hand, in cases where the user is already holding a doll in his non-dominant hand this could entail the user dropping that doll, creating a new reference doll, dropping it, and re-creating the original reference doll. If possible I would prefer to save users this extra work. What I can do instead is provide a mechanism for user's to temporarily increase the visual angle occluded by a small number of objects. When the user is about to create a doll (which I can again detect by moding regions of the user's image plane) the user can lean toward his hand to zoom in on the objects that appear around his hand on his image plane. I detect a lean by looking for the user's head to move while his hand remains stationary. I create the zoom affect by narrowing the user's field of view; this has the effect of increasing the visual angle occluded by objects near the center of his field of view. I reset the user's field of view back to normal when the user either stops leaning forward or when he creates a doll.

3.4.4 Dominant Hand Doll Context

Providing a context for only the doll in the user's non-dominant hand requires the user to create dolls in a particular order to optimize efficiency. Consider the case of adjusting the position of a lamp on top of a table. If the user creates a doll for the table first with his non-dominant hand he can then just pull the doll for the table out of the context, while if he creates the doll for the lamp first he has to perform another image plane selection to create the doll for the table. I could always provide a context for both dolls, but the context around the doll in the user's dominant hand could occlude the user's view or clutter the space when he tries to position the doll around the reference doll in his non-dominant hand. A better approach might be to provide a context for the doll in the user's dominant hand only when the user's non-dominant hand is not current holding a doll. This would allow the user to create a doll for the lamp with his dominant hand and pull the doll for the table out of the doll's context with his non-dominant hand; as soon as he does so the system would then hide the context around the lamp doll.

3.4.5 Dropping Without Destroying

Currently dolls vanish as soon as users let go of them (unless, of course, the doll was in the user's dominant hand and is now part of the context around the reference doll still grasped in the user's non-dominant hand). There are situations where a user might want to temporarily drop a doll without it disappearing. For example, the user might want to adjust his grip in the doll to make it easier to subsequently place around a reference doll. If the previous enhancement that creates a context for the doll in the user's dominant hand is available, the user might want to switch to holding one of the dolls in the context without grabbing the doll with his non-dominant hand and passing it back to his dominant hand. To address these situations I can provide a mechanism for a user to temporarily drop dolls without destroying them by moding the drop action based on the orientation of the user's hand. If the user's hand is oriented palm-down the drop action destroys the doll. However, if the user's hand is oriented palm-up the doll (and its context, if it possesses one) remain floating in space until the user either picks up a doll again or moves his hand away from the dropped doll (beyond some pre-specified distance threshold).

3.5 Future work

I once commented in a discussion with colleagues about interaction techniques that we could turn any technique for manipulating objects into a technique for navigating the world. Users can fly into a WIM [Pausch95], pull themselves around a scaled world [SmartScene], or pull themselves toward a grasped object rather than pulling the grasped object to them [Pierce97][Mine][Bowman99b]. I believe that users could also employ a variant of the Voodoo Dolls technique for navigation. The naive implementation of this invariant might be similar to Pausch et al.'s implementation of flying into the WIM. The user could create a doll for himself and place it in the context around the doll in his non-dominant hand. When he lets go of the doll he flies into the context which grows up around him. However, flying into a WIM or context in this manner introduces a problem: creating the path between the user's current viewpoint and his doll's viewpoint, particularly when the user's new viewpoint is not parallel to the ground plane.

Rather than implementing the navigation variant this way, I suspect that a better approach might be to allow a user to grab the context with both hands and pull it around his head. As the context approaches the user's head it could both expand in both size and coverage, so that by the context reaches the user's head he appears to be in that new location.

This implementation would have two main advantages. First, the user would have more control over the transition to the new location, being able to both control the rate of the transition and fine-tune his final position. Second, this implementation would allow a temporary change in viewpoint. The user could hold the context around his head to get a view from that new position, but rather than letting go to put the place change into effect, the user could pull the context off his head to return to his previous location.

I also believe that other interaction techniques can take advantage of the idea of changing the properties of objects depending on the hand holding them. For example, an object could have control parts that are only visible when the user holds it in his left hand. This allows the user to move or examine the object with his right hand without these parts cluttering his view. If the user wants to manipulate these parts, he transfers the object to his left hand, the controls appear, and he can manipulate them with his right hand.

3.6 Summary

By breaking the assumptions that objects can only be in one place at a time, that objects are persistent, and that objects' behaviors are independent of the hand holding them I created the Voodoo Dolls technique. This technique allows a user to manipulate objects at a distance by creating dolls: miniature copies of objects whose properties depend on whether users hold them in their dominant or non-dominant hands. Dolls held in the non-dominant hand cause the object they represent to act as the frame of reference. Dolls held in the dominant hand control the position and orientation of the object they represent relative to this reference frame. In a formal experimental study comparing Voodoo Dolls with HOMER, an existing best practice technique, I determined that Voodoo Dolls users manipulated objects more accurately than HOMER users. On average, across all the experiment tasks, Voodoo Dolls users positioned objects 88.28% more accurately and oriented objects 72.86% more accurately. The Voodoo Dolls technique benefits by overcoming several shortcomings of existing techniques for manipulating objects at a distance.

The Voodoo Dolls technique overcomes the trade-off between range and accuracy or placement time. With this technique, accuracy placement depends on the user's choice of reference object. This allows users to place an object at any distance with great accuracy. In addition, the placement time does not depend on the distance to the object or the distance between the object's initial and final positions. Rather than a selection followed by a continuous transition between location or scale, the Voodoo Dolls technique employs two repeated selection operations that cause discontinuous transitions between the object's position and the placement accuracy.

The Voodoo Dolls technique allows user to manipulate both visible and occluded objects. The user can directly create a doll for any visible object. For objects that occlude a very small angle on his image plane, or for occluded objects, the user can create a reference doll for an appropriate reference object and grab the doll for the desired object out of the reference doll's context.

The Voodoo Dolls technique takes advantage of the asymmetric role human hands play when manipulating objects. In the real world the user's non-dominant hand defines the reference frame that his dominant hand works in. The Voodoo Dolls technique expands on this context by allowing the reference frame defined by the user's non-dominant hand correspond to the reference frame defined by any object in the 3D scene. To create this correspondence the user holds a doll for the particular object in his non-dominant hand.

The Voodoo Dolls technique allows users to always work relative to a stationary reference frame. Accurately placing objects can be difficult when the object the user is working relatively to is in motion, either because the object or the user is moving. The user can create a temporary stationary reference frame to work in by holding the doll for the moving object in his non-dominant hand. This allows the user to avoid the extra steps of pausing and re-starting the object's motion.

No matter where you go, there you are.
Buckaroo Bonzai

Out of mind as soon as out of sight.
Lord Brooke

CHAPTER 4 *Places and Landmarks*

Navigation is a fundamental part of our everyday lives. When a task we need to complete or an activity we desire to participate in is located somewhere else, we need to navigate from our current location to that other location. The scales at which we navigate in the real world can vary widely, from simply walking across a room to flying halfway around the world.

Navigation is playing an increasingly important role in virtual worlds. Thirty years ago, when researchers were just starting to implement virtual worlds, navigation was a simple task because users were limited to moving within an area of a few square feet [Sutherland]. Today virtual worlds are much larger and present more challenging navigation tasks. Visitors to ActiveWorlds [ActiveWorlds] navigate around a virtual world the size of a small city, while players of the game *Ultima IX: Ascension* [EA] must navigate about a world the size of a small country. With advances in processing power and 3D hardware, virtual worlds are poised to grow even larger; someday soon users may be able to step into a realization of Niven's Dream Park [Niven] or Stephenson's Multiverse [Stephenson].

The initial techniques researchers created for navigating virtual worlds resembled techniques for navigating the real world. The earliest technique was walking: as long as users remained within the range of the tracking system their motion in the real world was mapped directly to their motion in the virtual world [Sutherland]. The growth of virtual worlds quickly outstripped the range of tracking systems, so researchers turned to navigation techniques that did not require users to physically move about the real world.

The next common navigation technique was the "point to fly" technique. Conceptually users stood on a platform, or vehicle, in the virtual world [Pausch91]. They could walk about the platform by walking within the range of the tracking system in the real world. In addition, users could fly the platform around the scene by pointing in the desired direction of travel and pressing a button. This technique is perhaps still the most popular technique for navigation in virtual worlds, both for its ease of use and its ease of implementation. The primary drawback to this technique is that it does not scale well: adapting this technique for large virtual worlds introduces a trade-off between velocity and precision. In order to quickly traverse large distances users need to be able to travel at high velocity. However, the faster users move, the less precisely they can control their position.

To overcome this trade-off, designers started to abandon real world metaphors and create navigation techniques that relied on the unique characteristics of virtual environments. Two of these techniques allowed users to navigate to visible objects with a single gesture. With the first, image plane navigation [Pierce97], the user positioned the thumb and index finger of one hand to frame the target object in their view. When the user pinched his fingers together to select the object, the system calculated a scale factor by dividing the distance from the user to the target object by the distance from the user's hand to his eyes. When the user subsequently moved his hand, the system adjusted his position so that his viewpoint lay along the vector from the target object to his hand. The system determined the user's position along the vector by multiplying the distance from his eyes to his hand by the scaling factor. The user could thus move to an object in a single gesture by selecting the object and pulling his hand to his eyes.

The second technique, world scaling [Mine], achieved a similar effect by shrinking the world. When the user selected an object using an image plane technique the system instantaneously scaled the world down, moved it to position the selected object at the user's hand, and then attached the world to the user's hand. The user achieved the desired position by moving the scaled world to the correct location and then letting go of it to scale it back up.

These two techniques, image plane navigation and world scaling, allow users to navigate relative to any visible object with a single gesture. Unfortunately, these techniques also do not scale for very large virtual worlds. Both techniques limit users, roughly speaking, to moving with a single gesture at most twice the distance to the farthest visible object in any given direction. Users cannot navigate using any object that is occluded, too small to see, or removed from the scene by the far clipping plane.

I classify the navigation techniques that I have discussed so far as *continuous navigation techniques* because when using these techniques to travel along a path from point A to point B the user's position at any time t can be viewed as the output of some continuous function of t. This interpretation ignores the fact that in current implementations of these techniques (and 3D graphics in general) users technically travel along a discrete approximation of this idealized path, but this facet of the implementation of these techniques is irrelevant for this discussion.

Continuous navigation techniques typically do not scale well because they require the user to both specify and traverse a path between the user's current location and the target location. For a target location that is not immediately visible from the user's location, the user must know where the target location is relative to his current location, and he must plan a path to reach that location. The greater the distance between the locations, the more complex and involved the planned path may be. Instead of navigating directly to the desired destination the user will most likely need to navigate to a series of intermediate locations. Even if users can plan a simple, straight path to the destination, the time required to traverse the path grows at best linearly with the path's length.

Instead of specifying a path between locations, users of *discontinuous navigation techniques* instead focus on specifying the desired destination itself. A discontinuous navigation technique is any navigation technique where the user's path between two locations can only be described by a discontinuous function of time. These navigation techniques employ teleportation: the user instantaneously disappears from one location and appears in another.

Some researchers discount teleportation as a viable navigation technique because an instantaneous transition between locations can disorient users. However, in the only formal study of teleportation in virtual environments [Bowman99a] users did not know the target location in advance, so the fact that they were disoriented by teleportation is not surprising. When users specify the target location themselves they may experience less disorientation. Designers may also be able to reduce the amount of disorientation by providing a perceptually smoother transition between locations. For example, flying into a handheld world-in-miniature (WIM) [Pausch95] provides a smooth, yet still discontinuous, transition between two locations.

A priori analysis suggests that discontinuous navigation techniques would be better suited to navigation in large virtual worlds. Certainly this is true once the user has specified the target location, because the user can travel to any specified location, no matter how distant, in constant time. However, these navigation techniques exchange one problem, how to quickly traverse large distances, for another: how to quickly specify the target location.

One approach, often adopted in desktop virtual worlds, involves specifying the target location by choosing the name of the location or a worldlet [Elvins] representing it from a list. While simple, this approach does not scale very well. The larger the virtual world, the longer the list of potential destinations will be and the more time the user will have to spend looking through it to find the desired destination. In addition, a list of destinations will never provide complete coverage of the virtual world. Users can employ a discontinuous navigation technique to reach the listed destination nearest to the target location, but will then need to employ a continuous navigation technique to reach the actual target location.

Fixed portals between locations [Barrus] suffer from a similar problem. The user can instantaneously traverse the distance between the two portals, but must still employ a continuous navigation technique to reach the near portal and to reach the target location from the far portal.

The best existing approaches to specifying the target location involve a handheld widget, either a map or WIM of the world. The user either points to the target location on a map to teleport there [Angus] or moves a doll representing himself to the target location to cause the system to fly him into the WIM [Pausch]. While these approaches theoretically allow the user to reach any location depicted in the widget, they suffer from a resolution problem. The larger the world depicted in the widget, the less accurate the user will be when specifying the target location. For very large virtual worlds, an error of a few millimeters (the error associated with most tracking systems) could result in the user appearing kilometers away from the desired target location.

Another drawback to handheld widgets is that users have to focus their attention on the handheld widget. This can be problematic if users need to pay continuous attention to the world around them. For example, users might be engaged in a virtual combat training scenario and need to keep their eyes out for enemies.

Researchers have devised two methods to avoid the resolution problem raised by presenting users with a miniature replica of the entire world. The first is to present a fisheye representation [Furnas86] of the world consisting of a focus and a context. A fisheye representation reduces the focus scaling factor by allocating a disproportionate amount of the widget's space to it. Reducing the scaling factor allows users to act with more precision within the focus. However, fisheye representations present other problems. First, users must be able to change the focus. Setting the focus directly by pointing to the context (the area outside the focus) is problematic. In order to provide more area for the focus, a fisheye representation typically crams the context around the edges of the widget. With a fisheye representation that allocates half of the widget to the context, users will have roughly half the resolution available when pointing to the desired focus than they would if they were pointing to a uniformly scaled map or WIM of the same size. While users could instead drag the focus at some velocity, moving the focus to a distant location could take a significant amount of time. The second problem is the increased difficulty in finding locations in the fisheye representation. The spatial location of a particular place in the widget can change as users move the focus, preventing users from applying spatial memory to find a place (e.g. a town will not always necessarily be in the upper left of the map or WIM).

The other alternative is to provide users with a mechanism for panning and zooming the section of world visible in the widget. To move to a new location users typically zoom out until the location is visible, pan the displayed section until the widget is centered on that location, and then zoom in until they have enough resolution to accurately specify the exact desired destination. Although the ability to zoom in arbitrarily far solves the resolution problem, this alternative introduces the possibility that users may spend a significant amount of time panning and zooming the widget before achieving sufficient resolution to accurately teleport. This method thus suffers essentially the same drawbacks as continuous navigation techniques. Users can pan and zoom faster but less accurately, or more accurately but more slowly. In fact, there is a continuous navigation technique that corresponds to this method. Instead of manipulating a widget, users scale the world down, pull themselves around the compressed world, and then scale it back up [Robinett] [SmartScene]. The primary differences between the two techniques is that when scaling the world users do not have to divide their attention between the world and a handheld widget, but users navigating with a handheld WIM can inspect a distant location without moving from their initial position.

Both of these methods suffer from the problem that users may be unable to locate a destination because they do not recognize it. With a fisheye widget details in the context may be distorted, or they may be too small to distinguish. Similarly, with a panning and zooming widget details may become arbitrarily small when users zoom out.

Neither fisheye widgets nor panning and zooming widgets scale particularly well. While both approaches do address the resolution problem raised by the need to accurately specify a destination in a miniature representation of a very large world, their drawbacks become more significant as virtual worlds get larger. For fisheye widgets larger worlds either reduce the available accuracy for directly specifying the focus or increase the time required to drag the focus. For panning and zooming widgets larger worlds increase the time required to pan and zoom.

4.1 Toward a better technique

My primary goal in creating a new navigation technique was to draw on the advantages of both continuous and discontinuous navigation techniques while avoiding their disadvantages. From continuous navigation techniques, particularly image plane navigation, I wanted to draw on the ability to reach visible locations with a single gesture. From discontinuous navigation techniques I wanted to draw on the ability to travel to any location in constant time. In both cases I wanted to avoid the need to specify a continuous path to the destination, either in the world or in a handheld widget. In short, my primary goal was to create a technique that allowed users to quickly and accurately reach any destination in the virtual world with only a few, simple gestures.

I drew on a number of other subsidiary goals to shape my design:

- I wanted the new technique to help keep users spatially oriented in the virtual world. People are oriented when they know where they are in the environment and how to reach other places in the environment [Passini]. While this goal includes helping users maintain their orientation during continuous travel about the virtual world, I particularly wanted the new navigation technique to help users either maintain their orientation during or recover their orientation after a discontinuous transition because of the risk of disorientation.
- I wanted the new technique to help users develop a cognitive map of the virtual world. A cognitive map helps users maintain their orientation, locate other places on maps (and by extension WIMs), and plan paths between locations [Downs77][Lynch]. In addition, cognitive maps can facilitate exploration by helping users determine what locations they have and have not visited in the virtual world.
- In addition to facilitating exploration by helping users develop a cognitive map, I wanted the new navigation technique to facilitate exploration by making it easier for users to visit areas of the world containing interesting sights and activities while avoiding the empty or uninteresting sections of the virtual world. I believe that navigation techniques should help users determine what there is to do and where they can do it, and then help them reach those locations.
- I wanted the new technique to provide a smooth transition from novice to expert behavior both for use of the navigation technique and for knowledge of the virtual world. In other words, I wanted the technique to be easy for novices to use, to make it easy for novices to find their way around an unfamiliar virtual world, and to help novices learn the world while using the technique. I also wanted the technique to make it easy for experts familiar with both the technique and the world to rapidly reach any location without interference from any scaffolding for novices.
- I wanted the navigation technique to help shape the user's experience. For some virtual experiences (e.g. [DisneyQuest]) designers want to make users believe that they are within an extensive, vividly detailed virtual world. However, actually building an extensive, vividly detailed virtual world requires a significant amount of work, and is not usually worth the effort for sections of the world that exist just in case users try to visit them. I wanted to develop a new navigation technique that made it easier for designers to present users with the illusion of a large, compelling virtual world without actually expending large amounts of effort to completely develop every detail of the world.

To create a navigation technique that met these goals I broke the assumptions that distant objects are occluded by nearby objects, that objects get smaller and smaller as they get farther away until they are too small to see, that all objects get smaller at the same rate, that objects' appearances remain constant (except for size) at all distances, and that the abstract concept of place is not represented by a concrete object.

4.1.1 Notes on teleportation and verbal specification of a destination

At this point I need to discuss a discontinuous navigation technique that I did not previously mention: navigation by teleportation to a verbally specified location. With this technique the user tells the system where to place him (e.g. "teleport me to the center of my office") and the system moves him there instantaneously. I believe that navigation by teleportation to a verbally specified location will beat any other navigation technique for fast travel to a well known destination because the time required to reach any destination is limited only by how fast the user can say the destination. Other researchers have expressed similar beliefs. Herot, for example, expressed the belief that manipulating a spatial representation works best when users cannot easily specify a precise query for an object or location [Herot].

Despite its speed in ideal conditions, this navigation technique does not meet all the goals that I outlined above. The technique's primary weakness is the need to verbally specify the destination. When users do not yet know the names for locations in the world, for example when exploring a new virtual world, the technique is essentially useless. The technique also forces users to rely on pure, unassisted recall when traveling to another location. If the user cannot remember the name of the location, the user cannot travel there.

I therefore viewed navigation via teleportation to a verbally specified destination as complementary to the technique I wanted to create. My technique should help users explore and discover the interesting locations in the world, and should provide a fallback for users when they want to travel to a location but do not know or cannot remember its name. My technique should also provide a smooth transition to the expert behavior of teleporting to a verbally specified location by helping users discover and learn the names of locations in the world. Borrowing an analogy from desktop computing, teleportation by verbal specification is like a command line, while my technique should be like a GUI. In the next section I will describe the technique I created.

4.2 Navigation using Places and Landmarks

Downs and Stea [Downs77] proposed that to complete a navigation task in the real world users must first be oriented in the environment. Once they are oriented they must choose and correctly follow a route. Users successfully complete a navigation task when they discover the objective and move directly to it. Researchers had already created techniques for moving directly to visible objects (image plane and world scaling navigation), so if I could help users maintain their orientation and increase the range at which they could discover their objective I could help them navigate more efficiently.

4.2.1 Visible Landmarks

In the real world urban planners help people discover their objective by providing landmarks. A landmark is an object that serves as a point of orientation or frame of reference for locating other objects [Lynch]. When looking for a particular place, people often locate the nearest landmark and navigate relative to it. The ability to see a nearby landmark in essence increases the distance at which users can detect their objective, even if they cannot see it directly. Landmarks also help users structure their cognitive maps [Chase] and determine where they are in the world by providing a link between their cognitive maps and the perceived environment [Downs77]. When landmarks are visible, people tend to rely on them for orientation rather than on geographic orientation (North, South, East, West) [Chen].

In the real world the utility of landmarks is limited because at greater distances other objects occlude them or they get too small to see. I overcame these limitations by breaking the assumptions that distant objects are occluded by nearby objects, that objects get smaller and smaller as they get farther away until they are too small to see, and that all objects get smaller at the same rate to create *visible landmarks*. Visible landmarks are landmarks that always remain visible to the user.

In order to implement visible landmarks I created a method of making distant objects visible. Distant objects can be impossible to see because they are occluded by closer objects, because they lie beyond the far clipping plane, or because they are so far away that they are too small to see. I allow users to see an occluded or clipped object by placing a scaled copy of the object along the vector from the user's viewpoint to the original object. I position the copy a meter away along this vector to prevent the copy from interfering with the user's view of objects in his immediate work area while still occluding more distant objects. I calculate the initial size for the copy using the distance to the copy and the ratio between the distance to and size of the original object. By subsequently updating the position and size of a copy as the user moves around, I can provide users with the illusion of x-ray vision that sees through nearby objects to the original object. Because the scaled copy could still be too small to see, I impose a minimum visual size on the copy. As users move away from the original object it will get visibly smaller and smaller until it reaches that minimum size, at which point its visible size will remain constant until they move closer again. I also display a translucent "halo" around the visible landmarks to help them stand out against the background in crowded environments.

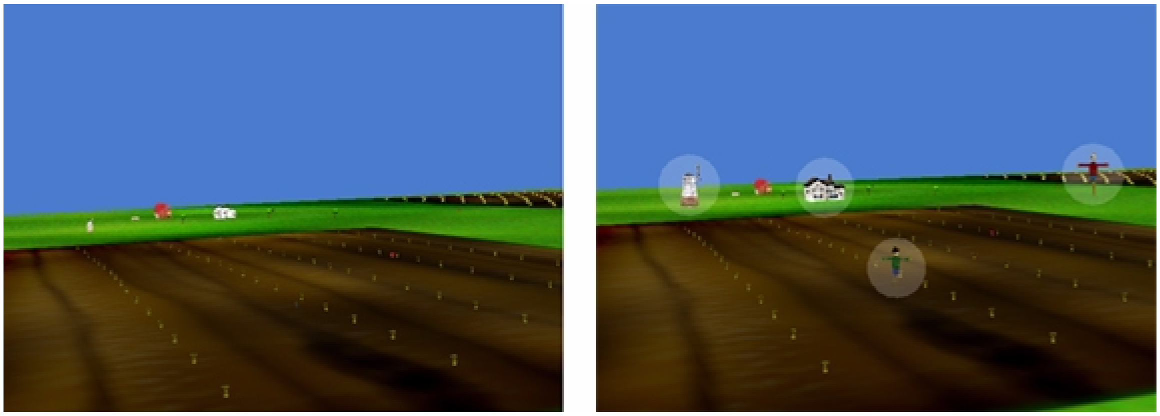

FIGURE 4-1. The farm's visible landmarks (right) make it easier to locate the scarecrows in the fields, which would normally be hard to see (right).

FIGURE 4-2. The city's visible landmarks (right) help users locate landmarks that would normally be obscured by other buildings (left).

Additional visual enhancements to visible landmarks can provide users with additional information, but designers need to be careful to avoid crowding the user's view or occluding the landmarks themselves. For example, overlaying visible landmarks with textual labels displaying their names would facilitate teleportation to a named destination by helping users learn landmark names. I chose not to add this enhancements in the implementation I describe in this chapter because the users would not be naming destinations to teleport, so the names would just clutter up the view.

Making a world's landmarks visible also requires determining which objects in the world are landmarks. In the real world people typically choose landmarks in the real world based on their size, visual prominence, and distinctiveness [Lynch]. In a virtual world designers, users, or the system itself can make objects landmarks. When they create a virtual world from scratch designers an add objects that they intend to be landmarks. At run time users of a virtual world should be able to make objects that they want to be able to easily find (e.g. their home, or a favorite location) landmarks. The system should also be able to temporarily or permanently make objects landmarks in order to provide shortcuts through the space. For example, the system could temporarily make an object in a location that a user recently left a landmark so that the user can easily return to that location. The system could also make an object in each of the locations that the user visits frequently a landmark.

The idea of adding landmarks to a virtual world is not new, but to date these landmarks have always been passive aids intended merely to help users organize a cognitive map and recognize particular locations in the world [Vinson] [Chalmers][Darken93]. Visible landmarks go beyond these passive landmarks in two ways. First, they actively aid the user: the system positions and scales copies of them to ensure that they remain visible. By remaining visible they allow users to always determine where they are in the world by looking about them and locating particular landmarks.

Second, visible landmarks facilitate navigation to distant locations. When used in combination with image plane navigation visible landmarks allow users to reach distant locations with a single gesture. When a user image plane selects a visible landmark the system in essence passes the selection to the actual landmark instead of its visible copy, and then user can then image plane navigate directly to or relative to that landmark.

Users can also choose to image plane select a landmark, and instead of image plane navigating to it pull away a WIM with a pre-determined scale focused on that landmark. The pre-determined focus and scale of this WIM makes it easy for users to locate a position near that landmark. The WIM shows only the area around that landmark, and should thus provide sufficient resolution to allow users to accurately specify their desired location without adjusting the WIM's focus or scale. Instead of pointing to teleport or position a doll of themselves to fly into the WIM, users instead position the WIM to achieve the viewpoint they desire. When they let go of the WIM, the system teleports them to the corresponding position in the virtual world. Users can remain in their current location without teleporting by instead holding the WIM down by their waist before releasing it.

4.2.2 Places and the Place Hierarchy

By themselves visible landmarks do not scale very well: in very large worlds the sheer number of visible landmarks could visually overwhelm users. I needed a way to reduce the number of visible landmarks while still allowing users to detect distant destination and travel to them with a few gestures. Because real world motion exhibits locality I did not want to remove any visible landmarks near the user; that meant displaying fewer distant visible landmarks. I considered culling landmarks more and more aggressively with increasing distance from the user, but that would have reduced the set of detectable destinations. I needed a different approach.

I settled on displaying a smaller number of semantically higher level objects. Just as people do in the real world, I organize the virtual world into semantic units. Lynch used the generic term *district* to refer to these semantic units [Lynch], but in this thesis I will use the more common term *place*. Instead of displaying distant visible landmarks, I display representations of the places that contain them. These *place representations* are akin to visible landmarks: they facilitate both orientation and travel.

Organizing a virtual world into a set of places is still not enough. Because each place will typically contain only a handful of landmarks, displaying place representations instead of distant visible landmarks only achieves a linear reduction in the amount of visual clutter. I reduce the number of visible objects even by organizing the virtual world not into a set of places, but into a *place hierarchy.* I then choose which objects to show based on the hierarchy and the user's current position.

To aid the user's navigation within the place where he is currently located, I display all the visible landmarks in that place. To aid the user in navigating to a different place, I display a selection of places at different levels in the hierarchy. Whether a particular place in the hierarchy is visible depends on the place's distance from the user: places low in the hierarchy near the user are immediately visible, while places farther and farther from the user are represented by higher and higher places in the hierarchy (see Figure 4-3). This approach achieves a logarithmic reduction in the amount of visual clutter and results in a view similar to that in "View of the World from 9th Avenue" by Saul Steinberg (Figure 4-4), more popularly referred to as the "New Yorker's view of the world". The piece shows the user's current location in a great deal of detail while presenting more distant locations as higher and higher level semantic places.

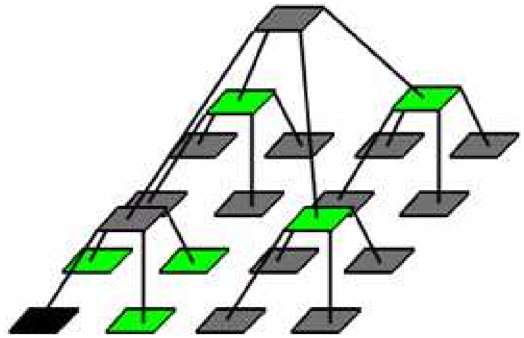

FIGURE 4-3. A simple example place hierarchy. The black node represents the place the user is in. The green nodes represent displayed places.

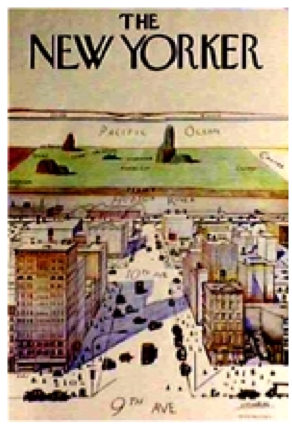

FIGURE 4-4. Saul Steinberg's "View of the World from 9th Avenue" from the March 26, 1979 cover of The New Yorker (© The New Yorker).

The idea of displaying places near the user that are low in the hierarchy and aggregating places more and more as they get farther away is supported by research that suggests that people group areas that are farther away but distinguish between areas in the surrounding region [Dornic][Gould]. For example, people who live in Connecticut tend to refer to the Deep South as a single region, but rather than referring to the Northeast they refer individually to Maine, New Hampshire, Vermont, etc.

Previous research also supports the practice of only displaying the landmarks for the place the user is currently in, because people appear to only encode spatial relationships between two locations if those locations appear in the same semantic unit [Steves]. Otherwise people infer the spatial relationship by combining the relationships between and within semantic units. This allows people to balance the trade-off between memory and computation, but can lead to errors. For example, most people would state that Montreal is farther north than Seattle, but in fact the reverse is true. People are lead astray by the known fact that Canada is farther north than the United States.

I chose a semantic hierarchy for two reasons. First, previous research suggests that people naturally organize their cognitive map of the world into a semantic hierarchy [Chase][Lynch][Stevens]. In our everyday lives we organize the world into neighborhoods, towns, cities, counties, states, countries, planets, solar systems, etc. Second, I could only show users a small number of objects without visually overwhelming them or cluttering their view, so I wanted those objects to be as meaningful as possible. Dividing the world using quadtrees or oct-trees, as existing systems have done (e.g. [Koller95][Maciel]), would have resulted in places without any meaning apart from their existence (e.g. grid square 23 instead of the East Coast).

The best time to create the semantic place hierarchy is before actually building the world. The world's designers can then organize the virtual world to reflect and reinforce the hierarchy. Designers may be able to study the structure of an existing world and create a hierarchy for it, but if the original creators of the virtual world did not incorporate structure into the world the designers may be reduced to imposing an arbitrary hierarchy on it. Because over time users may create their own semantic places (e.g. the Northeast, the Deep South, the Midwest), designers may also need to consult with users when creating a hierarchy for an existing virtual world. While it may be possible to generate a place hierarchy automatically instead of manually for an existing virtual world, I will defer that discussion until the future work section of this chapter.

4.2.3 Place Representations

My next design decision was how to display places to users. I wanted users to be able to determine the location of places relative to their current position, and to be able to distinguish the important features of a place. I settled on using miniature copies of the places to represent them. To provide users with a good bird's eye view of the place representations while keeping them out of the users' way I position the representations at waist level. To communicate the direction to each place I keep its representation positioned along the vector from the user to the center of the place as the user moves about the world.

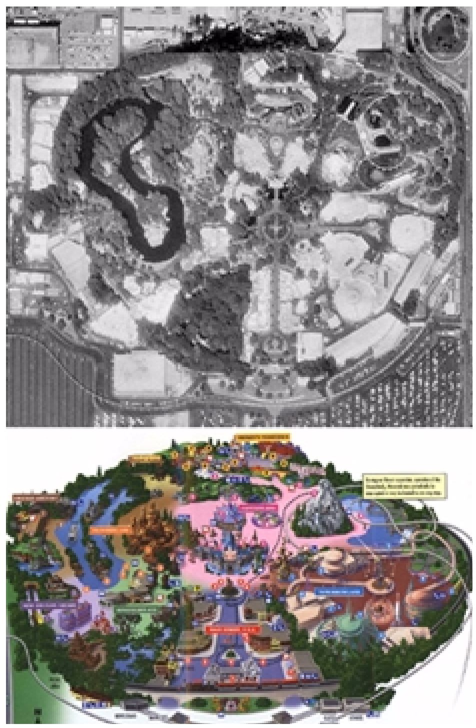

FIGURE 4-5. The "artist's conception" map of Disneyland (© Disney) is easier to read than the satellite photo of Disneyland.

Although I wanted to display the places using miniature representations, representing the places as traditional WIMs (exact miniature replicas of the places) was problematic. For larger places I would need to either compress the details in the representations so much that users would not be able to distinguish them, or I would need to enlarge the representations so much that they would block too much of the rest of the world. I solved this problem by creating a second type of place representation. I represent the places at the lowest level of the place hierarchy, which tended to be fairly small, using traditional WIMs or *actual representations*. However, I represent places higher in the hierarchy using *symbolic representations*.

Representing larger places symbolically gives designers the freedom to present important details (e.g. landmarks, available activities) without worrying about maintaining the correct scale, aspect ratio, etc. These symbolic representations are essentially "artist's conception" representations that focus on effective communication instead of strict accuracy. A side benefit of symbolic representations is that they typically have a reduced rendering cost compared to actual representations. Instead of showing all the objects and details in a place, symbolic representations provide a sketch of a place and only contain details for the important objects.

To help users navigate effectively symbolic representations must help users detect destinations contained within the places they represent. In theoretical terms, they must provide residue [Furnas97] or information scent [Pirolli] for those destinations. I provide this residue for each place by choosing the most important landmark in its representation and displaying a scaled copy of the landmark in its parent's symbolic representation (see Figure 4-6). For the most important landmark I try to select a landmark that is easily recognizable and offers a shorthand method of characterizing the place. This approach mirrors the way people often use a single landmark to represent an entire place in the real world [Downs77]; Figure 4-7 contains some real world examples.

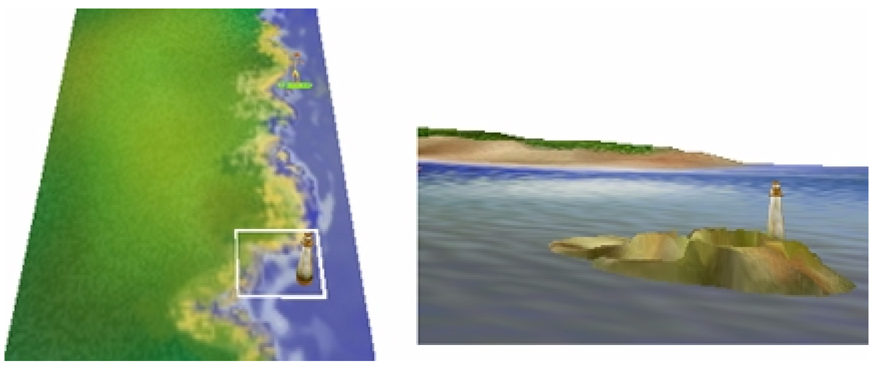

FIGURE 4-6. The lighthouse in the symbolic representation (left) of the Coast provides scent for and helps users locate the Beach (right).

FIGURE 4-7. Prominent landmarks that people often use as a shorthand to represent actual places.

I chose to represent a place as a single object at the next higher level based in part on one of Lynch's observations. He observed that when people organize places into a hierarchy, places at a particular level in the hierarchy become points or nodes in the next higher level [Lynch]. For example, the downtown area of a city and its suburbs would be places at one level, but would be points at a higher level where states were places. While I could also use a small number of objects to represent a place, in practice this can quickly lead to the symbolic representation appearing crowded.

My approach results in landmarks at the lowest level of the hierarchy also functioning as landmarks at the highest level. This approach is contrary to the advice of some researchers, who suggest using different landmarks at different levels [Vinson]. Creating new landmarks for each level, however, can be too time consuming and forces users to learn a larger number of landmarks. More importantly, propagating landmarks all the way up the hierarchy makes it easier for users to detect destinations. Using low level landmarks as scent in higher level symbolic place representations allows users to locate a place by visually recognizing it instead of consciously recalling its location relative to some higher level landmark. For example, rather than recalling that Disneyland is near Anaheim, which is southeast of Los Angeles in California, users could simply recognize the Disneyland castle in the symbolic representation for California.

In addition to providing scent or residue for actual objects in the world, symbolic representations can also contain objects that provide scent for potential activities (e.g. skiing, surfing) in different places. Instead of helping users identify particular locations, this type of scent can help users locate interesting activities in the virtual world. This facet of symbolic representations may be particularly useful for facilitating exploration of unfamiliar worlds.

Both actual and symbolic place representations help users maintain their orientation in the world. People employ both local and global landmarks when navigating [Lynch][Steck]. In the real world, global landmarks are distant landmarks visible over a large area that help users determine their location relative to other places in the world, while local landmarks are landmarks visible only for short distances that help users find nearby locations and define intermediate goals along a path. When navigating with places and landmarks, the actual and symbolic place representations serve the role of global landmarks. By glancing at the representations that surround them, users can get a sense of where other places in the world are located relative to their current position.Visible landmarks act a little like local landmarks, but function less as intermediate goals along paths because users should be able to move more or less directly to their destinations.

Visible landmarks and place representations are also useful for navigation in poor visibility conditions. In inclement weather (e.g. rain or fog) landmarks and places representations can enable users to remain oriented and navigate about the world even when most of the world is obscured. At night or in poorly lit areas the system can light landmarks and place representations even if the corresponding objects and places are not lit.

I considered allowing users to image plane navigate to a place using its representation, but this approach suffers from accuracy problems. When a user selects an object to image plane navigate to it, the system computes a scaling factor by dividing the distance from the user to the selected object by the distance from the user’s hand to his eyes. When the user subsequently moves his hand some distance closer to or farther from his eyes, the system moves the user that distance multiplied by the scaling factor closer to or farther from the object. Considering that users can typically extend their arm only half a meter to a meter, trying to image plane navigate to a very distant object can result in extremely large scaling factors. The problem with image plane navigating to distant places is that the scaling factors are so high that very small motions of the user’s hand result in very large jumps in the user’s position in the virtual world, making accurate travel to a particular location nearly impossible. Instead of image plane navigation, I developed a discontinuous method for users to travel using place representations.

Users initiate travel to a place with a visible actual place representation by image plane selecting the representation with either hand. This action “pulls away” the representation: the system teleports and attaches the selected representation to the user’s hand. The user achieves the desired view of the place by positioning the handheld representation. When the user lets go of the representation, the system determines the user’s viewpoint relative to the representation, calculates the corresponding viewpoint relative to the actual place, and instantaneously teleports the user to that viewpoint. Users can thus travel to any position in a place with a single gesture using its actual representation: select, position, and release. Users can also pull away an actual representation but decide not to travel there. In that case the user holds the representation at waist level before letting go of it.

Teleporting by achieving the desired view of a handheld representation draws inspiration from a method of transitioning between two locations in the VIEW system [Fisher89]. In the VIEW system, users could view a remote location through a “window” overlaid on their view of the current location. Users could travel to that remote location by stretching the window to fill their view.

Users travel to a place with a visible symbolic place representation with a slightly different process. While users still pull away symbolic representations by image plane selecting them, achieving a viewpoint and teleporting does not make much sense for a symbolic representation because the possible views of the representation do not directly match any views of the represented place. Instead, users reach into a symbolic representation and pull out the representation of a lower place in the hierarchy. Typically users can only pull out the places that are the immediate children in the hierarchy of the represented place, although designers could allow users to pull out even lower level places. The scaled copies of landmarks in a symbolic representation provide residue or scent of where in the representation to grab to navigate to a particular place. The representation that the user pulls out will be either actual or symbolic depending on whether or not the retrieved place is a leaf in the hierarchy. If the retrieved place is a leaf node in the place hierarchy, the user pulls out an actual representation of the place and can step into it using the method described above. Otherwise the user pulls out a symbolic representation of the place and reaches into it to access yet another, lower level in the place hierarchy.

FIGURE 4-8. Symbolic representations of the East and South Plains.

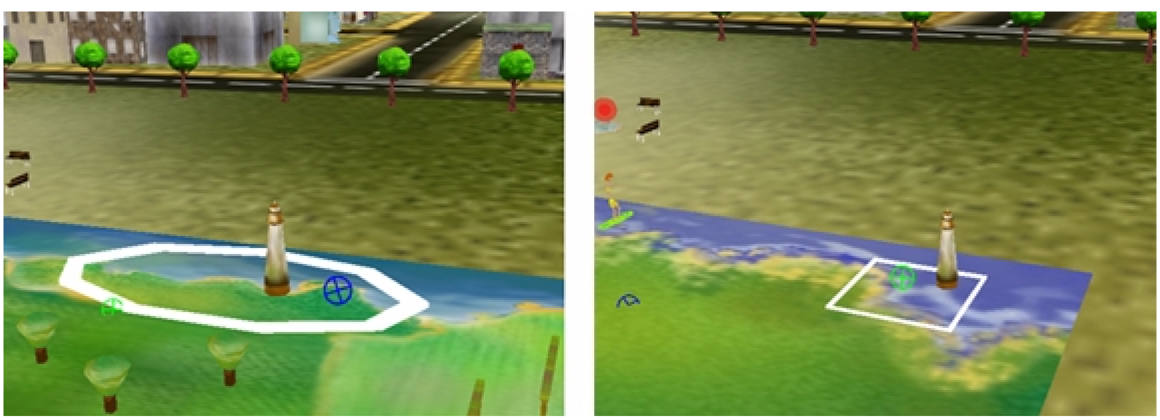

FIGURE 4-9. The user pulls away the representation of the East (left) and then pulls out the representation of the Coast (right).

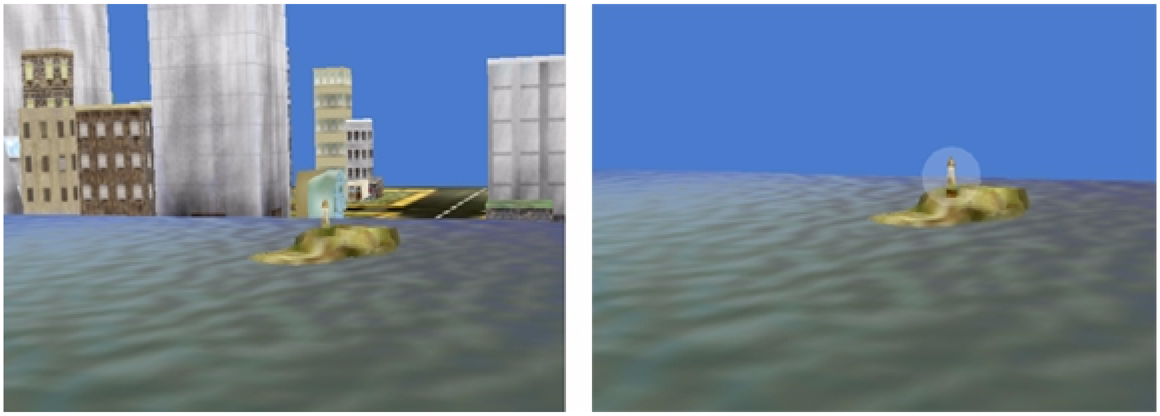

FIGURE 4-10. The user, standing in the City, pulls out the representation of the Beach and positions it to achieve the desired view (left). The user then lets go of the representation to teleport to that location in the Beach (right).

Figures 4-8 through 4-10 present examples of what this process looks like. In Figure 4-8 the user is located in a city on the west coast. He looks to the east to see symbolic representations of the East and the South Plains floating at waist-level along the vector from the user to the represented place. The distance to the representations depends on the relative distances to the represented places. The user pulls away the representation of East, shown on the left in Figure 4-9, by selecting it with his left hand and positions his right hand (represented by the blue crosshair) to pull out the East Coast. The user pulls out the symbolic representation of the East Coast, on the right in Figure 4-9, and positions his left hand (represented by the green crosshair) to pull out the Beach. The user pulls the out the actual representation of the Beach, positions it to achieve the desired view, shown in the left in Figure 4-10, and teleports there. The right side of Figure 4-10 shows the user's new view in the Beach.

I provide one final method of navigating using place representations. Just like in the real world, every place in the hierarchy has a name. Users can verbally specify the name of a place in the hierarchy to cause the system to display the representation for that place. The user can then navigate using that representation. This method provides a shortcut for accessing places low in the hierarchy that are not currently visible. In addition, this method provides a fallback when the user forgets the name of a particular place, but remembers the name of the enclosing place. The user can say the remembered place name to display its representation, recognize the desired place depicted within it, and travel there.

The combination of these two components, visible landmarks and place representations, comprises a new technique for navigating large virtual worlds. I will subsequently use "place and landmarks" as a shorthand to refer to this technique.

4.3 Implementation Details

In the previous section I described navigation with places and landmarks in general terms. In this section I will focus on the design decisions and implementation details for the particular version that I created for the user study. The design choices I made were heavily influenced by my decision to build the user study world as a discontinuous world.

There are two primary ways to structure a large virtual world. The first, obvious approach is to structure the 3D world as a continuous whole, just like the real world. This approach allows users to travel between any two points in the world using a continuous navigation technique. While a continuous structure is familiar to users, it can raise resolution problems on 32-bit computers. The spatial resolution that a 32-bit computer can provide decreases with distance from the world's origin; at 1000 kilometers from the origin it can only provide 2.6 inches of resolution

[Barrus]. Another drawback to continuous virtual worlds is that the designer has to build and the system has to represent the entire world, even if there are large, empty stretches between the main areas of interest that users will likely never visit.

Instead of constructing a continuous world, a designer can instead build the areas of interest as separate pieces, each with its own coordinate system. Designers only need to spend time constructing those parts of the virtual world that users can actually reach. An advantage of navigating with places and landmarks is that designers can use the symbolic representations to create the illusion of a large, continuous virtual world even though the actual structure of the world is discontinuous. Designers can also reinforce the semantic structure of the world without wasting space by adding wide spaces between semantic units in the symbolic representations and not in the actual places. Combining symbolic representations with a discontinuous world can also make it easier for designers to add to a world. Designers can create the new section of world separately, and then insert it into the world by adding it to place hierarchy and creating scent for it in the symbolic representations of places above it in the hierarchy. This allows designers to create broad sketches of the whole world using the symbolic representations, and then fill in the details with actual places later on.

Because the user study participants would only be visiting a small number of pre-determined locations, I chose to implement the user study world as a discontinuous world. Although the symbolic representations of the virtual world present the illusion that the world is a very large space (approximately 1,000,000 square kilometers), users can actually visit only a small fraction of that space (around 10 square kilometers). The design decisions that I made implementing the technique for this study reflect the world structure I chose. While I will focus on the decisions that I made, where appropriate I will also discuss how the implementation might differ for a contiguous virtual world.

4.3.1 Hierarchy details

My implementation of the place hierarchy structures the hierarchy as a tree where each node corresponds to a place. The children of a particular node correspond to the places contained in that particular place. Every node stores a link to the 3D object that represents the corresponding place. Leaf nodes in the hierarchy, *ActualPlace* instances, link to actual representations, while the root and interior nodes, *SymbolicPlace* instances, link to symbolic representations.The root and interior nodes store the boundaries of the places they contain. In my implementation the different places in the world are distinct 3D objects, so leaf nodes also link to the corresponding place object. In an implementation of this technique for a continuous world, leaf nodes might instead store the midpoints and semantic boundaries of the places they represent.

A primary advantage of this structure is that it scales well. For example, borrowing a sample hierarchy from the real world, labeling the levels of the hierarchy neighborhood, town/city, county, state, region, country, continent, hemisphere, and planet results in a hierarchy for the Earth only 9 levels deep. Given this hierarchy, a user navigating with places and landmarks should be able to reach any location in the world in at most 9 discrete steps (assuming the user knows where the location is).

4.3.2 Place and place representation details

I implemented places using the base class AbstractPlace and two subclasses, ActualPlace and SymbolicPlace. The base class AbstractPlace implements shared functionality, such as storing the place's parent and children in the place hierarchy and displaying the place's representation to the user. The SymbolicPlace class implements the functionality for pulling out the representations of its children. The ActualPlace class stores a list of the place representations that are visible in it and the locations of the represented places relative to it. The ActualPlace class also implements the functionality for teleportation.

I chose to explicitly link each place to the place representations that are visible in it. The user study world was small enough (four actual representations, and eight symbolic representations) that explicitly creating the links for each place was a viable choice. For larger virtual worlds and virtual worlds that change over time another possibility would be to choose the place representations to display based on the current place's neighbors in the place hierarchy.

Every frame the ActualPlace instance for the user's current location positions the visible place representations about the user. For each representation the instance calculates the 2D (X,Y) vector from the user to the center of the represented place and positions the representation at waist-level along that vector. The instance positions the representation for the nearest place 1.5 meters away along its vector and the representation for the farthest place 2 meters away along its vector; the instance assigns the other representations a distance between these two extremes based on the relative distance to the place they represent. In my implementation the actual and symbolic representations are slightly different sizes: actual representations are 2/3 of a meter on a side, while symbolic representations are 1/2 of a meter on a side.

I created the actual representations of places by creating copies of the 3D place objects and scaling the copies down. I chose to create the symbolic representations manually instead of attempting to automatically synthesize them to maximize the communicated information. I also decided to manually select in advance which landmark copies to incorporate into symbolic representations. I could have elected to use a more flexible process (e.g. weighting the landmarks within each place and determining which landmarks to use for scent in a symbolic representation by dynamically polling each of the contained places for its most important landmark), but unless the scent needs to change frequently (e.g. when designers frequently add new sections to the virtual world) a flexible process is not really necessary.

Unlike a WIM [Stoakley], users holding actual or symbolic representations can change the position of the representation, but not its orientation. There are two reasons for this difference. First, it forces users to rotate their heads when achieving a desired viewpoint in order to teleport, and research on spatial updating suggests that users maintain their orientation better when they turn their heads than when they turn the world around themselves [Chance]. Second, it avoids a problem shared by flying into the WIM [Pausch95] when the orientation of the desired viewpoint that users specify does not match the current orientation of their head. If I allowed users to change the orientation of place representations then a user could tilt an actual representation to achieve a viewpoint looking down at the ground while his head faces forward in the real world. When the user teleports, two possible things could happen. First, I could reorient the user so that when facing forward he is looking down. Second, when I teleport the user I could preserve his orientation so that he appears in the new location facing forward instead of down; the result instantaneous change in view could disorient the user. Neither choice is ideal, so I chose instead to avoid the problem by not allowing users to change the orientation of place representations.

4.3.3 Landmark details

Each ActualPlace instance also stores a list of the landmarks in that place. The designer makes an object in the place a landmark by placing pointers to both the original object and a copy of it in the list. At runtime the ActualPlace instance for the user's current location is responsible for displaying its landmarks to the user. Every frame the instance positions the copy of each landmark one meter away from the user's viewpoint along the vector from the user's viewpoint to the actual landmark object. The instance calculates the new size for the copy by multiplying the landmark's size by the distance from the user to the copy and dividing by the distance from the user to the landmark. The instance then adjusts the copy's size if necessary to ensure that the copy maintains a minimum size; in my implementation the minimum size is 0.05 meters.

The ActualPlace instance is also responsible for displaying the transparent halo behind each landmark copy (see Figure 4-11). Each halo is a transparent circle that the instance centers slightly beyond a copy of a landmark (as seen by the user), resizes to half again the size of the copy's largest dimension, and then orients orthogonal to the user. These halos also help users select landmark copies because users can select anywhere on a landmark's halo to select that copy.

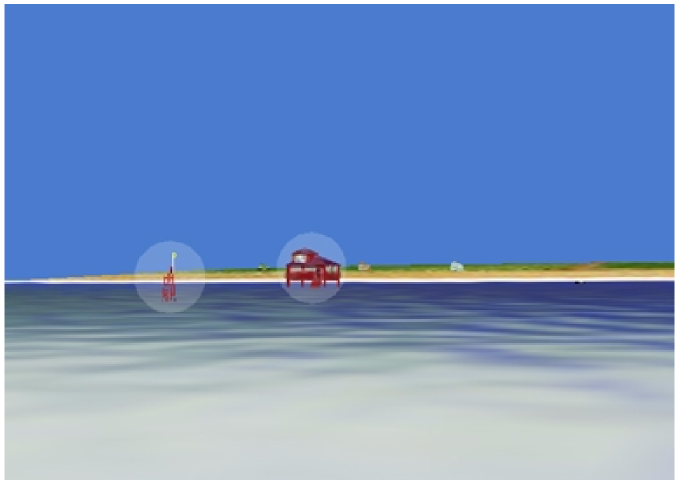

FIGURE 4-11. Halos help users locate and select objects. Instead of trying to select the legs on the lifeguard chair, a user can select anywhere on the halo.

4.4 Theoretical Analysis

George Furnas' theory of effective view navigation [Furnas97] shares the same basic premise as this chapter: scale issues are critical. Furnas proposes visualizing the visibility between nodes with a view graph. The view graph connects each node with the nodes visible from it. His theory focuses on information visualization where discrete nodes represent items of information, but his ideas can also apply to continuous 3D virtual worlds by making the discrete nodes arbitrarily small. According to Furnas' theory, to provide effective navigation a technique needs to provide both effective view traversal and view navigation.

Effective view traversal imposes two requirements: each node in the view graph must have a small number of outgoing links relative to the size of the graph, and the distance between any pair of nodes in the view graph must be small relative to the size of the structure. These two requirements can be simply expressed as *small views* and *short paths*. The former requirement essentially states that users should have a small number of immediately visible destination choices (to avoid overwhelming the user), while the latter requirement states that users should be able to reach any destination with a small number of discrete steps.

View navigation also imposes two requirements: every node must have good residue at every other node, and the outlink information at a node must be small. Furnas defines residue as the remote indication of a target in the outlink information throughout the view graph. He defines outlink information as the information available to the user about what locations can be reached by traveling to a visible destination. These requirements essentially state that users should be able to easily determine what path to follow to reach any destination, and that the amount of information about what destinations are available along a path should not overwhelm users.

Navigating with places and landmarks provides both effective view traversal and view navigation. Within any place users have small views because they can only view the landmarks in the current location and a small number of place representations, and the landmarks and place representations are typically distributed around them in the scene. This technique also provides short paths: the number of steps required to reach any place is at most the number of levels in the place hierarchy. Landmarks and symbolic representations balance the requirements of good residue for all possible destinations and small outlink information by displaying the important objects in the current place and represented places.

Based on this theoretical analysis, navigating with places and landmarks should provide effective view navigation. In the next section I will discuss an experimental evaluation that attempts to verify this analysis.

4.5 Experimental Evaluation

Although useful in predicting the potential strengths of navigating using places and landmarks, theoretical analysis alone is insufficient for demonstrating concrete improvements over other best practice navigation techniques. Therefore I conducted a formal experimental evaluation of navigation with places and landmarks.

4.5.1 Choice of metric

The first step in preparing the formal evaluation was choosing a suitable metric for demonstrating the advantages of the technique. Unlike studies of manipulation techniques, where the primary metrics are speed and accuracy, studies of navigation techniques can focus on a wider variety of metrics. The navigation task is composed of several sub-tasks, primarily building a cognitive map, planning, and planning execution. Formal studies of navigation techniques can focus on metrics associated with any or all of these sub-tasks.

Previous studies on how navigation techniques and aids affect the development of cognitive maps [Darken96] [Patrick] have studied continuous navigation techniques (generally flying or walking) in relatively small, continuous virtual worlds. In theory, navigating using places and landmarks should aid users in developing a cognitive map. Continuously visible landmarks and place representations should help users to construct a cognitive map by helping them locate new areas relative to known reference objects. When navigating to distant locations, the ability to get a bird's eye view of larger portions of the world should help users locate places relative to each other. Finally, viewing large portions of the world symbolically and interacting with the place hierarchy should help users learn the organizing principle of the world, which researchers believe facilitates the development of a cognitive map.

In addition to aiding the construction of an accurate cognitive map, a good navigation technique should help users determine what information to add to the cognitive map. Merely choosing destinations from a list or flying about a virtual world can make it very difficult for novices to explore the virtual world and locate the important or interesting areas. A good navigation technique should include aids that make it easier to discover these areas. Navigating with places and landmarks will provide information scent that should do exactly that, provided designers choose the landmarks in the world and the residue for symbolic places carefully.

Places and landmarks should also help users search efficiently. Dividing a world into places provides an organization for search in a virtual world [Darken95]. When users navigate with places and landmarks they only have to examine a discrete number of places (represented by the leaf nodes in the place hierarchy) in order to completely search everywhere in the virtual world. Users do not even need to travel to all of those places; they can simply examine they places' actual representations.

Other studies of navigation techniques have measured how well users remain oriented in the world during and after travel [Bowman99a][Chance]. These studies examine how a navigation technique affects spatial updating. Maintaining orientation during travel is important because users need to be able to evaluate their progress in executing their navigation plan. Maintaining orientation after travel is important if the user needs to locate a nearby object in the world, or navigate to a new location. Places and landmarks should aid users in maintaining orientation during continuous travel because nearby landmarks and surrounding places remain visible at all times and provide information about their relative positions.

More importantly, navigating using places and landmarks should aid users in maintaining and recovering orientation after a discontinuous transition. Existing techniques involving discontinuous motion can disorient users. Teleportation to a named destination may force users to spend significant effort reorienting themselves if they are not completely familiar with the destination. Flying into a WIM may aid users in orienting themselves in the place depicted in the WIM, but may disorient them with respect to areas outside the WIM. For example, users may be unable to determine where their previous location is relative to their new location. When users teleport using places and landmarks, however, the presence of visible places and landmarks surrounding them after the transition should make it easier for users to reorient themselves both in their immediate surroundings and in the larger world.

One of the simplest metrics to obtain is the time required to travel from the user's current location to a specified target location. While measuring travel time is easy, the processes involved in this task may be complex, particularly if the target location is out of sight. The user must draw on his cognitive map of the space to create a plan for reaching the target location. In theory the more accurate the user's cognitive map, the more efficient his plan will be. Then user must then execute his plan as quickly as possible. Navigating with places and landmarks should yield low travel times. Visible landmarks and place representations that provide good residue should reduce the need for planning. Users should also be able to efficiently execute their plan, reaching any location in the world with a small number of discrete gestures.

A formal study can also examine how effectively experts with a navigation technique can move about the environment, and how efficiently the navigate technique allows novices to become experts. With continuous navigation techniques, expert behavior is typically associated with a more comprehensive cognitive map that allows experts to develop more efficient plans. However, I believe the most efficient technique is one that eliminates planning for experts altogether. The most efficient navigation technique for experts should be to verbal specify the desired location and teleport there. The travel time would be limited only by the time required for the user to say the destination and for the system to recognize it. However, most navigation techniques do not provide a smooth transition path to reaching this expert status. Navigating with places and landmarks can aid this transition by overlaying names over places and landmarks so that users can learn them while they move about the world. In addition, the place hierarchy provides a fallback if the user cannot remember the exact name of a location but can remember the name of the enclosing place. For example, if the user cannot remember the name Paris but can remember that it is a city in France, he can tell the system to display the symbolic representation of France and use the information scent in the representation to recognize Paris.

Formal studies of navigation techniques can also examine qualitative factors as well as quantitative ones. For example, a designer may wish to tell a story in a virtual world, to provide users with a compelling experience. Presenting users with a large, lushly detailed world is still a difficult task because of the time required to create the content and the processing power required to display it in real time. We are currently usually forced to choose between small, detailed worlds or large, sparse ones. In some cases designers can rely on stories in other media to make a world more compelling [Pausch96], but this approach is generally not possible for completely novel stories. Navigating with places and landmarks can make a world seem larger and more complex than it actually is by displaying regions in the symbolic representations that have no corresponding actual representation. The lack of any actual representation for those regions prevents users from actually visiting them. This constraint contrasts with techniques like pointing to a map to teleport or flying into a WIM because those techniques do not provide any mechanism that prevents users from visiting depicted locations.

All of these studies could potentially demonstrate some advantages of navigating with places and landmarks over existing techniques. I chose to focus on measuring travel time because the ability to efficiently navigate around a space is the most widely applicable feature of any navigation technique. Once I have established the value of navigating with places and landmarks by determining its efficiency relative to other navigation techniques, then in the future I proceed to establish the relative value of the technique for other metrics.

4.5.2 Choice of "best practice" competing technique

Next I needed to choose an existing "best practice" technique for comparison. Navigating with places and landmarks allows users to quickly reach any location in the virtual world regardless of distance. I wanted to compare it with a technique that provided users with similar freedom and efficiency.

I briefly considered comparing against a continuous navigation technique such as pointing-to-fly, image plane navigation, and world scaling. These techniques allow users to navigate anywhere in the world and are fairly efficient for navigation to nearby locations, particularly when those locations are in view. However, they are much less suitable for navigation over long distances. Even if users utilize a navigation aid such as a map [Darken93] to simplify the planning step, the time to reach distant locations still increases linearly with the distance to the target location.

That left discontinuous navigation techniques as potential candidates. I decided not to compare against teleportation when the destination is choosing from a list of names or worldlets for two reasons. First, a list of destinations does not scale well as the world gets larger and the number of destinations increases. The user could spend a significant amount of time just trying to locate the desired destination in the list. Second, a discrete list of destinations does not allow users to travel directly to any location in the world. Instead, users are forced to find the closest listed point and then utilize another technique to navigate from there to the actual target location.

I also briefly considered comparing navigation with places and landmarks against teleportation when the destination is verbally specified. However, I believe these techniques are more complementary than competitive. As I argued in section 4.1.1, navigating with places and landmarks is probably more effective when users are exploring new areas or have forgotten the names of destinations in previously visited areas, while teleporting to verbally specified destinations is probably more effective when the user is travelling to a well-known destination.

The most promising candidate was one of the techniques where the user specifies the desired destination on a handheld widget to teleport there. This type of technique has the potential to allow users to quickly specify and reach the desired destination. I considered three issues in choosing the appropriate variant of this technique for comparison. First, I needed to decide on the form of the handheld widget. Second, I had to decide how the technique would provide sufficient resolution for users to accurately specify the desired destination. Third, I needed to decide how the technique would manage the user’s transition from his current location to the specified destination.

Researchers have developed two main widgets for users to specify destinations: a map and a world in miniature (WIM) [Stoakley]. The former is a familiar navigation aid in the real world, while the second allows users to view locations in three dimensions instead of two. I chose to use a WIM for the handheld widget because the ability to view locations from the side facilitates recall: users typically view objects from the side when working the world, so recognizing them from the side is typically easier than recognizing them from the top.

The next issue was that a handheld WIM that depicted the entire virtual world would not provide sufficient resolution to allow users to accurately specify a destination. Consider a square virtual world approximately a thousand kilometers on a side. If the WIM is a meter on a side (which is actually somewhat large for a widget of this type) a millimeter in the WIM would correspond to a kilometer in the virtual world. Because existing tracking systems are typically only accurate within a few millimeters, users could be several kilometers off when trying to specify a destination. As a result, users need a method for improving the resolution.

Existing work suggested two possibilities. I could implement the WIM as a fisheye display [Furnas86][Lamping]. A large area of the WIM, the focus, would display a small portion of the virtual world. The rest of the WIM, the context, would display a compressed version of the rest of the virtual world. Alternately I could add panning and zooming to the WIM [Stoakley][LaViola]. Panning would allow users to change the displayed section of the world without affecting the zoom level. Zooming out would allow users to display more of the world, while zooming in would allow users to display a smaller section with more resolution.

Both of these alternatives are largely hypothetical. Researchers have developed simple panning and zooming WIMs [LaViola], but only for very small virtual worlds, leaving as open questions what the strengths and weaknesses of panning and zooming WIMs are for very large virtual worlds. There has been even less work on developing dynamic fisheye views of 3D information. The Perspective Wall [Mackinlay91] is a 3D fisheye view, but displays 2D information. Carpendale explored how to extend fisheye or distortion views from 2D information to 3D information, but focused on static views of simple 3D structures [Carpendale] rather than dynamic interaction with a complex 3D object. Because of the hypothetical nature of these alternatives, I resorted to a theoretical analysis of their strengths and weaknesses.

The theoretical advantages of a fisheye WIM are that the focus provides sufficient resolution to specify a destination, and that users can quickly shift the focus to a different part of the virtual world. A significant potential drawback, however, is that users may have difficulty positioning the focus to display the desired part of the world. Consider a WIM that is half a meter on a side and a world that is 1,000,000 square kilometers. If the focus is a quarter of a meter on a side and displays a square kilometer section of the world, the remaining WIM area must display the remaining 999,999 square kilometers. If users point to a part of the world in the context to position the focus there, they may have trouble indicating the desired focus accurately. More importantly, users may have significant difficulty even locating the desired section of the world in the context because of the extremely high scaling factor (approximately 16000000 to 1).

The theoretically advantage of a panning and zooming WIM is that the accuracy available for specifying the destination is limited only by the amount of time that users spend zooming in. The most likely drawback is that a panning and zooming WIM does not provide any context; the user cannot see the world outside the currently displayed section without zooming out. The lack of a context means that panning is only useful over short distances; to travel to a distant locations users must zoom out to find the location and then zoom back in on it. As a result users may require more time to travel between distant locations.

In the end I chose to compare navigation with Places and Landmarks against a panning and zooming WIM instead of a fisheye WIM. While panning and zooming WIM users might take more time to zoom in on a particular destination, I was more confident that they would be able to locate destinations in the WIM. My goal was also to compare navigation with Places and Landmarks against an existing best practice navigation technique, and fisheye WIMs are largely conjectural; researchers have at least implemented simple versions of panning and zooming WIMs.

Having settled on building a panning and zooming WIM, I had to make several design decisions in order to extend its functionality to very large virtual worlds. I experimented with several implementations of panning and zooming for the WIM. I found that constant zoom rates, although simple to activate, did not provide sufficient control. Small constant zoom rates made zooming in or out take too long, but large constant zoom rates made it too difficult to make small adjustments to the zoom level. I finally settled on a variable zoom rate where users moves their hand down to zoom in and up to zoom out; the farther users move their hand the larger the zoom rate. I experimented with a variable pan rate as well, but found it too difficult to control. I instead implemented panning using direct manipulation: users grab the miniature world and drag it around the handheld widget.

I also needed to decide how the technique would manage the user’s transition to a specified location. The existing approaches are to abruptly teleport the user [Angus] or to fly the user into the miniature and, when he reaches the specified location in the miniature, teleport him to that location in the world [Pausch95]. The drawback to an abrupt transition is that it disorients users, who must spend time recovering their orientation in the space. The drawback to flying the user into the WIM is that the transition introduces a delay before the user reaches that location. In addition, the system needs to be able to generate an appropriate path to fly the user into the WIM, which can be difficult if the user specifies a change in orientation as well as position. I chose to avoid both drawbacks by using the same type of transition that navigation with places and landmarks provides: the user positions the WIM to achieve the desired viewpoint, and when the user lets go of the WIM the system teleports him to the corresponding position in the world.

In addition to these design decisions, I added several features to my version of a panning and zooming WIM to avoid making a “straw man” comparison. If a WIM merely showed a scaled version of the virtual world, users would have a hard time finding particular features or locations when the WIM was zoomed out. To help address this problem, I added semantic zooming [Donelson] so that the WIM displays the same symbolic place representations that navigation with places and landmarks provides. My implementation of the WIM chooses which symbolic representation(s) to display based on the current zoom level of the WIM. I also added the same residue to these place representations.

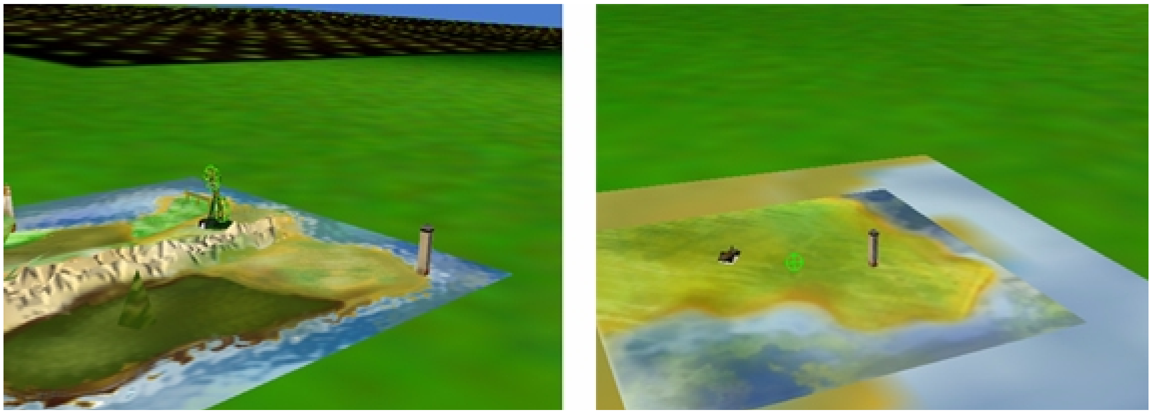

FIGURE 4-12. The fully zoomed out the WIM (left) shows the entire world. As the user zooms in on the hospital the WIM present a more detailed view of the area around the city (right).

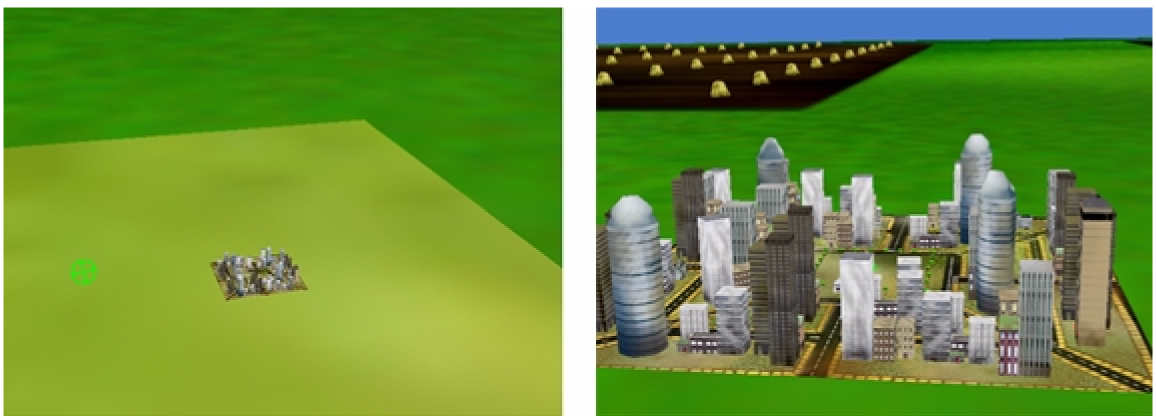

FIGURE 4-13. If the user zooms in farther the WIM shows the actual downtown (left). The user can zoom in farther to view this area in more detail (right).

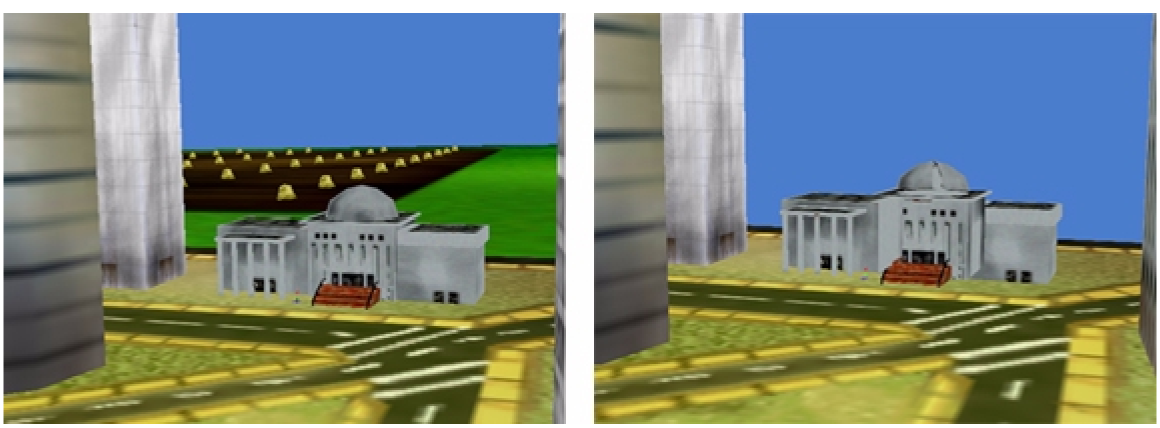

FIGURE 4-14. To travel to the city, the user positions the WIM to achieve the desired viewpoint (left) and lets go of it to teleport to that location (right).

Figures 4-12 through 4-14 show examples of what navigating with my implementation of a panning and zooming WIM looks like. The user starts with a fully zoomed out version of the WIM, on the left in Figure 4-12, that displays the symbolic representation of the entire world. The 3D objects in the WIM provide residue for particular locations. If the user zooms in on the hospital, the WIM will display the symbolic representation of the SouthWest where the city is located, on the right of Figure 4-12, overlaid over the symbolic representation of the entire world. The house in this representation provides residue for the suburbs, while the hospital provides residue for the downtown area. If the user zooms in even further on the hospital the WIM will display the city downtown, on the left in Figure 4-13, overlaid over the SouthWest symbolic representation. The user can view this area in more detail, shown on the right in Figure 4-13, by zooming in farther. To travel to the city, the user positions the WIM to achieve the desired viewpoint, shown on the left of Figure 4-14, and lets go of it to teleport to that location. The right side of Figure 4-14 shows the user now located in the city.

4.5.3 Analysis of Advantages

The primary advantage of navigating with places and landmarks over a panning and zooming WIM is that users choose place representations with pre-assigned semantic zoom levels and focus points instead of manually specifying the zoom level and focus point of a WIM. Choosing simplifies a task by constraining the available actions. As a result, choosing is generally faster than specifying. The advantage of choosing instead of specifying should make navigating with places and landmarks easier and more efficient for users. While navigating with places and landmarks simplifies the user's task, it does not restrict travel: users can travel to any location reachable with the WIM.

Another advantage of navigating with places and landmarks over a WIM is that landmarks and place representations appear around the user in an egocentric frame of reference, while a WIM appears in an exocentric frame of reference. Landmarks and place representations saves user a mental coordinate transformation when they know where a place is relative to their current position, because users can look in the correct direction to find the place representation. Users of a WIM must instead first find their location on the WIM and then transform the direction from their (egocentric) coordinate system to the WIM's coordinate system.

Navigating with places and landmarks possesses other advantages over navigating with a WIM that I chose not to test in this experiment. First, users can take advantage of the hierarchy of named places by verbally specifying the desired place to show its symbolic or actual representation. I chose not allow users to draw on this advantage because it would have focused the experiment more on studying the transition from novice to expert behavior. Second, navigating with places and landmarks is more effective for true 3D worlds than navigating with a WIM because visible landmarks and place representations can appear in all directions around the user. WIMs are less effective in 3D spaces, because panning a WIM of a true 3D world requires manipulating an additional degree of freedom, while zooming out a WIM of a true 3D world can cause locations on the edge of the world to obscure locations on the interior. I chose to focus on comparing these techniques in a 2 1/2D world because most virtual worlds only have two and a half dimensions.

Navigation with place and landmarks may also help users develop superior spatial knowledge. Some research on the acquisition of spatial knowledge suggests that users who acquire survey knowledge within an environment may be better at orienting themselves within the environment than users who acquire survey knowledge from a map [Thorndike]. If this research is correct, I believe that users who navigate with places and landmarks stand a better chance of acquiring the former type of spatial knowledge than users who navigate with a WIM because visible landmarks and place representations expose users to the relative positions of different locations in the virtual world while also providing users with top-down views of places.

4.5.4 Task Description

None of the existing testbeds for comparing VR techniques include tasks that compare the ability of techniques to support navigation over long distances. The VRMAT testbed focuses on comparing manipulation tasks, while the VEPAB testbed concentrates on tasks that require walking or flying through small environments. The VR-SUITE testbed [Bowman99a] also concentrates on navigation through small environments.

Since I was unable to choose a task from an existing testbed, I was forced to create my own. As I discussed in chapter 3, there is a general tension when choosing an experimental task. I could choose a highly focused, abstract task and learn a great deal about a small facet of the techniques, or I could choose a more realistic task that provides greater confidence about the representativeness of the results in exchange for less certainty about the contributions of individual factors. I chose the latter, because I wanted to learn how navigating with places and landmarks fared in general against navigation with a handheld WIM.

I settled on the task of navigating to a depicted location. Before each task the system showed users a static view from the next destination of the nearest landmark. Users had six seconds to study the view before the system restored their previous view. Users then had to travel to that destination as quickly as possible. The system displayed a volumetric target (a red sphere two meters across) at each destination; when users judged that they were within the marker they verbally informed the experimenter. I chose this type of task because users often mentally encode the location of a particular place relative to the nearest landmark.

I wanted to vary the task presentation order between subjects, but this was problematic because in order to compare task performance between users both the start and end points for a task needed to be the same. For example, I could not simply have half the users travel from point A to point B to point C and the other half travel from point A to point C to point B. If I had users first travel to the start point for a task and then travel to the end point I would effectively double the number of navigation steps without increasing the amount of task data I collected. Teleporting users to the starting point of each task was not a viable option either, because teleportation under system (rather than user) control can disorient users [Bowman99b]. I chose to settle on a middle ground and arranged the tasks into groups of three. For each group users travel to the start point, and then complete the three tasks within the group one after another as quickly as possible. This arrangement allowed me to vary the task presentation order between subjects, although to a lesser extent, while collecting task performance data for three out of every four navigation steps.

Because both techniques allow users to travel a variety of distances, I divided up the task groups into two sets. The first set consisted of task groups where all of the destinations were in the same place. The second set consisted of task groups where all of the destinations within a group were in different places. Although I varied the order of the task groups within each set, users always performed the sets in the same order. Specifically, users performed the tasks in the within-place set first and the tasks in the between-places set second. I chose this order to allow users to familiarize themselves with the layout of the individual places before navigating between them.

FIGURE 4-15. The symbolic representation of the world the user visits.

The virtual world I constructed (Figure 4-15) contained four discrete places: a Beach, a Farm, an Amusement Park, and a City. The amount of time and effort required to create content for the world constrained the number and size of these places. Modeling and painting the landscapes, place representations, and scenery required two months of artists' time and yielded four places varying from 0.25 to 2.5 square kilometers in size. However, time limitations did not constrain the size of the virtual world itself because I could design the world's symbolic representation to represent as much space as I desired. I settled on an area of 1,000,000 square kilometers in order to make the world large enough to adequately compare these techniques.

I positioned the places in the virtual world to provide between-place navigation tasks where the places were nearby, on the other side of the world, and midway between these two extremes. I placed the City in the south-west corner of the world, while I placed the Beach near the north-east corner. I placed the Amusement Park and Farm approximately 120 kilometers away from each other in the middle of the world's southern edge.

I chose the residue in the symbolic representations for each place from that place's landmarks, selecting a landmark that was both large and uniquely associated with that place. For example, I avoided using the Farm House as the Beach's residue because houses are also found in cities and on the beach. Based on these criteria I selected the lighthouse to provide the Beach's residue, the hospital to provide the City's residue, the ferris wheel to provide the Amusement Park's residue, and the barn to provide the Farm's residue.

If I were constructing a virtual world to only support navigation using places and landmarks I would have some flexibility in positioning the residue in the symbolic representations. However, choosing to comparing against a panning and zooming WIM constrained the placement of the residue objects because users zooming in on a particular object in a WIM had to find the associated place. As a result, I positioned the residue so that it was centered on the associated place in the symbolic representations. However, this posed a problem: the barn and ferris wheel had to occupy essentially the same position in the world's symbolic representation. I could not simply offset the barn and ferris wheel slightly, as I would if the users were to only navigate with places and landmarks, because then zooming in on the barn might not lead to the Farm. While I could have chosen to display both objects and allow them to overlap, I instead chose to display only the ferris wheel in the world's symbolic representation and instruct users that zooming in the ferris wheel would reveal both the barn and the ferris wheel at the next higher level of detail.

FIGURE 4-16. The symbolic and actual place representations users see arranged into the world's place hierarchy.

When I sketched out the place hierarchy for this world (see Figure 4-16), I divided the world into East and West Coasts. The East Coast contained the Middle Plains and East Shore. The Middle Plains contained the Northern Plains and Southern Plains. The Southern Plains contained both the Amusement Park and the Farm. The East Shore contained the North Shore, Middle Shore, and Southern Shore. The Middle Shore contained the Beach. The West Coast contained the Northwest and the Southwest. The Southwest contained the City. This place hierarchy turned out to be more comprehensive than strictly necessary because users never see the representations of many of these nodes. This place hierarchy does facilitate incremental development of the world, however, as a designer could attach a new place to one of those nodes (e.g. adding a ski resort to the Northwest).

In practice users only see the symbolic representations for the world, West Coast, Southwest, South Plains, East Shore, and Middle Shore. With the panning and zooming WIM users do not see the symbolic representations of the other nodes because they have no incentive to zoom in on those areas. Places & Landmarks users only see the representations of places linked to their current location. In the City users see representations of the South Plains and East Shore. In the Beach users see representations of the South Plains and West Coast. In the Farm and Amusement Park users see representations of the Southwest and Middle Shore. Users in the Farm can also see the actual representation of the Amusement Park, while users in the Amusement Park can see the actual representation of the Farm.

The Beach is 1.61 by 1.55 kilometers in size. Its landmarks are the lighthouse (on the right in Figure 4-17), the lifeguard chair (on the beach at the far left of the figure), the red beach house (the second beach house from the left in the figure), the picnic table (on the hill at the middle of the figure), and the red buoy (beyond the lighthouse at the right of the figure). The lighthouse and red beach house are visible from most locations in the Beach, but the other landmarks are only visible from a short distance unless enhanced with visible landmarks.

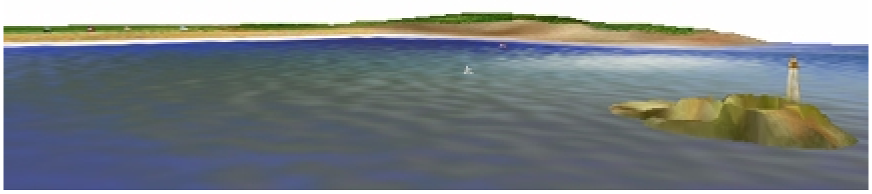

FIGURE 4-17. The Beach.

The Farm is 1 kilometer by 1 kilometers in size. Its landmarks are the white farm house (at the top left of Figure 4-18), the windmill (to the left of the red barn in the figure), and scarecrows wearing green, red, and blue shirts, one in each field (at the bottom, top, and right of the figure). The farm house and windmill are visible from most locations in the Farm, but the scarecrows are only visible from a short distance unless enhanced with visible landmarks.

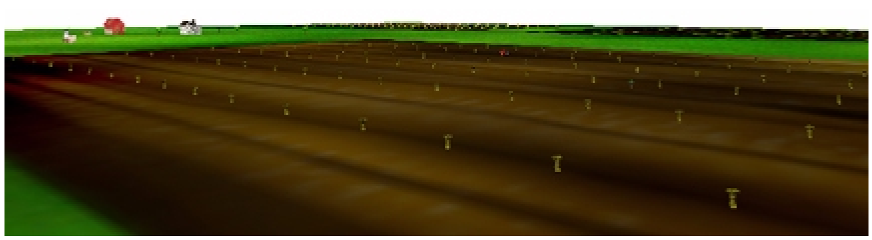

FIGURE 4-18. The Farm.

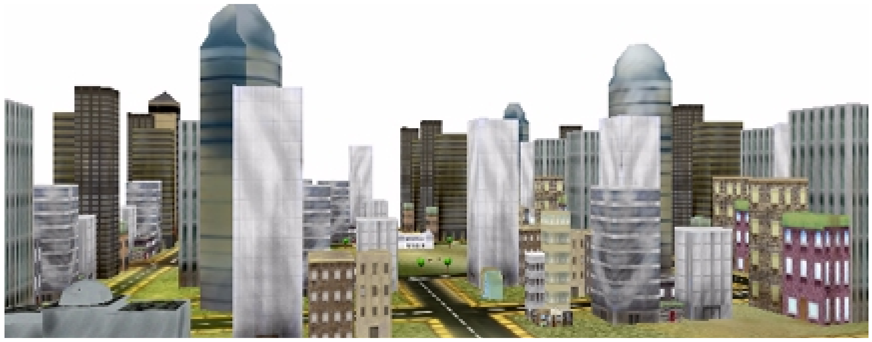

FIGURE 4-19. The City.

The City is 0.5 kilometers by 0.5 kilometers in size. Its landmarks are the hospital (at the back left of Figure 4-19), the museum (at the front left in the figure), city hall (in the middle of the figure), the park statue (in front of city hall in the figure), and the church (hidden in the figure, but located on the right side). All of the landmarks are large enough to be visible from some distance away, but they are frequently obscured by nearby buildings unless enhanced with visible landmarks.

FIGURE 4-20. The Amusement Park.

The Amusement Park is 0.87 kilometers by 0.98 kilometers in size. Its landmarks are the roller coaster (in the middle at the top of Figure 4-20), the ferris wheel (on the right in the figure), the octopus ride (on the bottom right of the figure), the swings (to the right and a little below the parking lot in the figure), and the carousel (above and to the right of the swings in the figure). All of the rides are large enough to be visible from the opposite side of the Park.

4.5.4.1 Practice for navigation within places

All users completed the same twelve practice tasks. The initial practice tasks blended navigation within places and between places. Starting in the Beach near the red beach house, users navigated to two other locations in the beach (near the lighthouse and near the red buoy). Users then navigated to two locations in the City, one near the hospital and one near the church. Next users navigated to two locations in the Farm, one near the windmill and one near the blue scarecrow. After those tasks users navigated to two locations in the Amusement Park, one near the octopus ride and one near the roller coaster. I gave users a short break (2-3 minutes) after completing these initial practice tasks. The remaining practice tasks all focused on navigation between places. Users navigated to the City near the museum, to the Beach near the picnic table, to the Amusement Park near the ferris wheel, and to the Farm near the green scarecrow. After users completed these tasks I gave them a longer break (4-5 minutes) before starting the experimental tasks.

4.5.4.2 Navigation within places

The first set of experiment task groups focused on within-place navigation. Users completed four task groups, one within each place, for a total of 16 navigation steps, 12 of which were within-place experiment tasks and 4 of which were between-place navigation to the starting point of the next task group. I randomized the group order within this set between users to guard against effects from a particular task order. I gave participants a short break (1-2 minutes) after completing half of the task groups, and gave them a longer break (4-5 minutes) after completing all the task groups in the set.

In the Beach users started near the red beach house and navigated to locations near the picnic table, lifeguard chair, and red buoy. In the Farm users started near the farm house and navigated to locations near green, blue, and red scarecrows. In the Amusement Park users started near the octopus ride and navigated to locations near the ferris wheel, swings, and roller coaster. In the City users started near city hall, and navigated to locations near the church, museum, and city hall. This set of task groups yielded 12 measurements of within-place navigation.

4.5.4.3 Navigation between places

The second and final set of experiment task groups focused on between-place navigation. Users completed six task groups for a total of 24 navigation steps, 18 of which were between-place experiment tasks and 6 of which were navigation to the starting point of the next task group (which may have been within- or between-place depending on the end point of the previous task group). I again randomized the group order within this set between users to guard against effects from a particular task order. I gave users short breaks (2-3 minutes) each time they completed two task groups (8 navigation steps).

Starting near the red beach house in the Beach, users navigated to the Amusement Park carousel, the red scarecrow in the Farm, and the City hospital. In the first task group starting near the Farm house, users navigated to the red Beach house, the City hall, and the Amusement Park carousel. In the second task group starting near the Farm house, users navigated to the Beach lifeguard chair, the Amusement Park swings, and the City church. In the first task group starting near the City hall, users navigated to the Farm house, the Amusement Park ferris wheel, and the Beach picnic table. In the first task group starting near the Amusement Park octopus ride, users navigated to the City fountain, the Farm windmill, and the red Beach buoy. In the second task group starting near the Amusement Park octopus ride, users navigated to the City museum, the Beach lighthouse, and the blue scarecrow in the Farm. This set of task groups yielded 18 measurements of between-place navigation, bringing the total to 30 experimental task measurements.

4.5.5 Method

20 undergraduate and graduate students, 18 male and 2 female, participated in this experiment. Half of the users navigated using places and landmarks, while the other half navigated using a handheld WIM. I chose a between-subjects design rather than a within-subjects design primarily to reduce the amount of time users had to spend in the head-mounted display (HMD). Before starting the experiment users completed a simple questionnaire to determine their gender, age (within a 5 year range), handedness, computer experience, and previous experiences with virtual reality. Users also completed an absorption questionnaire, and I administered the ETS S-2 (cube comparison) spatial ability test to each user.

After participants completed the paperwork I showed them a map of the virtual world and pointed out the location of each of the places that they would visit. I pointed out the landmarks in a top-down view of each place, and I showed users a short video fly-through of each place that visited its landmarks. I then stepped into the virtual world and demonstrated how to use the assigned navigation technique. Finally, I demonstrated how to perform a navigation task by completing a sample between-place navigation task from the Farm to the red Beach house.

Participants then stepped into the virtual world and completed the practice tasks. After a break, they completed the within-place experimental tasks. The participants then took another break before completing the between-place experimental tasks. Participants then completed another short questionnaire to determine the easiest and hardest parts of using the assigned navigation technique, and whether they had experienced any discernible arm fatigue, dizziness, or nausea.

4.5.5.1 Apparatus

I implemented the virtual worlds in the Java version of Alice [JAlice], an in-house authoring tool for building 3D worlds. For the experiment I ran the worlds on a Pentium III PC. All participants wore a Virtuality Visette Pro (640 x 480 resolution, 60 degrees x 46.8 degrees field of view) and FakeSpace PinchGloves. The system tracked the HMD and input devices using an Ascension SpacePad with a latency of approximately 100 milliseconds. All of the worlds ran at a minimum of 30 frames per second.

4.5.5.2 Performance measures

For each task I measured the time between when the system stopped displaying the next destination and when participants announced that they had reached that destination and were within the target marker. I also measured participants' final distance from the center of the target marker so that I could determine if they had really reached the destination.

4.5.6 Results

Spatial ability had a significant and inverse effect on task performance for Places and Landmarks users ($p < 0.001$), but did not have a significant effect on performance for WIM users ($p > 0.5$). This result is not surprising. Knowledge of the world's layout can help Places and Landmarks users reach a destination faster because they will know which direction to turn to locate a landmark or place representation. WIM users, by contrast, can always refer to the handheld representation of the world instead of trying to memorize its layout. Absorption did not have a significant effect on task performance for either Places and Landmarks or WIM users.

I conducted a repeated measures analysis of variance on the results with the interaction technique as the between-subjects variable and performance on the tasks as the within-subjects variable. Overall Places and Landmarks users navigated to the destinations faster than WIM users ($F_{1,18} = 24.163$, $p < 0.001$). Places and Landmarks users completed within-place tasks faster than WIM users ($F_{1, 18} = 9.701$, $p = 0.006$), suggesting that visible landmarks did help users navigate to a destination relative to a landmark faster. Places and Landmarks users also completed the between-place tasks faster than WIM users ($F_{1,18} = 24.982$, $p < 0.001$). Because the difference in performance for the between-place tasks could have been due solely to the visible landmarks and not the place representations, I measured the time from the start of the task until the user teleported into the place containing the destination for ten of the users (five for each technique). Places and Landmarks users reached destination places faster than WIM users ($F_{1,8} = 18.564$, $p = 0.003$), suggesting that teleporting with the place representations was faster than teleporting with the WIM. Visible landmarks probably did aid performance on between-place tasks; Places and Landmarks users took less time to reach the target after teleporting than WIM users ($F_{1,8} = 19.347$, $p = 0.002$).

Quantifying exactly how much faster Places and Landmarks users were than panning and zooming WIM users is difficult because task performance varied by task. However, I can quantify the time difference in general by comparing the average performance on the within-place and between-place tasks for both techniques. On average Places and Landmarks users completed the within-place tasks 21.67% faster (29.813 seconds vs. 38.063 seconds) and the between-place tasks 37.77% faster (36.403 seconds vs. 58.495 seconds).

In addition to superior task performance, Places and Landmarks users experienced less arm fatigue ($F_{1, 18} = 4.629$, $p = 0.045$) and dizziness ($F_{1, 18} = 6.818$, $p = 0.018$) than WIM users; two of the ten WIM users reported that arm fatigue was one of the "hardest three things" about using the technique. The greater dizziness experienced by WIM users is mostly likely due to the fact that WIM users spent more time in the virtual world than Places and Landmarks users. The difference in arm fatigue could also be due to this reason, or it could be that using a WIM causes more arm fatigue. The nausea experienced by users, by contrast, was not significant differently ($F_{1, 18} = 0.093$, $p = 0.764$).

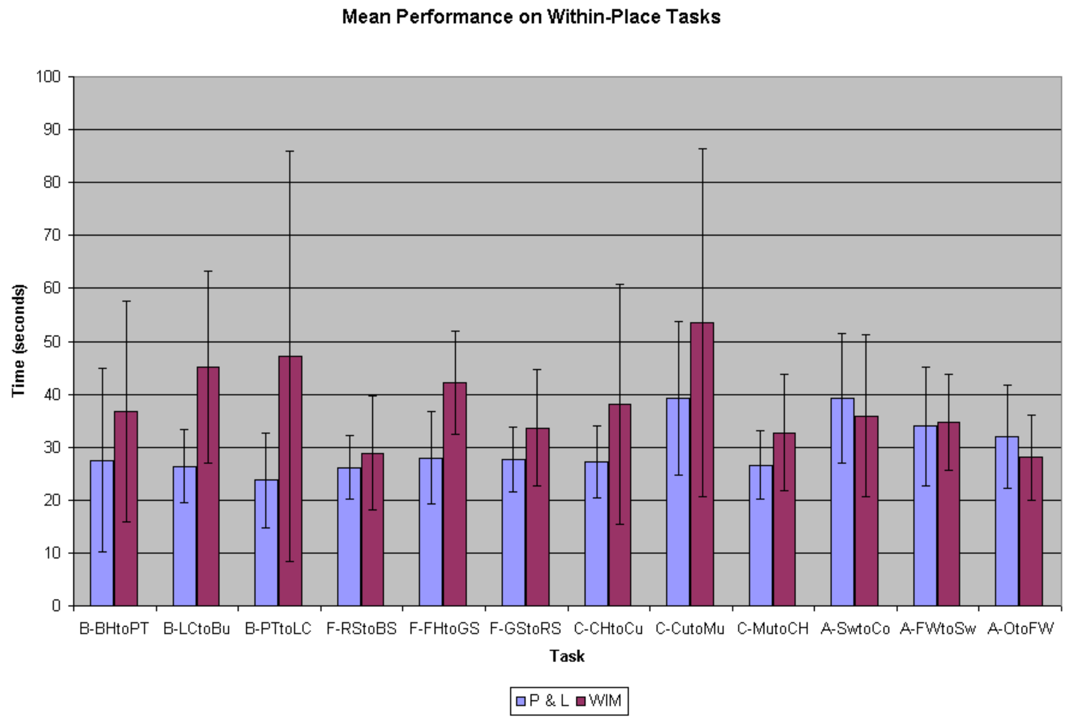

FIGURE 4-21. Mean task performance time for within-place tasks.

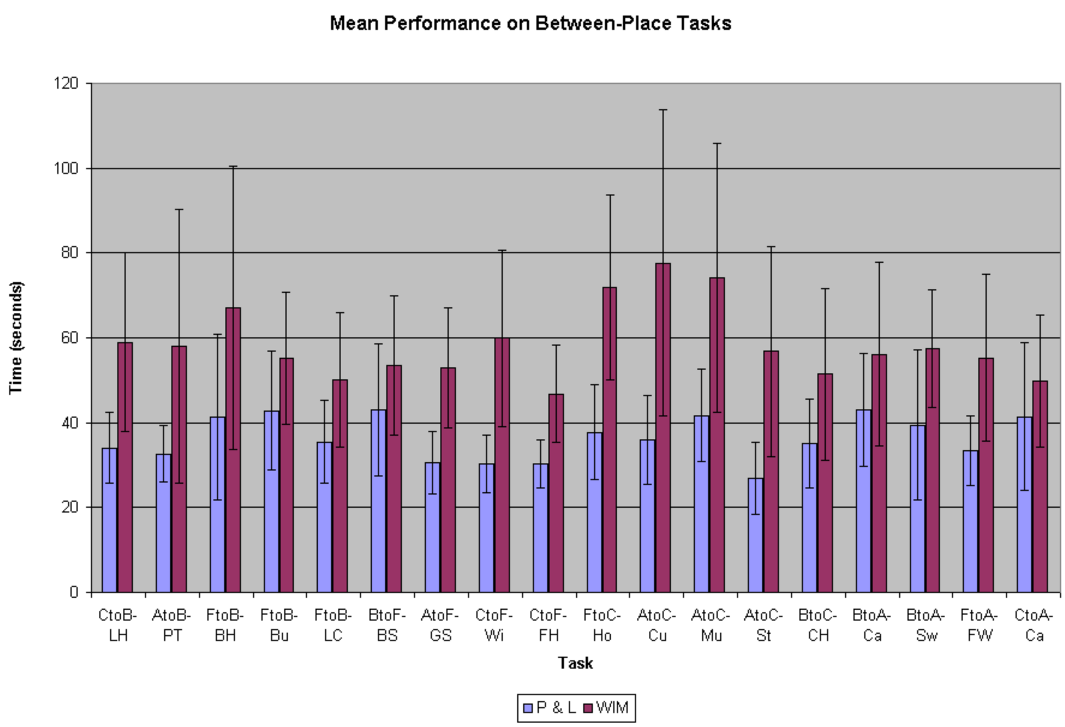

FIGURE 4-22. Mean task performance time for between-place tasks.

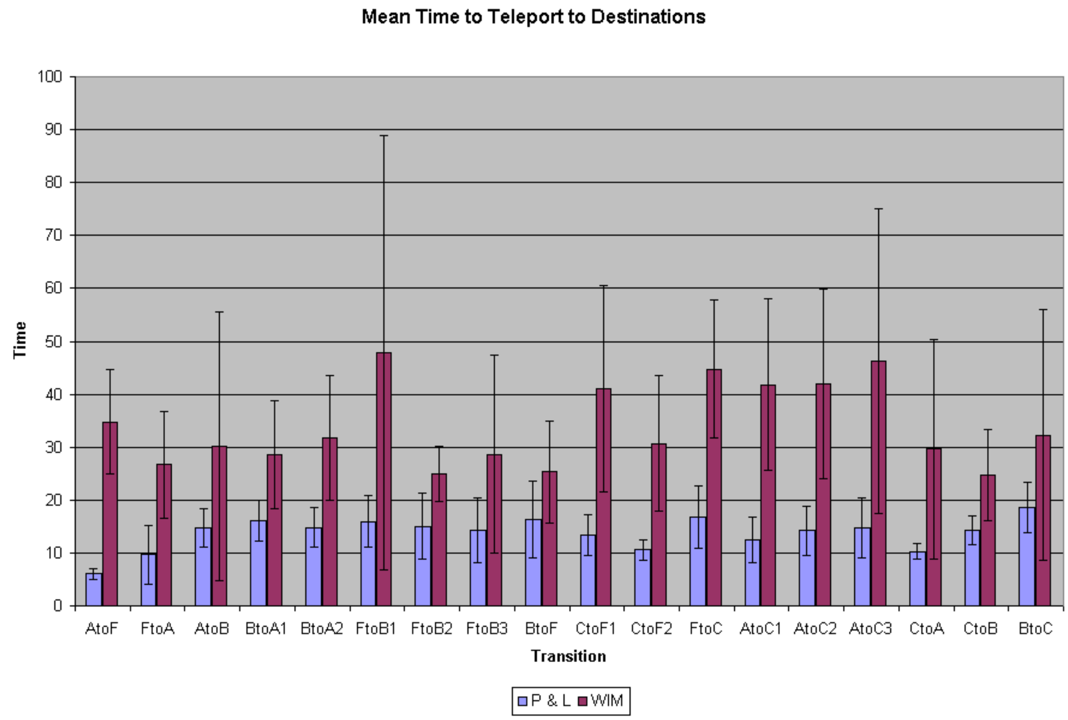

FIGURE 4-23. Mean time to teleport to destination places.

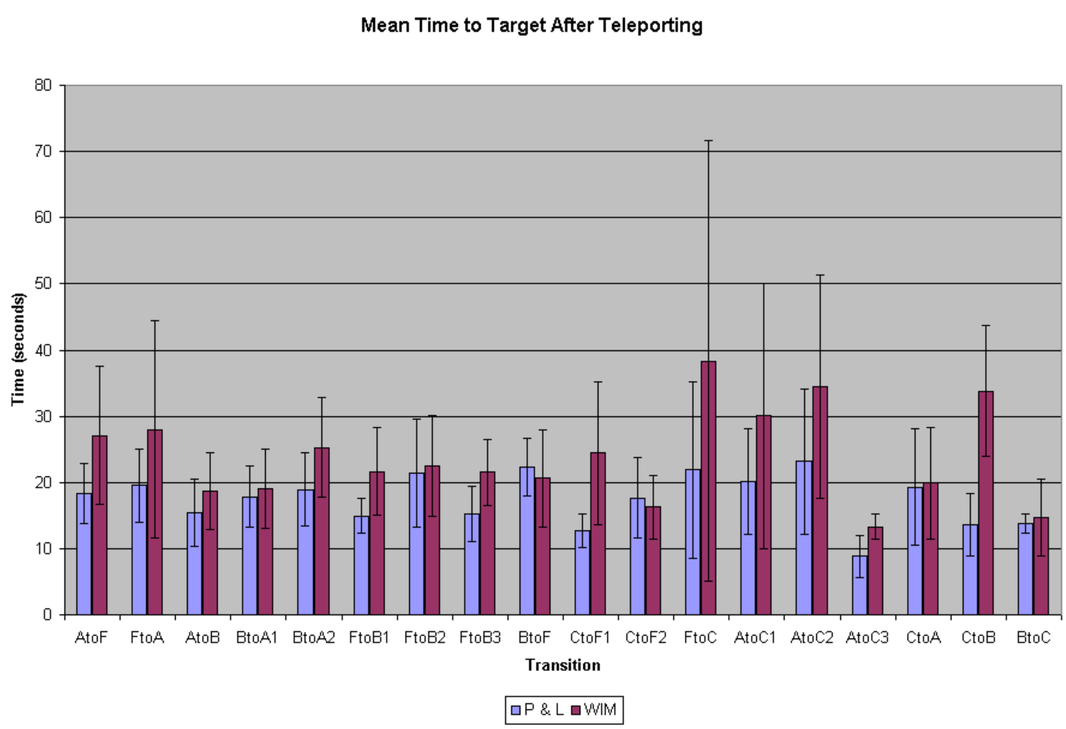

FIGURE 4-24. Mean time to reach target after teleporting to destination place.

4.5.7 Discussion

The shorter task completion times and the smaller standard deviations for Places and Landmarks users on the within-place tasks suggest that those users spent less time searching for the landmarks than the WIM users. I believe that the reason is that navigating with Places and Landmarks fundamentally changes the search task. Instead of searching for the landmarks by navigating through the space, Places and Landmarks users searched by visually scanning the space around them. This type of visual search is easier to carry out because the limited search volume reduces the time to locate a target and makes it easier for users to tell when they have exhausted the search space. Six of the ten Places and landmarks users reported that traveling using the visible landmarks was one of the "easiest three things" about using the technique. By contrast, four of the ten WIM users reported that finding small landmarks in the places was one of the "hardest three things" about using the technique. Visible landmarks could still be improved, however. Four of the ten Places and Landmarks users reported that locating the visible landmarks in the city was one of the "hardest three things" about using the technique. Assigning different halo colors or translucency to different places could make visible landmarks easier to spot in a variety of conditions.

Users of both techniques found image plane navigation easy to use for navigation with places. In the user comments about the techniques nine out of ten of both the WIM and Places and Landmarks users reported that image plane navigation was one of the "easiest three things" about using the respective technique. However, there is still some room for improvement. Two of the WIM users and five of the Places and Landmarks users reported that fine tuning their position was one of the "hardest three things" about using the respective technique. A possible solution might be to replace the constant scale factor of hand motion to user motion with a variable scale factor. For example, the scale factor could depend on the hand's distance from the user's eyes and the selection point. When the hand is near the selection point or the user's eyes the scale factor could be small to facilitate precise positioning, while the scale factor could be large when the hand is farther from the selection point or the user's eyes to permit travel to the selected destination.

The fact that Places and Landmarks users reached destination places faster than WIM users suggests that specifying a destination by choosing from place representations with fixed foci and zoom levels is indeed faster than manually specifying the focus and zoom level. In addition to shorter teleportation times, the standard deviations for the teleportation times were also smaller for Places and Landmarks users. I suspect, based on my observations of users, that the one reason for this difference is that zooming in efficiently with the WIM depends more on manual dexterity and hand-eye coordination: while zooming in users have to maneuver the cursor to keep the zoom focus correctly positioned. While some users were able to simultaneously zoom and update the focus, other users adopted a strategy of repeatedly zooming a little and then adjusting the zoom focus. The process of teleportation itself did not appear to be a problem for users of either technique. Six of the WIM users and five of the Places and Landmarks users reported that teleporting was one of the "easiest three things" about using the respective technique.

Although I did not explicitly measure whether place representations and visible landmarks affected orientation, three of the ten Places and Landmarks users reported that the one of the "easiest three things" about using the technique was that it helped them maintain their orientation in the world. On the other hand, three of ten Places and Landmarks users listed locating the place representations as one of the "hardest three things" about using the technique. Users who were not oriented were forced to visually scan around them to find the appropriate representation. A possible middle ground between helping users maintain orientation and helping them find representations would be to provide a command to temporarily gather the representations in front of the user. Gathering the representations would in essence temporarily create a more traditional fisheye view, although the focus (the current place) would not actually be included. Users could quickly scan the assembled representations, choose the desired representation, and scatter the others back to their original positions. Another potential improvement would be to add halos (with a different color than the halos for visible landmarks) to place representations to make them more visible.

Another improvement to place representations would be to allow users to pass representations from hand to hand in order to adjust their grip before pulling out a child representation. This improvement would address the problem of users' hands colliding when pulling out a representation, which two of ten Places and Landmarks users reported as one of the "hardest three things" about using the technique.

Two of ten Places and Landmarks users reported that the lack of visible landmarks in the actual place representations was one of the "hardest three things" about using the technique. Users could locate landmarks by teleporting into the place and visually scanning for them, but in retrospect presenting visible landmarks in the actual representations would allow users to visually search a smaller area (the representation) to locate a particular landmark and then teleport to a viewpoint near it. I initially chose not to display the visible landmarks in the actual place representations because I was concerned that they would visually crowd the representations. However, by adopting a suitably smaller minimum size for the visible landmarks I believe that I could prevent them from crowding the environment.

A potential drawback to navigating with Places and Landmarks is that users need to physically turn around more than users navigating with a WIM. As a result, users may be more likely to get tangled in the tracker cables. Two of the ten Places and Landmarks users mentioned getting tangled in the cables as one of the "three hardest things" about using the technique. The increased need for turning may also be more awkward for inexperienced VR users, who are sometimes initially reluctant to move their feet.

A common problem with the techniques was a handheld place representation or the WIM occluding the other hand. Four of ten Places and Landmarks users listed this problem as one of "three hardest things" about using the technique, while six out of ten WIM users listed positioning the crosshair to specify the zoom focus. Given that all users had to do to locate the occluded hand was move the holding the WIM or representation away from their head while moving the occluded hand closer, this result suggests that some users do not always have a very good idea of where their hands are relative to each other. A possible solution to this problem would be to make the WIM or handheld representation translucent when the other hand is located beyond it (relative to the user's point of view), and opaque when the other hand is located in front of it.

The report by six out of ten WIM users that positioning the crosshair to specify the zoom focus was one of the "three hardest things" about using the technique was not just due to problems finding their other hand. Some users had trouble learning that the projection of their other hand straight down (or up) onto the WIM specified the zoom focus; they expected to specify the zoom focus by image plane selecting it. I chose to use the hand's projection onto the WIM to determine the zoom focus because users need to be able to specify the zoom focus while zooming in or out, and when users move their hands straight up or down relative to the WIM the projection of their hand onto the WIM does not move. By contrast, if users image plane selected the zoom focus then moving their hand up or down to zoom in or out would also change the zoom focus. In practice I mark the zoom focus by a small red sphere to help users find the projection of their hand onto the WIM. It might be possible to allow users to image plane select the initial zoom focus, and then change the zoom focus based on horizontal motion of their hand rather than continuing to image plane select the zoom focus. However, this would depend on whether or not users switched their attention from their hand to the red sphere once they started zooming. Further experimentation is needed to determine whether or not this is the case.

The desire to always use image plane selection was also a problem for place representations. I implemented proximity selection for choosing the child representation to pull out from a symbolic representation because I conceived it as a physical process of reaching in and pulling out. While most users easily learned this convention, I observed that a few users occasionally tried to image plane select the desired child representation instead of physically reaching in. Adding this functionality would be straightforward, and would not even require replacing proximity selection: the system could check proximity first, and if the user's hand was not near a region for a child representation it could fall back on image plane selection.

Future implementations of panning and zooming WIMs should allow users to choose which direction (up or down) causes the WIM to zoom out, and which causes it to zoom in. While some users found the current mapping very natural, others expected the opposite mapping. Three of ten WIM users reported that zooming in (once they had positioned the zoom focus) was one of the "three easiest things" about using the technique, but two of the ten WIM users reported that the fact that zooming felt reversed was one of the "three hardest things" about using the technique.

I also observed a few situations where users needed better feedback. I observed several situations where a WIM user briefly got confused as to whether they were looking at the world or at the WIM. Although they quickly determined which by waving their hand around, adding a halo to the WIM (similar to the halos I proposed adding to the place representations) could reduce or eliminate those situations. More importantly, it was not always clear to WIM users when they could and could not teleport using the WIM. My implementation of the WIM did not allow teleportation when the WIM was zoomed out too far or when it was zoomed in on an unimplemented area (based on the focus of the WIM and the current scale level). However, it was possible that a small section of an implemented place would be visible in the WIM even though the WIM was focused on another area. In a few cases users tried to teleport using the WIM while it was prohibiting teleportation. Although the users quickly recovered by zooming in further on the desired place and then teleporting, adding visual feedback as to whether or not the WIM was currently permitting teleportation might avoid the problem completely. One possibility would be to change the color of the WIM's halo based on whether or not users could teleport.

I observed a few Places and Landmarks users who tried to image plane navigate into a handheld actual place representation instead of using it to teleport and then image plane navigating. While adding halos to place representations could address this problem, a better solution might be to let users image plane navigate using actual place representations by immediately teleporting the user into the represented place as soon as they tried to image plane select a target.

Another feedback problem with place representations is that users occasionally lost track of how many child representations they had to pull out to reach an actual place representation. One approach would be to color code the halos for place representations so that symbolic representations were one color and actual representations another. A complementary approach would be to always show the child place borders within symbolic representations. I chose to hide these borders by default to avoid cluttering the symbolic representations; a particular border appears only to provide feedback as to which child place representation users will pull out. Instead of hiding and showing the borders to provide feedback, I could show the borders all the time (to provide feedback that users need to pull out another representation) and change the color of the borders to show which child place representation users are selecting.

While navigating with Places and Landmarks was more efficient than navigating with a WIM for my study world, I believe that designers will need to strike an appropriate balance between depth and breadth when planning the place hierarchies of new worlds in order for navigation with Places and Landmarks to be efficient in their worlds. Deeper hierarchies make it easier for designers to reduce the visual clutter because they can display a few places high in the hierarchy that enclose many places low in the hierarchy. Deeper hierarchies, however, increase the amount of time that users will require to reach a location. Only displaying a few places high in the hierarchy also forces users to rely more on their memory because they will have less residue to remind them which child places a particular place contains. Broader hierarchies can reduce the number of steps required to reach a location and reduce the demands on memory, but can also visually overwhelm the user with choices at any particular level. If the place representations are not sufficiently distinctive users will have to take more time studying the place representations to locate the correct representation. To phrase the tension in terms of Furnas' effective view navigation theory, deeper hierarchies do not provide short paths and good residue at every node for every other node, but broader hierarchies do not provide small views and small outlink information.

4.6 Related work

In this section I discuss previous work related to navigation aids, places, symbolic maps, scent or residue, and visible landmarks.

4.6.1 Navigation aids

In addition to the navigation techniques I have already discussed, researchers have developed a number of passive navigation aids. Many of these aids are based on Lynch's five classes of elements in users' cognitive maps of cities: districts, nodes, landmarks, paths, and edges [Lynch]. Darken was one of the first to propose adding these elements to virtual worlds [Darken93]. He added synthetic objects (e.g. poles or flags) to virtual worlds to act as landmarks, and allowed users to create their own simple landmarks by dropping "breadcrumbs" [Darken93]. He added a fixed sun to virtual worlds to act as a global landmark, although unlike place representations this type of global landmark aids geographic (North, South, East, West) orientation instead of spatial orientation. Darken also experimented with imposing radial and square grids on the world to divide the world into districts [Darken96].

Other researchers soon extended these elements to abstract information worlds [Chalmers]. WayMaker, a tool for building simple 3D environments, allows designers to build worlds using Lynch's elements as building blocks [Strohecker]. Other researchers focused on easing the designer's burden by automatically adding these elements to virtual worlds [Ingram], or by automatically constructed worlds organized into a rigid hierarchy of districts (archipelagos, islands, buildings, floors, offices) given an information or service hierarchy [Waterworth].

When incorporating Lynch's elements into virtual worlds, designers seem to assume that these elements should function exactly the same in virtual worlds as they do in the real world. Vinson's guidelines for the design and placement of landmarks are an example of this thinking [Vinson]. His guidelines are based on real world research from urban design, psychology, and geography, but they do not take into account the possibility that landmarks might function differently in virtual worlds. Consider the guideline that designers should choose landmarks of differing size so that a subset of landmarks will be visible at different scales. A better guideline might take into account the fact that landmarks can resize in virtual worlds, so that designers can reduce the number of landmarks users need to learn while still providing recognizable landmarks at many different scales. I believe that designers might be able to apply Lynch's elements more effectively if they consider how these elements function differently in virtual worlds.

Navigation with places and landmarks provides one example of how these elements can play different roles. Landmarks and districts (or places) in virtual worlds can remain visible even when their real world counterparts would be occluded or too small to see. Instead of merely being passive aids that help users structure their cognitive maps and maintain orientation, both of these elements can also actively aid navigation by acting as shortcuts to distant locations. Districts become nodes at larger scales; with place representations nodes are the anchor points for the scent or residue of the contained places in symbolic representations. Paths take the form of discontinuous links between places, either to specific locations (portals) or regions (place representations). Edges, however, take their traditional form. Designers can place additional space around places to make their edges more distinct. Although edges, too, may take on a new form for virtual worlds, I am not yet sure what that form will be.

Other research on navigation aids has explored a classification for aids. Chen proposed five different classes of aids: aids that display the user's current position, aids that display the user's current orientation, aids that logs the user's movements, aids that demonstrate the surrounding environment, and guided navigation systems (e.g. automated tours of the world). The aids provided by places and landmarks fit into many of these categories. Visible landmarks and place representations help communicate the user's current position and orientation, while also demonstrating the surrounding environment. The system could also log users' movements to provide shortcuts to recently visited locations.

4.6.2 Places

Barrus' work explored dividing a virtual world into *locales*: compact chunks with their own coordinate systems that could be rendered and transmitted separately [Barrus]. He was primarily interested in reducing the flow of data for multi-user worlds instead of aiding navigation; locales provided useful boundaries for determining which events in the world to inform users about. In addition to determining what information a user receives based on his current locale, Barrus allowed users to receive information by specifying *beacons:* objects of interest that provide users with their location, available services, etc.

4.6.3 Structuring worlds

A common thread that runs through research on urban planning, wayfinding, and cognitive maps is that designers should structure places with a comprehensible organizational principle so that people can more easily structure their cognitive maps of those places [Canter][Lynch][Passini]. Where places do not clearly present such a principle, designers can impose a simplifying system on top of those places [Parunak]. The map of the London Underground is a famous example that "imposes a simplifying system on an intricate entity" [Canter]. Navigation with places and landmarks adopts this approach. Rather than navigating in the complex virtual environment, users can navigate using the simpler place hierarchy imposed on the virtual environment.

4.6.4 Maps

Numerous researchers have also provided a “you are here” map (occasionally implemented as a second, smaller view of the world [Fukatsu]). Users can employ these maps to help maintain their orientation in the world, and to plan routes to destinations [Darken93]. However, when users can navigate discontinuously, traditional maps become less useful. Maps tend to show more about routes than places [Canter], so for discontinuous navigation techniques I would argue that navigation aids like visible landmarks and place representations are more useful than maps. Maps suffer from other problems: they force users to divide their attention between the world and the map, and they force designers to choose between the drawbacks of “north up” or user-aligned orientations [Levin] for the maps.

Most map navigation aids depict a scaled version of the world, but in the real world symbolic maps are a well established type of map. Passini terms these symbolic maps *fantasy drawings* that attract and amuse while also informing [Passini]. A traditional media example is the guest map of Disneyland. Instead of providing an accurate scale map of Disneyland, the map exaggerates the shape of the park and the proportions of the attractions to draw attention to the objects that guests are most interested in. Computer games are also starting to adopt symbolic maps, although typically in 2D form, to draw attention to targets, waypoints, or other interesting items.

4.6.5 Multi-scale displays

Because he felt that panning and zooming was inadequate (users lose context when they zoom in, and constantly zooming in and out can be tedious), Lieberman created a multi-scale displayed that he dubbed the *macroscope* [Lieberman]. Instead of displaying the world at multiple scales by presenting a fisheye distortion of the world, the macroscope overlays both zoomed in and zoomed out views using multiple translucent layers. Users can control the translucency of individual layers to dynamically change which layers are easiest to see. By providing discrete zoom levels this approach does reduce the need for continuous zooming. However, users still need to pan the macroscope, and adjusting the translucency of the different layers might require just as much time and be just as tedious as zooming.

4.6.6 Scent and residue

While maps typically provide the only scent or residue in 3D worlds, Jul explored methods of providing residue in zoomable or multi-scale user interfaces [Jul], which are somewhat similar to panning and zooming 3D worlds. She restricted her approach, however, by neither assuming nor implying that the spatial layout of the world is meaningful; I assume exactly the opposite. This restriction prevented her from using the structure of the world to reduce the clutter caused by providing residue for all objects. Instead of providing residue for objects, she turned to providing residue for views. She provides residue for views by tracing the outline of the view; as the user zooms out, the outline gets smaller but maintains a minimum size. While, like visible landmarks, her residue maintains a minimum size, she chose to provide anonymous residue for the existence of the objects instead of providing specific residue for individual objects.

4.6.7 Visible landmarks

Visible landmarks are perhaps closest to heads-up displays (HUDs). HUDs, however, typically display symbols instead of the actual objects. This approach can make it more difficult for users to distinguish between landmarks, particularly if multiple objects share the same symbol. HUDs also overlay symbols directly on the user’s view, which unlike visible landmarks means that these symbols can occlude objects in the user’s work volume. The primary difference, however, is that the symbols displayed on HUDs only passively provide information, while users can actively employ visible landmarks to reach distant locations.

4.7 Future work

As an additional step toward providing an illusion of a large compelling world, designers could allow the system to dynamically create actual representations for initially non-existent places. This would allow designers to permit users to pull out actual place representations for any place in a symbolic representation, regardless of whether or not the place currently existed. If a user pulls an actual representation for a non-existent place, the system could randomly generate a place with the needed terrain type (e.g. plain, mountain, forest) on the fly and populate it with randomly placed objects. Once the system had generated a place in this way it could store that place in case the user visited it again. Alternately, designers could create a small number of generic places in advance. When the user subsequently pulled out an actual representation of a non-existent place, the system would assign one of the generic places (and preserve that assignment for the future) to it. The system could also assign landmarks to these places based on users' actions. For example, if users leave objects behind in one of these generic places the system could make a subset of them landmarks, and provide scent for them in the symbolic representations of the enclosing places.

Once a designer has organized a virtual world into a place hierarchy, he could put that hierarchy to more uses than facilitating navigation for users. Another possible use would be to gradually restructure frequently used worlds to make navigation more efficient. The designer could have the system log which places users visit, and in which places they spend the most time. The designer could then have the system use the log data to gradually move frequently used places closer together and infrequently used places to the edges of the world. This restructuring would be easiest for discontinuous worlds, because the system would only need to rearrange the symbolic representations. However, it might also be possible for a continuous virtual world by restructuring the terrain to eliminate or introduce empty space between places. Because users navigating with places and landmarks can travel between close places with a small number of steps, over time restructuring the world would lead to faster travel between popular locations. Rearranging the world could incur costs, however. While experienced users that travel by verbally specifying a destination might not even notice the changes to the world's layout, other users could have difficulty adapting.

The place hierarchy might also be useful in providing a variant of scaling for continuous virtual worlds. Instead of scaling the world uniformly, which treats places and the "empty" space between them as equally important, the system could first shrink the space between places to draw the places closer together. If the user continued to scale the world the system would then start to scale the places themselves. This could facilitate travel between places, particular in worlds with lots of empty space, because users might be able to scale the world to a level where they can see all the places simultaneously while maintaining a sufficiently high level of detail to identify a particular destination.

The place hierarchy could also be useful for assigning different properties to different places. For example, time could pass at different rates in different places, or different places could have different gravitational constants. In these cases the boundaries between places also act as boundaries for the different properties. The system could also use the boundaries between places for render or simulation culling.

In addition to the links between places based on the place hierarchy, the system could dynamically create links between places, for example to provide shortcuts to recently or frequently visited places. While the links based on the place hierarchy should be permanent, the system could assign a duration to the links it creates. These links could then slowly fade away, disappearing when their duration elapses. The disappearance of old links would allow the system to periodically add new links without cluttering the scene. Determining the ideal duration for links could take a little trial and error, however. If the duration is too short the links might fade away too soon, while if the duration is too long the user's view might become cluttered with place representations.

The system could also create temporary landmarks and temporary links between places for other reasons. Consider a system that keeps track of upcoming scheduled events, for example an online sporting event. The system could make the sporting venue a landmark and create links to the place containing the venue from everywhere in the world right before sporting event starts, and then remove the links and the venue's landmark status after the event ends.

The idea of a sporting event attended by many users in a virtual world raises the issue of how navigating with places and landmarks could function in multi-user worlds. Making landmarks and the links between places common to all users would be the simplest approach, but it would eliminate some useful functionality. For example, it would be useful if all the sporting event attendees had their own individual link to where they were before they traveled to the venue. Providing customized links for each user would also allow the system to use place representations for access control in discontinuous worlds. Because users in discontinuous worlds need access to an actual place representation to visit the place, the system could deny a user access to a place by not displaying its place representation to the user. The system could also hide the scent for restricted places in symbolic place representations from users without access rights to try to prevent them from learning that those places exist.

In multi-user worlds the system could also allow a user to designate another user's avatar as a landmark. This would allow the user to keep track of the avatar as it moves around the virtual world, and would provide a shortcut for traveling to the avatar's location at need.

I previously discussed how navigating with places and landmarks can facilitate incremental development of 3D worlds, where a designer uses the symbolic representations to provide broad sketches of the 3D world and adds in detailed locations later as he finishes them. Navigating with places and landmarks can also facilitate incremental exposure to a 3D world. This functionality can be useful for applications like games, where the designer does not want users to be able to access some areas until they have completed certain tasks or visited other areas. To provide this functionality, the system would not provide scent in symbolic representations or display actual representations for particular places until a user had met certain conditions.

Real world maps often overlay different types of information to accommodate different uses. Symbolic place representations could do the same thing in virtual worlds. In addition to providing scent for important places by displaying important buildings, these place representations could display available activities, tourist attractions, population centers, or even industries. Users would simply specify the type of scent they want.

In very large places image plane navigation relative to a landmark may not provide sufficient resolution for users to accurately reach a destination. Users can always solve this problem by choosing a new reference objects: they image plane navigate relative to a landmark to get close to the desired destination, and then select a new object close to the destination to navigate more accurately. Another solution might be to provide a modified version of image plane navigation. Rather than a linear mapping of hand motion to user motion, the mapping could be non-linear. An inverse exponential mapping would provide more accuracy as the user gets closer to the reference accuracy. Another possibility would be to provide a parabolic mapping that scales hand motion more in the middle and less close to the hand's initial position and the user's viewpoint. This mapping would provide reduced accuracy in the middle, where users are more likely to be employing a ballistic motion anyway, and more accuracy near the user's initial position and the selected object.

In my study users always traveled between places by achieving the desired viewpoint of an actual representation and teleporting. As a result, I did not need a mechanism for smoothly animating the transition between local landmarks when the user moved between places. However, for virtual worlds with a continuous layout designers might need to provide such a mechanism because users might move between places using a continuous navigation technique. The mechanism could be fairly straightforward. For example, when the user crosses the boundary between places the system could fade out the old landmarks and fade in the new landmarks. Another possibility would be for the system to provide a boundary zone between places where it displays the landmarks from both places. As a user entered the boundary zone the system would fade in the new set of landmarks, and as the user left the boundary zone the system would fade out the old set of landmarks. In addition to changing the landmarks, the system could also display a signpost along the border between places to communicate the name of the place that users are entering.

Adapting navigation using places and landmarks to desktop virtual reality systems should be fairly straightforward. Instead of image plane navigating to landmarks, users can employ a point-of-interest technique [Mackinlay90] to navigate relative to landmarks. To navigate using place representations, desktop users could left-click on symbolic representations to pull out the representation of a contained place. To navigate with an actual representation, desktop users could use a point-of-interest technique to navigate relative to the representation, and then issue a command to teleport once they had achieved the desired view.

Anecdotal evidence [Murray] suggests that users are reluctant to travel through objects even if the system does not enforce collision detection. For the user study world this was not a problem; users either followed a slightly parabolic route while image plane navigating to a new location or displayed no qualms about traveling through objects. However, for worlds that are very crowded or involve travel into and out of buildings, the system could turn objects in the current place slightly transparent when the user selects a landmark for navigation, and then turn the objects opaque again after the user releases the landmark.

There are a number of remaining research questions about navigation with places and landmarks that could be answered by further formal experiments. These studies could attempt to determine, for example, how this technique affects the development of users' cognitive maps. Do users form cognitive maps faster or more accurately with this technique? Another study could try to determine whether or not this technique provides users with a more compelling experience. Do users feel that they have visited only part of a much larger world? Do they experience a stronger sense of "presence"?

Other formal studies could examine differences between continuous and discontinuous navigation techniques. For example, if users have both types of techniques available, when do they prefer one over the other and why? My suspicion is that the choice is heavily influenced by distance, but there are most likely other factors involved. Users might prefer to walk instead of teleport to the other side of a room and teleport instead of walk to the other side of town, but how would preferences change if users could fly instead of walk?

I focused the discussion in this chapter on virtual worlds designed to match a previously created structure because implementing navigation with places and landmarks is easiest for these worlds. I believe that navigation with places and landmarks would also work, although not as well, for automatically generated worlds with emergent structure. An example of this class of worlds is a 3D visualization of a document collection where the documents are clustered by similarity (e.g. the Galaxy of News [Rennison]). The system could try to group objects into places by computing the density distribution of the 3D objects and using the density contours to assign place boundaries. From there the system would have to organize the places into a hierarchy, using information about the 3D objects (e.g. topics of documents) where possible and falling back on arbitrary rules when necessary. The last step would be to choose objects to act as landmarks. In a world full of different types of objects the system could try to choose large, unique objects that are spatially distributed about a place (so that it does not choose objects all in one place). If all the objects in the world are all fairly similar in size and shape (e.g. documents), the system would have to rely on other information about the objects. In a document collection, the system could create landmarks by adding signposts with keywords contained in nearby documents on them. Alternately, a designer could provide the system with a library of 3D objects, and the system could try to match the keywords in a cluster of documents to an object in the library. For example, the system could add a fish as a landmark for stories about fishing.

The chief limitation of navigating with places and landmarks is the extra work required to create the place hierarchy, build the semantic structure into the virtual world, designate objects as landmarks, and build links between places. I believe that the payoff for this hard work, more efficient user navigation, is well worth the cost. While automatic processes could perhaps handle some of this work for the designer (e.g. automatically choosing landmarks based on heuristics), the reduced workload for designers could result in an increased workload for users because of non-optimal choices by the system (e.g. the choice of inappropriate objects as landmarks). In the end, designers themselves will have to choose how much of the burden the wish to shift off of themselves and on to their users.

4.8 Summary

By breaking the assumptions that distance objects are occluded by nearby objects, that objects get smaller and smaller as they get farther away until they are too small to see, that all objects get smaller at the same rate, that objects' appearance remains constant (except for size) at all distances, and that the abstract concept of place is not represented by a concrete object I created a technique for navigating with place representations and visible landmarks. In a formal experimental study I determined that this technique allows users to navigate more efficiently than a panning and zooming WIM, a former best practice navigation technique. On average Places and Landmarks users completed the within-place tasks 21.67% faster (29.813 seconds vs. 38.063 seconds) and the between-place tasks 37.77% faster (36.403 seconds vs. 58.495 seconds). Navigation with Places and Landmarks benefits by overcoming some of the shortcomings of existing navigation techniques and aids.

Navigating with places and landmarks allows users to quickly traverse small distances with a single gesture. Users can quickly travel short distances by image plane navigating relative to a visible landmark, even if the landmark object would normally be occluded.

Navigating with places and landmarks allows users to quickly traverse larger distances with a few discrete choices. Users can quickly travel larger distances by selecting the place representation containing the desired location, narrowing in on the desired location with a few discrete choices, and then teleporting to the location.

Navigating with places and landmarks helps users maintain their orientation. The presence of visible landmarks and place representations helps users determine where they are relative to other places in the virtual world.

Navigating with places and landmarks scales smoothly from a world the size of a small town to a world the size of planet. Organizing the world into a place hierarchy allows the system to avoid visually overwhelming users by only presenting users with a small number of place representations at any one time. However, even with this small number of visible place representations users can travel anywhere in the world with a small number of discrete choices. Designers can increase the world's size by a semantic order of magnitude by adding one more level to the place hierarchy, which requires users to make at most one more choice when navigating.

Navigating with places and landmarks allows designers to present the illusion of a continuous world even if the designer has only implemented a few discrete places. Designers can present the illusion of a large, compelling world by showing users large and detailed symbolic representations of places. Designers do not have to worry about users trying to navigate into those places, because users can only travel to places with actual place representations.

We shape out tools, and thereafter our tools shape us.
Marshall McLuhan

It is not reason that is the guide of life, but custom.
David Hume

CHAPTER 5 *Interaction Surfaces*

In order to provide a compelling experience in a virtual world, we need to create techniques that allow users to move around the world and manipulate the objects that they encounter. But before users can interact in virtual worlds, we first need to create those worlds. In recent years researchers have worked to simplify the process of building interactive 3D worlds by creating tools that allow users to focus on their particular task instead of its underlying implementation. Modelers can now create a 3D model by sketching it [Igarashi99][Zeleznik] instead of explicitly specifying its polygonal mesh, and artists can paint a texture on a model without explicitly specifying the underlying UV mapping [DeepPaint][Igarashi01]. Users can avoid traditional modeling and painting altogether by taking pictures of an existing object and constructing an image-based 3D model [Debevec] from them. Animators can make a model move and turn without manipulating 4x4 homogeneous matrices [Conway][Cult3D][WorldUp]. These tools empower novices and make (expensive) experts more productive.

Unfortunately, the crucial step of defining how models respond to user actions remains a weak link in the process of building an interactive 3D world. In current authoring tools interaction designers assign responses to groups of a model's polygons: when a particular event happens to a polygon in a particular group, the model responds in a particular way. The problem with this method is that it ties a model's interaction semantics to its geometric structure. As a result, an interaction designer cannot assign a response to a part of the model that does not exactly correspond to a pre-defined group of polygons.

To illustrate this problem, consider constructing a 3D e-commerce application using the MP3 player model in Figure 5-1. We would like to specify that the model plays a song when a user left-clicks on the "play" button, but the model's polygons are unstructured because I created it using a photogrammetric method (e.g. [PhotoModeler] or [ShapeCapture]). We cannot simply group a subset of the model's polygons because they do not provide enough resolution. We must first restructure the existing polygons to create a set that exactly matches the shape and position of the button, update the model's UV mapping so that the texture still projects correctly, and then explicitly group the polygons. This process is time consuming even for experienced artists and modelers. Designer without recourse to these professionals will most likely be unable to accomplish the task at all.

FIGURE 5-1: A 3D photogrammetric model of an MP3 player with (left) and without (right) its texture. Note that the buttons are not reflected in the model's geometry.

This problem is not restricted to photogrammetric models. The current method of allowing designers to only assign responses to groups of polygons makes several assumptions: that a modeler explicitly creates and groups the polygons, that the modeler can foresee what parts of the model the designer will want users to interact with, and that a model's interaction semantics will not change during its lifetime. The first assumption fails when a tool, rather than a modeler, creates and groups the polygons. Example tools include CSG and procedural modelers [Igarashi99][Thibault][Snyder][Zeleznik], photogrammetric modelers [Debevec][PhotoModeler][ShapeCapture], and 3D scanners [Bertram][Cyberware]. The second assumption fails when an artist paints details into the model's texture that are not reflected in its polygonal structure, or when the designer downloads a model from the Internet (e.g. [Viewpoint]). The third assumption fails when an artist modifies a model's texture, for example to use a single model of a human as multiple characters in a 3D game.

5.1 Designing Interaction Surfaces

An ideal solution to this problem would allow interaction designers to easily and flexibly specify what parts of the model can respond independently to events; I call these parts *interaction surfaces*. Designers have the best grasp on how they want users to interact with models, and a tool that allowed them to easily create and modify interaction surfaces would facilitate rapid prototyping of interactive behaviors. The current solution, modifying the model's polygonal mesh, is not viable because it is labor intensive and inflexible. We need to break the assumption from existing practice that designers must assign interaction responses to groups of polygons. We need a process that makes interaction surfaces easy to create, easy to use, and easy to modify.

My solution is to allow a designer to specify an interaction surface by painting it onto the 3D model. The designer can specify multiple interaction surfaces by painting each a different color. The system captures and stores the painted areas by projecting them onto a 2D *interaction map*. An interaction map is similar to a traditional texture, but its painted regions specify interaction surfaces instead of appearance. When the designer finishes painting the system creates an interaction surface for each distinct color in the interaction map. Designers can name these surfaces and assign them responses to different events. To create the interactive behavior in our MP3 player example, a designer would paint over the "play" button, name the resulting interaction surface PlayButton, and then specify that PlayButton responds to left-click events by playing a song. The system also automatically creates a modified version of the model's texture, based on the locations of the interaction surfaces in the interaction map, which designers can apply to the model to help end users locate the interaction surfaces.

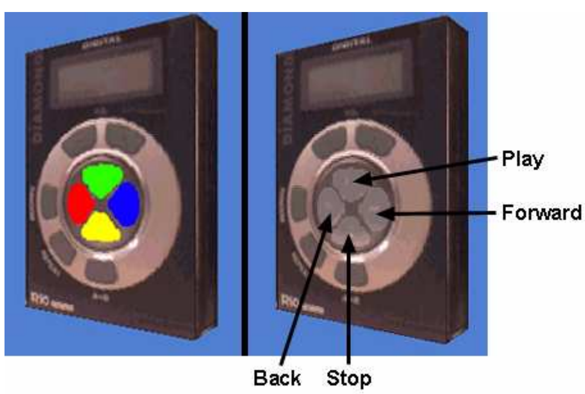

FIGURE 5-2: To allow users to interact with the buttons the designer paints interaction surfaces for the play, forward, back, and stop buttons (left). The designer also creates a modified version of the model's texture (right, compare with Figure 5-1) to help end users locate the interaction surfaces.

I believe that interaction surfaces may be particularly useful for rapidly prototyping interaction for Web-based 3D objects. To allow users to interact with a 3D model of a product for sale, a designer could create a photogrammetric model and paint in the desired interaction surfaces (Figure 5-3). Interaction surfaces also allow designers to quickly prototype different control layouts for products under development by modifying a model's texture and its interaction surfaces. Interaction surfaces can also simplify the process of creating a new 3D model, because modelers do not have to worry about how end users will interact with a model.

FIGURE 5-3: Designers can paint interaction maps for both photogrammetric models (left) and hand-painted models (right) to allow users to interact with parts of the models only present in their textures.

5.2 Related Work

Allowing users to interact with sections of a 2D object is a well established idea. Image maps [HTML] allow web pages to load different URLs when users click on different sections of a 2D image. While image maps only support circular or polygonal "active areas", QuickTime VR [QuickTime] allows designers to paint irregularly-shaped active areas onto a 2D panoramic image. In addition to extending the idea of active areas from 2D to 3D, I allow non-contiguous active areas, and I allow active areas to pass parameters to their responses. I also allow designers to change the size and shape of active areas, as well as their associated responses, at run-time.

My technique draws on the idea of using false color as a per polygon or object ID. Traditional uses of false color as an ID include picking [OpenGL] and finding intersections for ray-tracing using an item buffer [Weghorst]. I use images mapped to the model (interaction maps) to extend this idea to allow more than one ID (interaction surface) per polygon. I associate responses with these texture-based IDs to govern how sections of a 3D model respond to events.

I discussed this technique with video game developers, who often use textures in novel (and unpublished) ways, but found that they typically constrain the development process so that modelers know all interaction surfaces in advance. Researchers and game developers do use textures for more than affecting the visual appearance of a 3D model [Haeberli][Heckbert]. Textures for 3D terrain can designate "off limits" areas, paths for computer controlled characters, and the type and density of flora [Deussen][Martin]. Artists can also paint "attribute maps" [Softimage][Maya] onto 3D models that contain different types of data, for example fur, vertex set membership, and weights for vertex animation.

5.3 Interaction Surfaces In Detail

This section describes the steps that designers take in creating and assigning responses interaction surfaces. I will mainly discuss these steps in the context of the implementation I created for the Java version of the Alice authoring tool [JAlice] and evaluated with a short user study. For many of these steps I experimented with alternate implementations, and where appropriate I will discuss the strengths and weaknesses of these alternatives.

5.3.1 Painting an Interaction Map

A designer specifies interaction surfaces (Figure 5-4) by painting an interaction map. The simplest method of creating an interaction map is to load the model's texture image into a 2D painting program that provides layers (e.g. [Photoshop]). The designer displays the texture image in one layer while painting over it in another. After he finishes painting, the designer saves the painted layer separately as the interaction map (Figure 5). Although this method is simple and allows the designer to use 2D paint tools, designers may have difficulty determining which section of the texture image corresponds to the desired part of the 3D object. For example, in the texture map on the right of Figure 5-6, the dog's face is fairly easy to locate, but his body, legs, and tail are not.

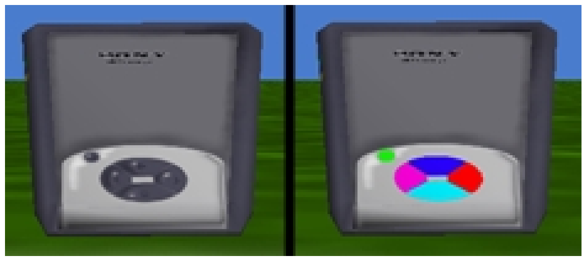

FIGURE 5-4: A CD player model (left) and the desired interaction surfaces shown superimposed over the model's texture.

FIGURE 5-5: The CD player's texture (left), its interaction map painted over its texture (middle), and the extracted interaction map (right).

FIGURE 5-6: The dog's body, legs, and tail are difficult to locate in the texture.

Using a 3D painting tool that provides layers (e.g. [DeepPaint]), designers can instead paint the interaction map directly on the 3D model [Hanrahan][Igarashi01]. The system displays the 3D model with its texture applied and the designer paints into a separate layer that he saves as the model's interaction map.

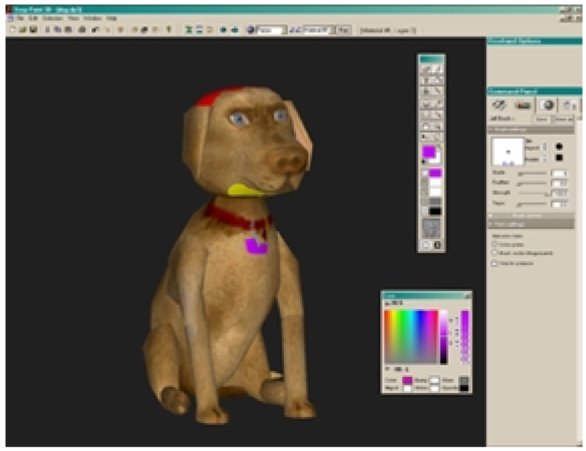

FIGURE 5-7: Designers can trace the interaction map over the texture map directly on a 3D object using a 3D painting tool that provides layers. Here the designer has painted surfaces on top of the dog's head and under the dog's chin, and is currently painting a surface over the dog's collar.

Creating hand-painted textures is a time consuming process even for an experienced artist. Creating a hand painted interaction map is a much simpler process because it avoids the hard parts of creating a texture: specifying a useful UV mapping (which requires time and experience) and painting details (which requires artistic talent). The interaction map can use the texture's UV mapping, and designers can trace over details instead of painting them. As a result anyone, regardless of artistic ability, can paint an interaction map.

There are two situations where the interaction map will not be able to reuse the texture's UV mapping. First, the model may be untextured and lack UV coordinates. Second, sections of a 3D object that are identical (e.g. eyes) or mirror images of each other (e.g. left and right arms) often share the same section of texture. The latter case prevents the designer from creating a different interaction surface for each section. In both situations the designer can create a new UV mapping specifically for the interaction map. The designer can still avoid explicitly specifying the UV mapping by using a tool (e.g. Chameleon [Igarashi01]) that dynamically manages the UV mapping while he paints. Automatically generated UV mappings are often non-optimal for complex hand-painted textures, but are sufficient for our purposes.

5.3.1.1 Modifications to Paint Tools

Although the designer can use an existing 2D or 3D paint tool to create a model's interaction map, a few additional modifications would improve the suitability of these tools for painting interaction maps. Existing tools either blend the new paint color with the existing paint color, or replace it entirely. For painting interaction maps, brushes should replace the existing color instead of blending if the current paint color is a different hue, while increasing the saturation of the existing color if the hues match. In addition, paint brushes could take new parameters. Rather than changing the hue and saturation of a brush, the user should be able to directly set the response and response strength. Users should also be able to create brushes that paint with a constant hue and possess both a pixel radius for maximum saturation and a fall off function that specifies how the saturation decreases as the distance from the center of the brush increases.

5.3.2 Creating Interaction Surfaces From An Interaction Map

After painting the interaction map the designer needs to associate it with a 3D model in his 3D authoring tool to create the specified interaction surfaces. In my implementation the designer selects a model's "interaction maps" property to display the interaction surface viewer. This viewer allows designers to load interaction maps and name the resulting surfaces.

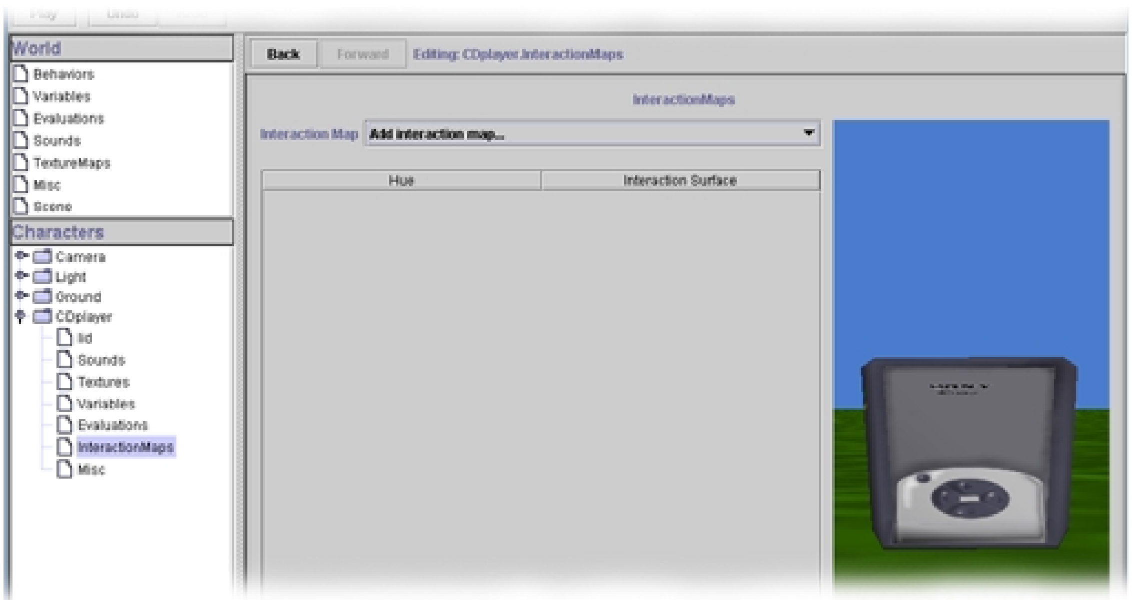

FIGURE 5-8: The interaction surface viewer for the CD player. This viewer allows designers to load interaction maps and edit interaction surfaces.

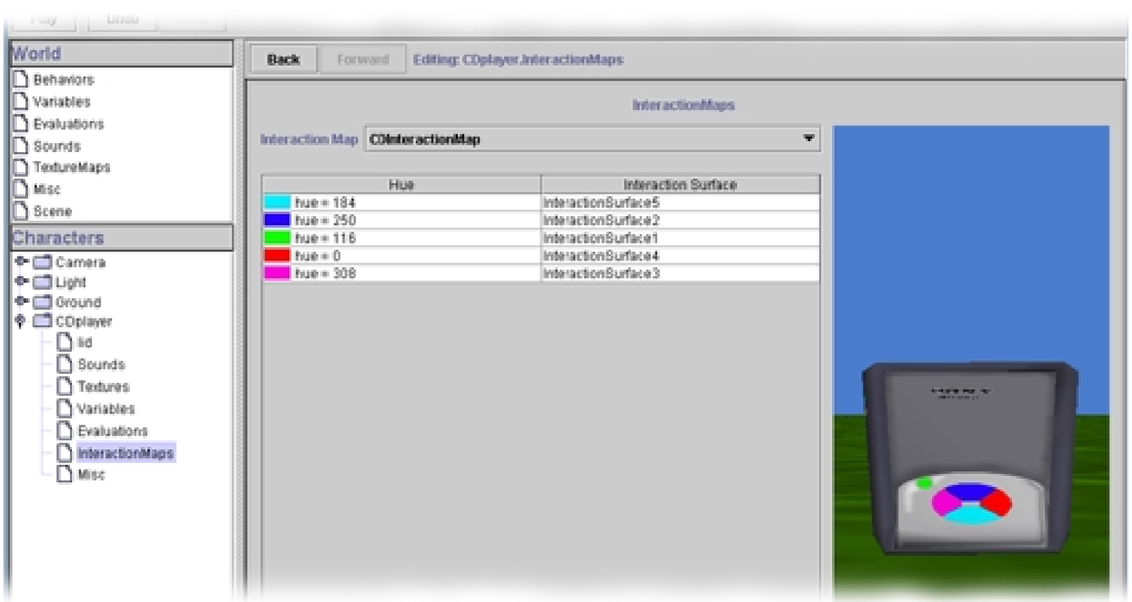

FIGURE 5-9: When the user loads an interaction map the system creates the interaction surfaces, assigns them temporary names, and creates a temporary texture to display the surfaces in context on the model.

My implementation creates an interaction surface for each unique hue in a loaded interaction map. Since image formats typically store pixel colors in RGB format, the system temporarily converts each pixel color from RGB to HSB to build a list of unique hues. To avoid problems with small conversion errors due to limited precision, the system scales the hue value so that it ranges from 0 to 360 and rounds it to the nearest integer. This allows designers to create 360 different interaction surfaces for each model (0 and 360 are the same hue).

When a designer loads an interaction map my implementation also creates a temporary texture image that overlays the interaction surfaces on top of the model's existing texture to make it easier for designers to locate the interaction surfaces on the model (Figure 5-9).

5.3.2.1 Alternate Implementation: Other Approaches to Identifying Surfaces

Using hues to identify interaction surfaces allows designers to create interaction surfaces that are not contiguous. Geographically distinct sections of the model can belong to the same interaction surface, even if those sections are spread across multiple polygon groups. However, there are other approaches that would also allow designers to create non-contiguous surfaces. Instead of using a single channel (hue) to identify a surface, I could simply have each unique color identify a surface. This would allow a much larger number of interaction surfaces at the cost of restricting their power: by using only a single channel (hue) of a pixel's color, I allow designers to store additional information in a pixel's color saturation and brightness (see section 5.1).

I could also have chosen to use the red (or green or blue) channel of the more common RGB encoding to identify surfaces instead of using hue and the more obscure HSB encoding. The primary advantage of using hue is perceptual. Using hue to identify surfaces provides a wider number of perceptually different identifiers than using red (or blue, or green), as can be seen in Figure 5-10. In addition, when the color's saturation is used to store additional information the resulting colors are perceptually easy to interpret as a single surface that is "fading away" (Figure 5-11). By contrast, when the green channel is used to store information, the colors shift from red to orange to yellow and appear perceptually to be unrelated surfaces.

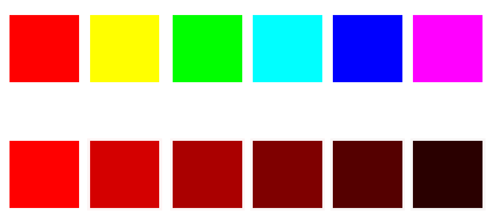

FIGURE 5-10: Color tiles with evenly distributed hue (top) are visually more distinct than color tiles with evenly distributed red (bottom) values.

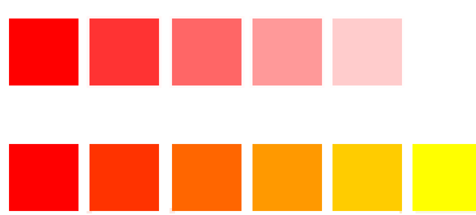

FIGURE 5-11: Color tiles with constant hue and equally distributed saturation (top) appear to be a single color that fades away, while color tiles with constant red values and equally distributed green values (bottom) appear to be different colors.

5.3.3 Naming Interaction Surfaces

Instead of a table that maps hues directly to responses, my implementation creates a table that maps hues to named instances of class InteractionSurface. Representing interaction surfaces as instances rather than as a simple lookup table makes them more flexible: interaction surfaces can inherit responses, respond differently to different types of events, and respond in multiple ways to a single event type. However, I believe that the most compelling argument for using named instances is that they are easier for designers to use. For example, remembering that the PlaySong response should be assigned to the PlayButton interaction surface is much easier than remembering that it should be assigned to hue 250. Designers can also use the names in a human-readable scripting language, for example when changing response mappings dynamically.

The interaction surface viewer assigns interaction surfaces temporary names when it creates them, and then allows designers to change the names. To rename a surface, the designer selects it in the viewer (Figure 5-12). The system creates a temporary texture that overlays the model's original texture image with just the selected interaction surface to help designers distinguish between perceptually similar hues using their location on the model.

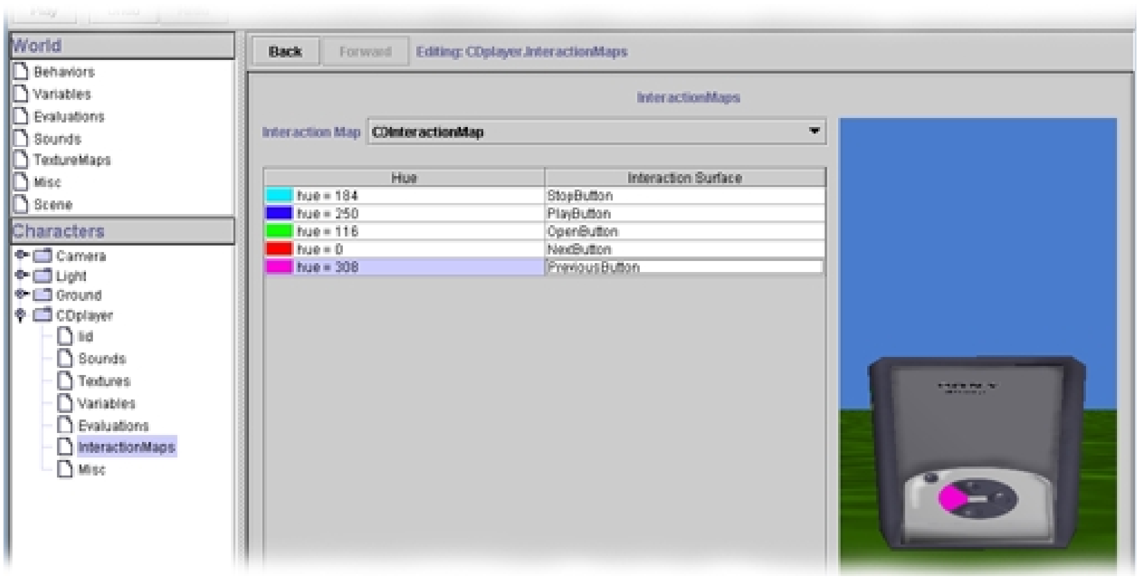

FIGURE 5-12: Designers can rename an interaction surface by selecting it and typing a new name. The system helps designers identify the surface by creating a texture that displays only that interaction surface in context.

5.3.3.1 Alternate Implementation: Sharing Surfaces

My implementation of interaction surfaces does not allow designers to map multiple hues to the same interaction surface. This capability could be useful when the designer wants to assign the same functionality to multiple interaction surfaces. Instead of creating multiple surfaces and assigning their responses separately, the designer shares a single interaction surface between models. For example, if a 3D world contained multiple CD players a designer could use the same PlayButton surface for all of them to simplify modifying their responses. The primary drawback to this alternative is the added difficulty of communicating to designers which models share a particular surface.

5.3.4 Assigning Responses to Surfaces

In the my implementation designers use the same process for assigning responses to interaction surfaces that they do for assigning responses to polygon groups (Figure 5-13). I allow designers to assign multiple responses (to the same type of event or to different types) to the same surface. For example, the designer could specify that the CD player both stops playing and opens its lid when an end user left-clicks on the OpenButton surface, but merely skips a few seconds in the current song if another 3D object in the scene collides with it.

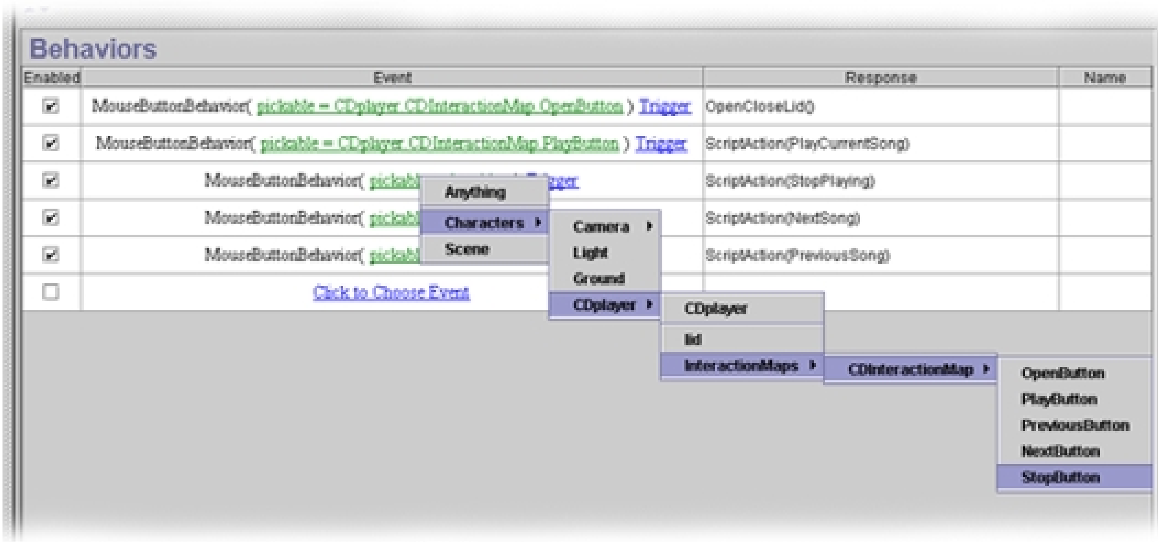

FIGURE 5-13: The designer associates the StopPlaying response with the StopButton interaction surface.

5.3.4.1 Alternate Implementation: Inheriting Default Responses

Interaction surfaces can inherit and provide default responses to and from each other, as well as to and from a model's groups of polygons. This allows interaction surfaces to work in combination with the traditional method of assigning responses to groups of polygons. For example, returning for a moment to the model of the dog in Figure 5-6: if the dog's Head is a pre-defined group of polygons the designer might create a Chin interaction surface but decide that he wants that surface to use the Head's responses to events until he explicitly assigns the Chin a response. I experimented with providing this functionality by allowing designers to insert a model's named interaction surfaces into its *response tree* (Figure 5-14).

Traditionally a model's response tree arranges its parts into a tree structure that is identical to the model's geometric structure. A part in the tree inherits responses from its parent, and provides default responses to its children. A part uses its inherited response to a particular event only when the designer has not explicitly assigned it a response to that event. Allowing a model's response tree to contain both groups of polygons and interaction surfaces provides designers with more flexibility in defining how models respond to user actions.

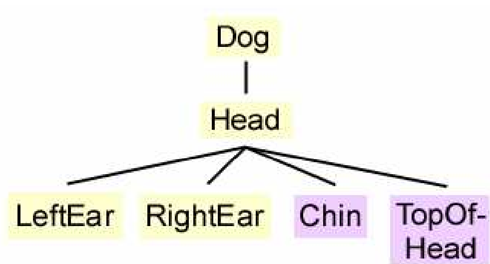

FIGURE 5-14: A section of the dog's response tree showing some of the dog's polygon groups (yellow) and interaction surfaces (purple).

Occasionally an interaction surface might want to inherit default responses from multiple objects, particularly when the interaction surface is distributed across several of the model's component parts. For example, a ticklish interaction surface for a character might be spread across the character's left arm, right arm, and body. For events other than left-clicking the designer might want the part the surface actually lies on to handle the event. Thus, for example, the right arm would handle right-clicks that hit the interaction surface on the right arm, while the left arm would handle right-clicks that hit the interaction surface on the left arm. Designers could choose this type of inheritance by not setting the parent of the interaction surface (i.e. not inserting the surface into the model's response tree). At run-time the system would then pass events to the polygon group behind the surface (as if the surface was not there) if the surface does not explicitly handle those events.

Although my earlier prototype implementations supported the insertion of interaction surfaces into the response tree, my current implementation does not because the Java version of Alice lacks a mechanism for building a response tree independent of the scene graph.

5.3.5 Setting the Current Interaction Map

Because designers can associate multiple interaction maps with a single model, a model can have multiple sets of interaction surfaces assigned to it. My implementation allows users to determine which set of interaction surfaces is active by setting the active interaction map for a model. Only interaction surfaces in a currently active interaction map will respond to events.

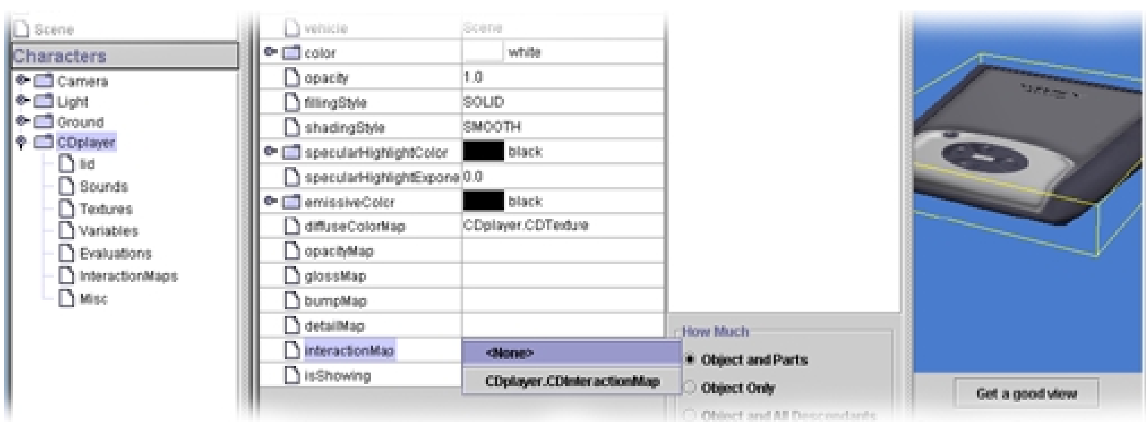

FIGURE 5-15: The designer can make any interaction map assigned to an object its active interaction map.

5.3.6 Run-time Implementation

My implementation retains the interaction map and uses it to route events to the correct interaction surface at run-time. This implementation allows designers to make a 3D model respond to any event (e.g. mouse motion, mouse button clicks, raycasting in immersive worlds, collisions) that yields the polygon and offset within the polygon where the event occurred. The system requires this information in order to determine which surface, if any, is affected by the event. I will use left-clicking with the mouse as a sample event to explain this implementation.

When the user positions the mouse cursor over a model and left-clicks, the system performs a pick into the scene to determine the polygon group, the polygon, and the offset within the polygon where the user clicked. The system then passes that information to each pickable object in the scene. Each object informs the system whether or not it was hit by the pick. When the system asks an interaction surface, the surface retrieves the UV coordinates for the vertices of that polygon and uses barycentric coordinates [Hocking] to calculate the UV coordinates for the offset position. The surface then looks up the color at that position in the active interaction map. If the color is white (because either the hue or the saturation is 0), the surface reports that it was not hit. If the color's hue maps to that surface in the active interaction map, the surface reports that it was hit.

5.3.6.1 Alternate Implementation: Picking

To determine whether or not a particular surface was clicked on, my implementation needs to perform a pick (which requires re-rendering the scene), calculate the polygon offset and UV coordinates for the offset, and look up those coordinates in the interaction map. Instead of performing the additional calculations, the system could identify the picked surface using only the rendering step. This would require modifying the interaction maps so that, while the hue identifies surfaces and the saturation carries designer-specified information, the brightness contains an object ID. To render for a pick, the system would swap every model's texture image with its active interaction map, set the ambient lighting to the maximum, and render the current scene off-screen without anti-aliasing. To determine which interaction surface the user clicked on, the system would look up the pixel in the rendered image corresponding to the mouse pointer position to determine what object and hue the user clicked on. The system would then look at that object's active interaction map and map the hue to the correct interaction surface. Finally, the system would reset every model's texture and the ambient light level.

The advantage to this approach is that the system can make full use of any available rendering hardware to determine the surface the user clicked on. There are two main disadvantages. First, using brightness to store the object ID restricts the number of objects that the system can display because brightness only provides 8 bits. However, only objects with an active interaction map require an object ID, so in practice this only restricts the system to rendering at most 255 objects with active interaction maps. The second, larger disadvantage is that the rendering hardware's texture memory will need to store both the interaction maps and texture images. If the interaction maps and texture images do not both fit in video memory, swapping between them slow down rendering. By contrast, my implementation only needs the interaction maps to be in main memory, and in fact will only load into memory those interaction maps whose surfaces possess designer-assigned responses.

5.3.6.2 Alternate Implementation: Internal Representation

The main advantages to retaining the interaction maps as images are ease of modification and fact access to pixel colors. Because the system keeps the interaction maps, designers can load them back into paint programs and quickly modify them to modify a model's interaction surfaces. In addition, at run time the system can access any pixel in constant time. However, this approach can impose a significant memory cost for high resolution interaction maps.

Designers can lower this cost by using a lower resolution image for the interaction map. For larger savings, I could replace the pixel colors with identifiers for the corresponding interaction surfaces and store the resulting map internally in a more efficient structure, such as a quad tree or run length encoding [Samet]. While either encoding can significantly reduce the memory cost, these encodings make it more difficult for designers to modify the interaction surfaces and require more time to access pixel colors. In order to modify the interaction surfaces, the system would need a process to regenerate the interaction map from the encoding before the designer can paint it. Instead of constant time access to a particular pixel, the access time would increase logarithmically with interaction map size for the quad tree encoding, and could increase linearly (in the worst case) with interaction map size for the run length encoding.

Another approach, suggested by John Hughes at Brown University is to steal the low two or three texture bits if the number of interaction surfaces for a given model is sufficiently small and avoid maintaining a separate structure altogether. This would eliminate the additional memory overhead, but would preclude some of the advanced uses of interaction maps, such as overlapping interaction surfaces and encoding parameters to response functions.

5.3.7 Providing Feedback

While the designer can display a model with its original texture to force users to explore the model to locate its interaction surfaces, in many cases the designer might want to provide users with feedback about the locations of a model's surfaces. Visual feedback (Figure 5-16) is the easiest to provide. The designer can create a feedback texture for the model in advance, either by hand-painting or automatically generating it. For the former the designer could overlay the model's interaction map on its texture map, and then trace outlines around the interaction surfaces in the texture map. For the latter, the designer can use the feedback texture that my implementation generates. When the designer loads a new interaction map the system examines both the current texture image and the interaction map and generates a feedback texture by decreasing the saturation of the texture pixels that correspond to interaction surfaces. The designer could replace the model's original texture with the feedback texture completely, on request, or when the user mouses over that model.

FIGURE 5-16: The MP3 player (left) with feedback textures showing the outlines of interaction surfaces (middle) and lighter areas corresponding to the interaction surfaces (right).

Designers can also provide feedback by making the 3D model respond when the mouse pointer moves over an interaction surface by assigning a response to mouse-over events for that surface. For example, when an end user moves the mouse over a character's ticklish interaction surface, the character could tell the user not to tickle him and raise his arms as if to ward off the user. Alternately, designers can borrow an idea from 2D applications and assign a response to mouse-over events that changes the mouse cursor's shape when it is over an interaction surface (e.g. from an arrow to a hand). By adding responses to both mouse-over and mouse-move events, designers can even provide tooltips that pop up when the user moves the mouse cursor over an interaction surface and pauses.

5.3.7.1 Alternate Implementation: Other Types of Auto-Generated Feedback

There are a variety of other types of feedback that the system could auto-generate. For instance, instead of creating a feedback texture by modifying the color saturation of the texture pixels, the system could instead create a height map for the model. By assigning a small, non-zero value to every pixel in the height map that corresponds to an interaction surface in the interaction map, the system could slightly raise those surfaces when it displays the model to the end user.

5.3.7.2 Alternate Implementation: Multi-texturing for Feedback

Instead of creating the feedback texture in advance, the designer could instead compose the interaction map and original texture dynamically using multi-texturing. In this case the choice of function for blending the two images determines the feedback provided to the user. The blend function could, for example, overlay the interaction map on the texture or modify the saturation of the texture pixels. Multi-texturing introduces an additional run-time cost because of the processing power required to blend the images each frame, but it also provides designers with more flexibility for displaying feedback. In addition to displaying all the interaction surfaces either all the time or on user request, the system could instead only provide feedback for the interaction surface that mouse is currently over, or it could provide feedback only for those interaction surfaces that currently have responses assigned to them. My implementation does not provide this type of feedback because the Java version of Alice does not currently support multi-texturing.

5.4 Advanced Applications for Interaction Surfaces

In this section I describe how users can pass parameters to an interaction surface's response functions, modify a model's interaction surfaces at run-time, and create overlapping interaction surfaces.

5.4.1 Passing Parameters to Surfaces

Because the system only uses a color's hue to uniquely identify an interaction surface, designers can use the color's saturation and brightness to encode parameters that the system passes to a surface's responses when the user interacts with it. For example, these parameters could signify the strength of the model's response when the user left-clicks on it, or the elasticity (hardness/squishiness) of the model when something collides with it. To illustrate the former, consider painting a Chin interaction surface onto the dog model (Figure 5-17). If this surface indicates a ticklish area, I can change how ticklish the dog is at different points throughout the surface by varying the saturation from 0 (white, no response) to 1 (yellow, strong response). The strength of the dog's response (e.g. how much he wiggles around) when an end user clicks on his chin would then depend on the saturation at the clicked point in the interaction map. The designer can thus specify a fairly complex parameterization in a visually intuitive way.

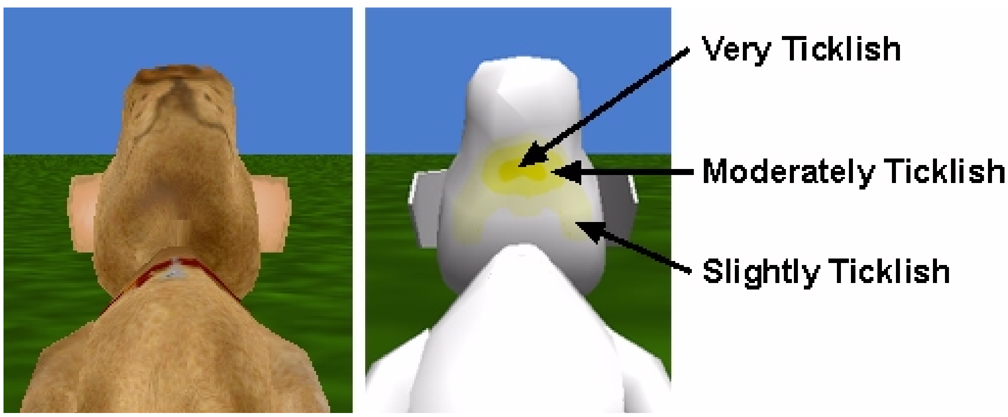

FIGURE 5-17: The interaction map projected onto the dog model reveals the areas where the designer wants the Chin surface to be very, moderately, and slightly ticklish.

5.4.2 Dynamically Modifying Surfaces

There are several ways to dynamically modify the size, shape, location, or response strength of a model's interaction surfaces. The simplest method is to completely switch from one interaction map to another. Consider a character in a 3D fighting game. The character might have two different textures, showing the character with and without armor (Figures 5-18 and 5-19). Because the character's responses will depend on where she is hit and whether she is wearing armor, the interaction designer could create two interaction maps, one for each texture. Note that the designer could link the red areas from each interaction map to the same interaction surface (VulnerableSpot) and define the responses for the red areas in both interaction map areas simultaneously rather than separately.

The system can also directly modify the pixels in the interaction map at run-time by using 3D texture painting algorithms [Hanrahan]. For example, in our 3D fighting game the system could write into a character's interaction map when that character is hit to create sore spots. If the character is subsequently hit in a sore spot she could respond differently than if hit in an untouched area. As another example, the system could slowly increase the saturation of the color in a walking character's interaction map that corresponds to the soles of the character's feet. The character's response to contacts with her feet could then depend on how tired and sore her feet are.

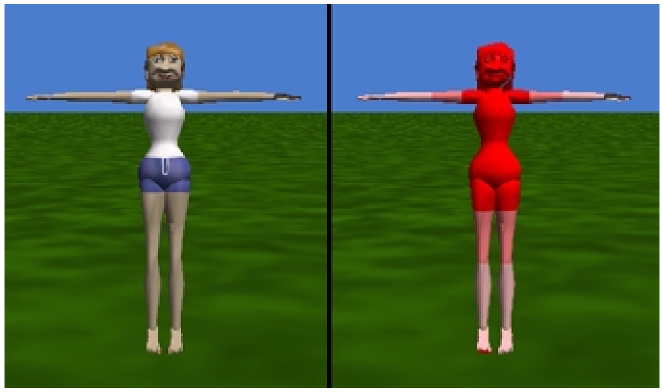

FIGURE 5-18: The designer can paint an interaction map for a character to reflect her vulnerable areas. The interaction map creates a single interaction surfaces; the reds are the same hue, but different saturations. In this case the designer is using the saturation to indicate the character's vulnerability.

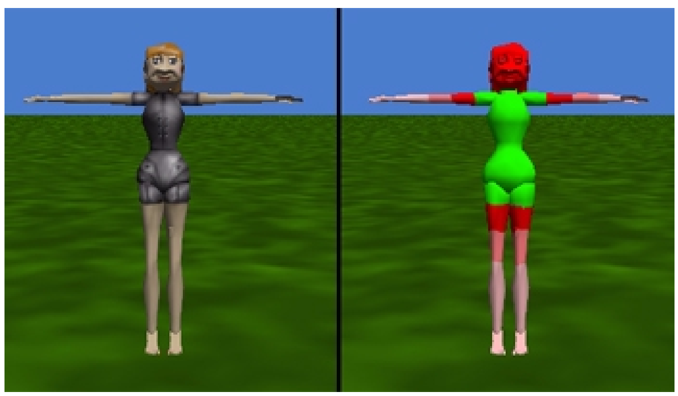

FIGURE 5-19: When the character dons armor (left), the designer swaps her interaction map to reflect the change in her vulnerable areas (right).

In order for dynamically modifying the pixels in an interaction map to work correctly, each area of the model must have sufficient resolution in the interaction map to support any desired modifications. In addition, different parts of a model can not share interaction map regions. If different parts of a model share the same texture region (e.g. a model's left and right arms might map to the same region) to conserve texture space, the interaction map will need its own UV mapping. Otherwise, to borrow the example above, when the character is shot in the left arm both his left and right arms would develop sore spots.

5.4.3 Layered Interaction Surfaces

Allowing models only a single active interaction map does not allow designers to create overlapping interaction surfaces. To designers to be able to create overlapping interaction surfaces, the system can allow models to have multiple active interaction maps. The active interactions maps can be ordered or unordered. If the interaction maps are ordered, the designer can specify a hierarchy for the layered interaction surfaces. The hierarchy specifies the order that surfaces respond to events, and it allows a surface to "capture" an action so that surfaces in subsequent layers will not respond to it. If the layers are unordered than all of the affected surfaces will respond to the action in no particular order.

In addition to allowing overlapping surfaces, systems can use multiple interaction maps to permit designers to create different interaction surfaces for different events. For example, a model might have one set of interaction surfaces that respond to left mouse clicks, and another set of interaction surfaces that respond to collisions. The designer could create these different sets by assigning different layers to different types of events. At run-time the system would only check the layer assigned to a particular event for interaction surfaces.

5.5 Evaluation

For the user study I performed a formative evaluation [Hix] with six students: one art undergraduate, one art graduate student, three computer science undergraduates, and one computer science graduate student. The goal of the study was to verify that people with and without artistic training could create interaction surfaces and assign them responses. I provided users with a model of a doughboy (see Figure 5-20) and a model of a CD player, and pre-built worlds containing responses written for them. Users had to paint interaction surfaces using Deep Paint [Deep Paint] on the doughboy’s stomach, so that they could make him respond to being poked in his stomach, and on the CD Player’s play, stop, forward, back, and open buttons. The users then had to load in the resulting interaction maps, names the interaction surfaces, assign the pre-written responses to the correct surfaces, and run the worlds to verify that they had completed the tasks successfully.

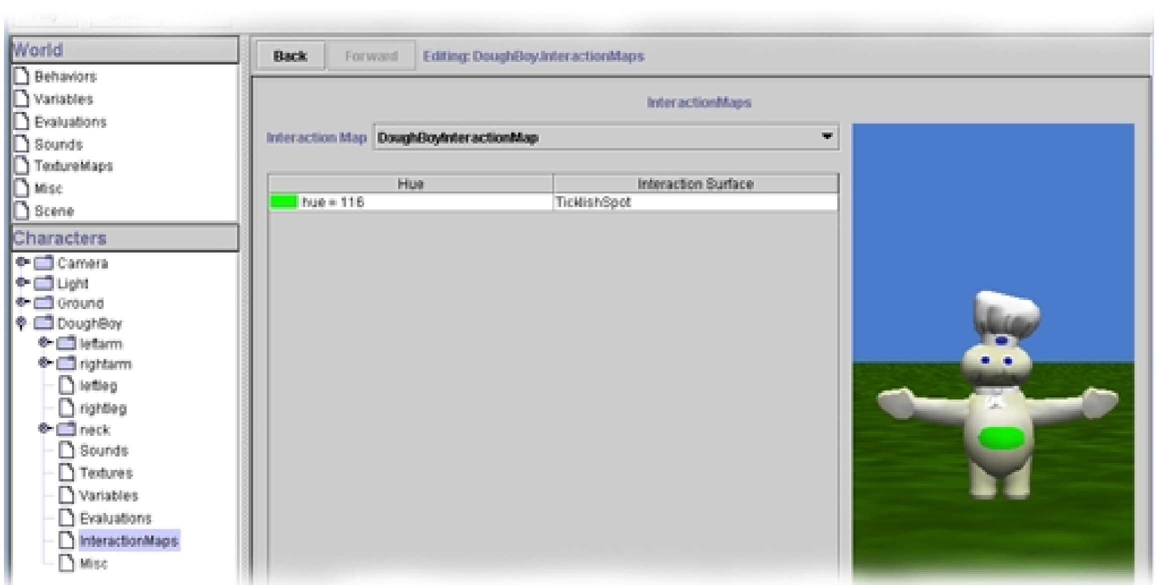

FIGURE 5-20: Creating the doughboy’s interaction surface

All of the study participants found these task easy to perform. Users typically completed painting the interaction surfaces for both objects within five minutes, even if they had no previous 3D painting experience. The primary observation I made while observing this part of the task was that users were not terribly concerned about achieving pixel accuracy while tracing the interaction surfaces over the texture. In particular, when tracing the interaction surfaces over the CD player’s buttons the users did not care if an interaction surface was a little bit larger or smaller than a particular button. Despite this lack of concern with absolute accuracy, users had no trouble activating the interaction surfaces, either by missing the surface or accidentally activating the wrong surface by mistake. I believe that the study participants anticipated that they would target the center of the interaction surfaces, and thus were not concerned with making the edges of the surfaces perfect.

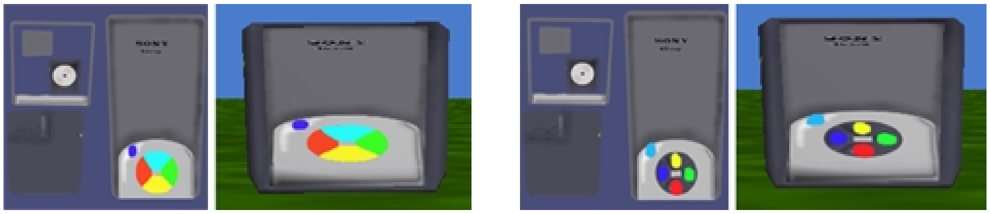

FIGURE 5-21: Sample interaction maps (overlaid on CD player texture) and resulting interaction surfaces created by two study participants.

After a brief explanation of how to use the system, users were also able to load each character's interaction maps, name the resulting surfaces, and assign their responses within five minutes. When naming multiple interaction surfaces, users commented that they found it helpful to view the interaction surface in context (overlaid on the model's texture map). When they had painted surfaces with similar hues, they also found it helpful that the system displayed each interaction surface individually over the model's texture map when editing that surface. One feature that the current implementation does not provide that might be helpful is the ability to hide the interaction surfaces; this would allow users to quickly glance "beneath" interaction surfaces to remember, for example, what button an interaction surface overlays.

5.6 Future Work

Future work related to interaction surfaces includes extending interaction surfaces to image-based rendering, and exploring what other geometric entities (e.g. points, lines, volumes) might prove useful for augmenting interaction with models.

5.6.1 Image-based Rendering

My implementation only works with geometry-based rendering engines. However, I believe that extending it to work with image-based rendering engines is straightforward. I could modify plenoptic image editing [Seitz] to allow designers to paint interaction map images. I could then determine what interaction surface an end user clicks on using a modified object buffer approach: calculate the (x,y) screen position of the mouse cursor, render the world using the interaction map images into an off screen buffer, and determine the color of the pixel in the buffer at the corresponding (x,y) position.

5.6.2 Interaction points, lines, surfaces, volumes, and other geometric entities

The current structure of 3D models is designed primarily to encode the model's appearance and its animation controls. The model's mesh describes its shape, and its texture describes its coloration. Any distinguished points (e.g. insertion points, control points) and lines (e.g. the model's skeleton) determine how animators can move the model. There has been very little effort to add separate entities to the 3D model to simply or improve interaction. In the past, interaction has typically had to borrow the existing structures. For example, attaching responses to groups of polygons. Interaction surfaces are a first step toward providing separate interaction controls. The next logical step is to examine how interaction points, lines, volumes, and other entities could improve the process of creating interactive behaviors. Just like modelers can specify control points to aid keyframing, modelers or designers should be able to specify interaction points, lines, surfaces, and volumes.

5.6.2.1 Interaction points

The only points typically associated with a model are its insertion point and animation control points. When a designer is trying to describe how users can interact with a model, there are several other points that would be useful.

Rotation points could go beyond insertion points to specify the allowable angles of rotation. The designer could then specify how the model will respond if the user tries to exceed those angles. Encoding a model's midpoint and its center of mass might make it easier for a designer to describe how a model responds to collisions. The designer could also encode interaction points like handle points. Handle points are locations on the object where a user is likely to try to select the object (for example, the handle on a coffee cup). When the user is trying to select a particular model in a crowded scene using an image plane technique the system could check which handle point the user's crosshair is closest to on his image plane.

5.6.2.2 Interaction lines

The only lines typically built into a 3D model are its coordinate system axes. We might want to also be able to talk about interaction lines like a model's midline. (parallel to the scene's up vector). A model's vertical axis is often different than its midline, particularly for composite objects (objects built out of different parts).

5.6.2.3 Interaction surfaces

Interaction surfaces have potential applications beyond determining what parts of a 3D model can respond to users. We might want to be able to distinguish models' support surfaces (e.g. the top of a table, the seat of a chair, the palm of a character's hand) and resting surfaces (the bottom of a character's feet, the bottom of a table's legs) to help determine how models in a scene can fit together. We might also want to be able to test for the proximity of surfaces instead of volumes. For example, people react very differently if their faces are close together (their personal space is invaded) than if the backs of their heads are close together.

5.6.2.4 Interaction volumes

Our modeling programs describe the surfaces of objects. Even CSG systems (e.g. Sketch [Zeleznik]), which naturally lend themselves to describing volumes, typically output only a surface description of models. In many worlds, however, it would simplify the designer's task if he could talk about the inside of a particular volume. For example, designers might want to be able to talk about the insides of desk drawers, characters' mouths, doorways, and rooms. These types of interaction volumes would make it easier for designers to assign reactions to events like characters entering and leaving a room.

Interaction volumes do not have to describe geometry. A character's eyesight volume might represent what the character can see, and his hearing volume might represent where the character can hear sounds. An interaction volume could also describe a character's personal space, and the system could use this volume for collision detection rather than a model's geometry.

Interaction volumes could also allow modelers to specify likely object placement sites where users might want to place objects about a particular model. For example, the object placement volumes for a desk model could be on top of the desk and inside the desk's drawers. When a designer is subsequently constructing a scene, he could ask a model to show its placement volumes. The designer could then click on a placement volume to move the currently selected model into that volume. Models that designers position within a placement volume could automatically reparent themselves to the model associated with the placement volume.

5.6.2.5 Interaction relationships

In addition to geometric entities like points and lines, the ability to describe different types of relationships could be useful for specifying interaction. The relationship provided by a scene graph is limited; it was initially designed only to specify how property or state changes to one node affect other nodes.

A model's response tree is one interaction relationship that designers might want to be able to describe separately of the model's scene graph structure, particularly when systems incorporate interaction surfaces. Designers might also want to create a relationship between places, encoding for example which rooms in a house are "next to" each other. A place hierarchy could be useful for designers of navigation interaction techniques.

Designers might also want to arrange interaction points, lines, surfaces, and volumes hierarchy. For example, interaction surfaces could be the distinguish surfaces for an interaction volume (for example, the exits from a room). Similarly, the midpoint of an interaction surface could be an interaction point.

5.7 Summary

We are limited by assumptions about existing practice just as much as our assumptions about how the real world works. By breaking the assumption that designers can only assign reactions to groups of polygons I created Interaction Surfaces. Interaction surfaces provide several advantages over existing approaches to assigning responses.

Interaction surfaces allow designers to distinguish and assign a reaction to any section of a model. Breaking the assumption that reactions can only be assigned to groups of polygons provides more flexibility in how users can interact with objects.

Designers can use the same process for specifying interaction surfaces regardless of the model's geometric representation. As long as a 3D model is parameterizable (such that a 2D texture image can be mapped onto it), a designer can paint interaction surfaces on it.

Specifying interaction surfaces by painting them on models provides designers, regardless of artistic ability, with more control over interaction. Unlike painting texture images, painting interaction maps does not require laying out the UV mapping, and entails tracing over existing details rather than painting new ones.

Designers can create interaction surfaces using existing commercial tools. The algorithms for painting an interaction map are the same as for painting texture images, allowing designers to use existing 2D and 3D paint tools.

Interaction surfaces work in combination with existing methods for specifying responses. In particular, the ability to insert interaction surfaces into a model's response tree means that designers can assign responses to both interaction surfaces and polygon groups.

Designers can create overlapping interaction surfaces. Multi-layer interaction maps allow designers to create interaction surfaces that overlap each other. By contrast, scene graphs expect polygons to appear in a single group, making the creation of overlapping groups difficult.

Designers can pass parameters to response functions. While painting interaction surfaces designers can vary the saturation of the paint color to encode a value like the response strength at that point in the surface. This allows designers to, for example, create a single Ticklish interaction surface where the character's ticklishness varies over that surface.

Interaction surfaces can change at run-time. By dynamically varying the pixel colors in a model's interaction surface, a designer can dynamically modify the size, shape, location, or response strength of a model's interaction surfaces. Re-grouping polygons at run-time is a much more difficult process.

Capabilities that violate the metaphor in order to provide enhanced functionality might be called magical. The power of magic makes it a good thing.
Randy Smith

CHAPTER 6 *Other Promising Ideas*

In this chapter I present techniques that I believe are valuable or interesting, but that are under-threshold for use as one of the three significant ideas that demonstrate my thesis for a variety of reasons. A particular idea might be applicable to a limited domain, a modification to an existing technique, in need of further development, or of questionable value because of the lack of a compelling application. The latter was particularly a problem for this thesis; when there are few proven applications for VR it is very difficult to argue that a technique is particularly valuable if it does not focus on selection, object manipulation, or navigation. My hope is that these techniques will eventually become integrated with existing interaction techniques, or will inspire researchers to the creation of new interaction techniques.

6.1 Interesting Ideas Under Threshold

6.1.1 Interaction techniques that depend on frame rate

Desktop interaction techniques for navigating or moving objects are often velocity-based: a user may move, pull objects toward himself, or push them away at a constant speed. In virtual worlds where the frame rate fluctuates, however, or varies from machine to machine designers might want to make the action's velocity depend on the current frame rate. This could help prevent problems with overshooting when the user moves himself or an object too far because the system cannot display updates fast enough.

6.1.2 Providing orientation indoors

Designers often put the sun in the sky of a virtual world to provide a directional cue to aid users in staying oriented in the world. In addition to locating the sun, users can also orient themselves by checking the direction of the shadows that objects cast. However, this only works when the user is outdoors. We could provide a directional cue indoors by having objects cast shadows as if the sun was shining on them, even though the sun cannot shine on them directly.

6.1.3 A heuristic extension to object associations

We can extend object associations [Bukowski] with the simple heuristic that objects determine whether to move on top of or beneath another object by the relative size of the surface areas that would touch. If the user is dragging a desk around the desk would move beneath the cup, while if the user is dragging the cup it would move on top of the table. If the surface areas are similar, the system could either pick one arbitrarily, rely on past history (whether one object has moved beneath or on top of the other before), or let the user decide. In the latter case, the system could create a copy of the object and show it both above and below the object and let the user click on one to select the desired arrangement. If the user continues to move the object then the copy will disappear as soon as the object's position is no longer ambiguous.

6.1.4 Resizing to fit

In virtual worlds objects can actively aid the user in manipulating them. One of the ways that objects can aid users is by resizing themselves to fit. Specifically, objects can resize themselves so that they fit through openings or in containers. When the user moves an object near an opening (e.g. the neck of a bottle, top of a box, door to a room) the object can resize itself to fit through that opening. This would make it very easy to put a ship in a bottle in a virtual world. Objects can also resize when a user moves them near a container so that they will fit inside. This would, for example, allow the user to carry objects around in his virtual pockets that would not normally fit (e.g. a sofa). Instead of resizing objects to fit in containers, we can instead resize containers so that objects fit inside. This could be particularly useful for multi-user worlds. A crowded room could gradually expand as more and more users enter to make sure that everyone can fit comfortably.

6.1.5 Creating copies to aid selection

While the Voodoo Dolls technique makes it easier for immersed users to select occluded objects or objects in a crowded space, selecting objects in these conditions can still be difficult on the desktop. Imagine trying to select a book off a bookshelf in a desktop world when both the bookshelf and books are squeezed into a few pixels on the screen. We can dynamically generate copies of objects to aid the user in selecting objects. The user positions the mouse pointer where the desired object projects onto the screen and asks the system for help. The system determines what objects project onto those pixels, creates copies of them, and positions the copies in a ring around the user's cursor on the screen. The user can then select the desired object by clicking on its copy; the copy animates to the position occupied by the original object, dragging the mouse cursor along with it.

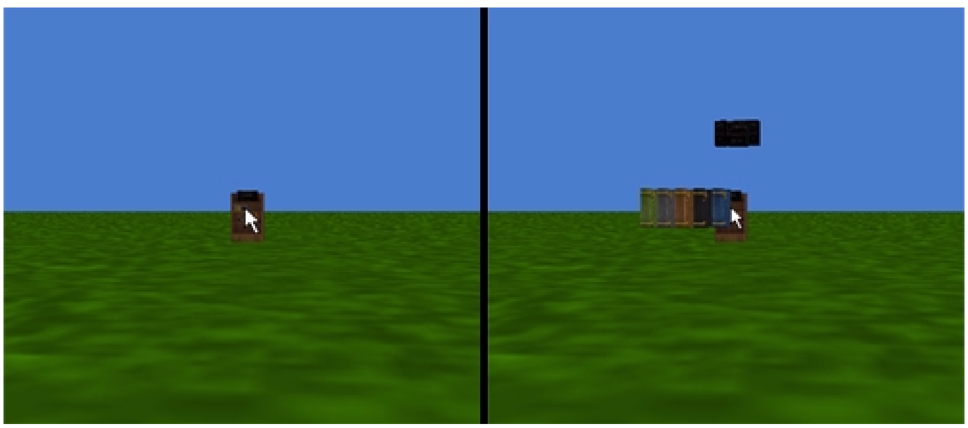

FIGURE 6-1: The select a book from a bookcase (left), the user asks the system to display copies of objects around his cursor on the screen (right).

6.1.6 Culling objects to aid searching

When a user is looking for a particular location or object in a virtual world we would like to help the user avoid accidentally re-tracing his steps. One of the ways we can do this is to temporarily cull places (and their contents) once the user leaves them. For example, once the user has walked down a street or through a room we can remove that area from the world. The user then knows that at any point in time the only places in the world are places that he has not yet checked. When the user finishes searching, we can then add all the culled places back into the world.

Instead of culling places automatically, we can allow the user to choose which places to cull. For example, if the user has a map of the world he can quickly indicate areas where he is sure the place or object is not located to remove them from the world. This method of culling has the advantage that we can then scale up the map to cover the removed places and show the world in more detail. The drawback to this approach, of course, is that if he is mistaken about where the object is located he might accidentally remove it from the world.

We can also allow users to temporarily remove particular types of objects from the world to reduce clutter when he is searching. If the user is looking for a particular building, then he can point to a tree and tell the system to hide all the trees in the world. This technique requires that the every object in the world belongs to a category so that the system knows which objects to remove. Alternately, the user could point to an object and tell the system to hide every *other* type of object. For example, if the user is looking for a building he can point at a building and tell the system to hide every object in that category. The system would then hide every non-building in the world.

6.1.7 Locomotion by selecting two objects

Mine's world compression technique [Pierce97][Mine] allows the user to compress the world by image plane selecting a distant object. Assuming for a moment that the world's clipping plane is set at 500 meters and that the user's arm is half a meter long, with a single action the user can shrink the world by a factor of 1000. However, for very large worlds this can still leave a large percentage of the world outside the user's reach. In addition, once the user has scaled the world down he is still limited to moving the selected object within arm's reach (or moving himself within arm's reach of the object) in the scaled world.

To increase the available scale factor we can allow the user to scale the world by image plane selecting *two* objects. Once the user has selected both objects the world instantaneously scales down so that the position of each selected object matches the hand that selects it. Because the world adjusts its scale, position, and orientation as the user moves his hands around the maximum scaling factor is limited by how closely the user can bring his hands together. Assuming a conservative limit of 0.05 meters, if the user selects two objects, each 500 meters away, he can scale the world by a factor of 10,000 with a single action. The user can then change his position in the world by changing the relative position of his head and hands. In addition, we can allow the user to walk his way hand over hand to move through the world. To scale the world back up the user simply lets go with both hands.

6.1.8 Moding locomotion: indoors vs. outdoors

Interaction techniques for locomotion currently work the same with the user is indoors or outdoors. However, indoor and outdoor environments tend to have different characteristics. Outdoor environments tend to be wide-open 2 1/2D spaces (horizontal motion with an up vector), while indoor environments tend to be enclosed 3D spaces. This begs the question as to whether or not locomotion techniques should function differently depending on whether the user is indoors or outdoors. For example, outdoor travel allows a fair amount of “slop”, but indoor travel requires greater position so that the user is not constantly bumping into furniture or walls (or passing through them if the world does not have collision detection).

One of the ways we can make locomotion different for indoor environments is to make rooms stretch around the user as he moves around. Consider the case where a user wants to step back to get a better view of the room, or fly up for a bird's eye view of the room layout. However, the user may not be able to move far enough because the walls or ceiling get in the way. To address this problem we can stretch the room to keep the user within the room unless the user leaves via the room's exits. As the user steps back to get a better view the wall behind him then with him to keep him within the room. The system also culls any objects from other rooms that would start to appear as the wall moves. The user can then get a side view or bird's eye view of the room without worrying about bumping into the walls or ceiling or having them occlude his view.

6.1.9 New dimensions for privacy in multi-user worlds

In multi-user worlds we can offer privacy along a number of different dimensions. Some systems, particularly games, provide coarse control over privacy by allowing avatar's to be visible or invisible, but adding other dimensions provides more flexibility. For example, users can choose whether or not a particular user can see the correct position and orientation of his avatar. Privileged users can see where a user's avatar is located and where it is looking, while other users might see the avatar at some offset from its actual position and facing in a random direction. We can also show the avatar as a directionless object (e.g. a sphere or cloud) to unprivileged users. We can also change the lighting on an avatar based on the whether or not users looking at it are privileged. Privileged users would see a lit avatar, while other users would just see a featureless black (unlit) shape.

In addition to modifying an avatar's physical appearance, we can also change the effects of the user's actions for some users to provide more privacy. For example, rather than determining whether or not other people can hear the user talking by how loud the user is talking and how far away the others are, we can instead determine whether or not people can hear the user based on the privilege level of the listeners.

6.1.10 Equating logical restrictions and physical restrictions

In the real world we frequently implement logical restrictions that are not backed up by physical restrictions. Speed limit signs on roads do not actually prevent cars from exceeding the speed limit, nor do “authorized personal only” signs alone actually prevent unauthorized areas from entering the area. In a virtual world we can back up logical restrictions with physical restrictions. Signs could locally affect the laws of physics, actually preventing people from exceeding the posted speed limit or from entering an area without authorization.

6.1.11 Adding memory to object manipulation

To aid the construction of virtual worlds we can add memory to objects that lasts across world-building sessions. Each time the designer saves his world the authoring tool can examine where he places objects relative to each other. Thus the tool might notice that a lamp and a phone are usually placed on top of a desk. The tool can also keep track of what objects typically occur in the same world. The tool could determine, for example, that whenever a world has a desk in it that it also generally has a chair as well. Based on this information, the authoring tool can provide shortcuts for positioning and loading objects. When the user adds an object to the scene, the tool can check what other objects are in the scene and suggest places where the user might want to put it. In the case of the lamp and desk, when the user loads the lamp and the world contains a desk the system might place a transparent copy of the lamp on top of the desk. The user can then click on a transparent copy to move the lamp to that position. The authoring tool can also provide shortcuts for loading objects to the world. If the tool has already noticed that when the user loads the desk he also usually loads a chair then when the user actually loads a desk (but has not yet loaded the chair) the system can place a transparent copy of the chair in the world. If the user wants to add the chair he clicks on the transparent copy to turn it opaque. Otherwise the transparent chair slowly fades away, and when it disappears the system removes it from the world.

FIGURE 6-2: When the user creates a new world, the system displays objects he typically adds to the world (left). After the user creates a desk, the system adds the phone (which the user has often placed on the desk) in the world (middle). When the user adds a lamp to the world the system determines that the user often places the lamp on the desk and places a shortcut there (right).

6.1.12 Ordering a space to “Spread Out”

Users occasionally have to select objects, manipulate objects, or maneuver in crowded environments. We can aid the user in these situations by allowing him to temporarily “spread out” the space he is working in. Spreading out a crowded space can make it easier for a user to select an object, provide more accuracy in positioning an object, and make it easier for users to maneuver in a tight spot.

The center of focus, the amount, and the duration for this space scaling depends on the task the user is trying to perform. If the user is selecting an object (e.g. using an image plane technique) the system can locate objects within ten to twenty pixels of his hand on his image plane and move them each another twenty pixels away from his hand. The scaling ends when the user selects an object or moves his hand more than thirty pixels away from its starting position. If the user is manipulating an object, the system can move objects away from his hand in the scene. The system could choose objects within two times the radius of the object the user is manipulating and move them away by half their distance to the user's hand. The scaling could then end when the user lets go of the object, or when he moves the object outside of the scaled space. Finally, if the user is navigating then we can move objects within five meters of the user to twice their current distance. The system could shrink the space back down either when the users commands it or when the user leaves the scaled space.

6.1.13 Duplication instead of locking in multi-user virtual worlds

Multi-user worlds are often extremely concerned about locking so that they can guarantee that only one user can work with an object at a time. In a virtual world, however, we can relax this restriction and allow multiple users to interact with an object by creating copies on demand. When a user picks up an object it leaves behind a transparent copy of itself that slowly fades away. If another user grabs the transparent copy before it disappears, the copy turns opaque and the user can carry it away; the system creates a new transparent copy that again slowly fades away. The system can also create copies for objects that users should not be able to carry away. In this case the user pulls away a copy of the object while the original remains behind.

Some multi-user worlds will still need to enforce locking to ensure that users do not work with an object in conflicting ways. For example, a world might want to guarantee that only one user will paint an object at a time. These worlds could lock objects by *operation* rather than by *possession*. The system could still create duplicates of an object on demand, but some operations on the object might be temporarily unavailable while other users are working on their copies. This approach would allow one user to paint an object while another user relocates it, but prevent multiple users from painting the object at once. The system could also allow users to perform operations simultaneously that are not mutually exclusive. For example, multiple users could simultaneously inspect an object. Implementing this scheme would require that designers generate a mapping between operations and the attributes those operations modify.

6.1.14 Reversing notion of indication and control

In the real world we often create artifacts that indicate the progress or status of some process. In a virtual world we can reverse this relationship so that the artifacts instead control the process's progress or status. Consider a clock. In the real world a clock indicates the current time. In a virtual world we can change the time displayed on the clock to change the time in the world. A user can make time go forward and backward by controlling the motion of the clock's hands. Ishii et al adopted a similar approach for their ambientROOM, using a physical clock prop to allow users to access earlier states of the ambientROOM [Ishii]. By controlling the second hand, minute hand, or hour hand of a clock the user can also control the rate time passes with different levels of resolution. The user could also change the rate time passes using a metronome. Each click of the metronome could cause a second to pass, so a user could set the metronome at a slow pace to slow down the passage of time. Setting the metronome at a fast pace could then speed up the passage of time.

In addition to using man-made artifacts, we can modify the effects of processes to control the processes that caused those effects. For example, as time passes the earth rotates and the sun moves across the sky. The user could grab the sun (or the sky itself) and give it a spin to control time at a larger time: days instead of hours, minutes, or seconds. To control time at an even larger scale the user could control the height of nearby plants. Since trees grow over many years, the user could change the height of a tree to control time at a resolution of months or years. To control time at a resolution of millennia, the user could control the height of a mountain (which is slowly weathered by the elements over time).

6.1.15 Reifying the abstract

We often implement constraints on the behavior of objects in virtual worlds. We might constrain a lamp to move on top of a desk, or specify that a book moves with the bookshelf it rests on. However, these constraints are usually purely abstract: they have no direct physical counterpart in the world. The lack of a physical counterpart makes it difficult for users to manipulate those constraints without falling back on 2D menus and GUIs. To allow users to directly manipulate constraints and other abstractions we can make reify the abstract in virtual worlds.

We might, for example, want to reify the constraints on where an object can be moved. Consider a constraint the controls where users can move an object. The constraint might dictate, for example, that users can move an object within a certain radius of the table it is currently resting on. We could reify this constraint as a sort of virtual tether. The owner of the object could set the length of the tether, attach the tether to the object on one side, and attach the tether to the center of its allowed motion on the other side. Other users could then move the object anywhere its tether permitted, while the owner could move the object anywhere by either stretching or removing the tether.

To provide more flexibility in determining where users can move objects we could instead create boundary markers and allow the object's owner to place them to specify the volume within which other users can move the object. To gain more resolution in specifying the volume the owner simply creates more boundary markers.

We can also extend how real world access cards work by reify access permissions in virtual worlds as access cards. In virtual worlds we can use permissions to determine, for example, whether a user can teleport to a location, pick up or use an object, or even see other users. Rather than having the system represent these permissions only internally, we can create access cards that embody these permissions. Users then have to be carrying the correct access card to receive the requisite permissions. Not only does this allow users to grant permissions by loaning or copying their own access cards, it could be a method of cash extraction in multi-user virtual worlds: virtual vending machines could disperse virtual access cards.

6.1.16 Multiple light-weight views of an object

When a user wants to move an object around on the desktop, he needs methods for both moving the object and for verifying whether or not he has positioned the object correctly. While we have a number of techniques and aids for moving objects (e.g. object associations [Bukowski], force fields [Xiao]). However, the only method we really have for verifying whether or not a position is correct is to divide the user's rendering window up into different views of the object. For example, by dedicating a part of the user's render window to a top down view of the scene we can provide the user with a bird's eye view of the object's position. There are two main drawbacks to this approach. First, the user is forced to dedicate part of his screen real estate to other views, even if he is not currently using them. Second, the canonical views of an object are often not the views that are the most useful. Rather than visualizing the object itself, we are trying to visualize the object's spatial relationship to other nearby objects. We need a light-weight method of viewing this relationship on demand.

We can do this by creating views of objects on demand. This views are very light-weight; the user simply presses a key and the system creates different views of the object and places them around the object on the screen. The user does not have to explicitly specify which views to use; the system chooses the views based on its built in heuristics. The system displays the views until the user releases the key, at which point the views fade away.

The key design decision for these views is where to position the camera and what to focus on. The position of the cameras in the views could depend on the number of factors. If the object is relatively small relative to the objects around it, then the cameras should be placed fairly close to the object. If the object is moving the position of the cameras could depend on the object's path; the camera's might want to be offset from the object in a direction orthogonal to the direction of travel so that the user can see where the object lies relative to objects in that direction. The camera could also position the cameras based on the location of other nearby objects. In this case the system could position a camera so that it is offset from the vector from the manipulated object to a nearby object in a direction orthogonal to the vector. The system also needs to decide what to focus the camera on; the obvious candidates are either the object itself or the point between the manipulated object and a nearby object.

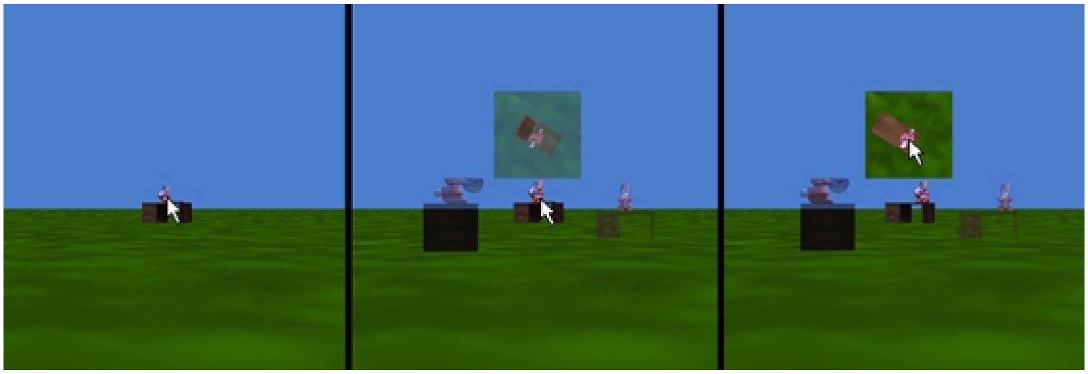

FIGURE 6-3: To help position the bunny on the desk (left) the system create temporary views from different angles (middle). The user can mouse into these views to manipulate the bunny (right).

We can also present views of the same side of the object but at different scales. This would allow the user to quickly make both drastic changes and fine adjustments to the object's position. A top down view from a high in the world would allow the user to quickly move the object long distances, but would not provide much resolution for accurate placement. A close-up view of the object from the top, by contrast, would allow the user to position the object with much greater accuracy.

Of course, we do not need views to be at different scales to allow users to place objects with different levels of accuracy. We could provide two views at approximately the same scale and vary the mapping of mouse motion to object motion between the windows. Small changes in the mouse's position might move the object large distances in one view, but only small distances in the other. We can also use different views to provide access to different constraints in addition to different mouse mappings. For example, the object's motion might be constrained by collision detection in some views and not in others.

6.1.17 System-maintained repositories

User-maintained respositories of both objects (e.g. [Butterworth][Pierce99b]) and places (e.g. [Elvins]) are an established idea for virtual environments. In addition these repositories, we can also explore the use of system-maintained repositories (SMRs). In essence these SMRs extend the notion of caching and pre-caching to virtual worlds. Systems can provide repositories based on user actions (e.g. objects the user has worked with recently) as well as repositories designed to aid likely user actions (e.g. a repository of all the objects in a room).

We can create SMRs that are tied to users, objects, tools, places, and worlds. For users the system can maintain repositories that contain objects they have worked with recently, people they have interacted with recently, places they have recently visited, or copies of their current location at scales they typically work at. For objects the system can maintain repositories of tools users use on them, places they have recently been, or previous versions of themselves. For tools the system can maintain repositories of objects they have worked on recently or other tools users recently used in conjunction with them. For places the system can maintain repositories of objects currently contained within them, objects that have recently been there, previous arrangements of themselves, positions or viewpoints within them that users tend to occupy, places that users leave the space for, or places that users enter the space from. For worlds the system can maintain repositories of places that users frequently visit or places where most users are currently located. The system can make any of these and other repositories available to users on command.

System-maintained repositories introduce some new design decisions in addition to traditional design decisions like the caching strategy to use. We need to determine whether or not the cache should be persistent across visits to a virtual world. We need to determine not only how to visually present the cache to the user, but also what subset of the repository contents to display. For example, if the user asks to see a repository of spaces and is standing in one of the places in the repository, we might want to save space by not displaying that place in the repository. We also need to determine what the effect of retrieving an object from the cache should be. If the user retrieves an object from the cache he might retrieve the actual object from elsewhere in the world, but he might also just retrieve a copy of the object.

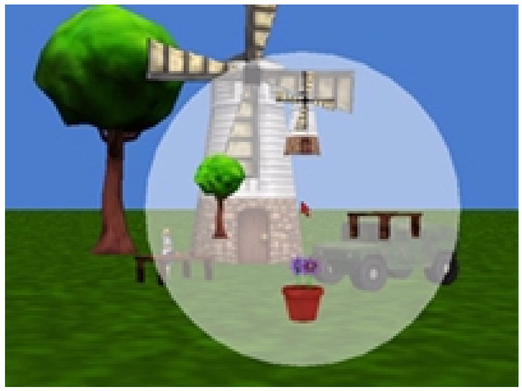

FIGURE 6-4: A system repository might contain objects the user has worked with recently. Note that the objects in the repository are copies of objects in the scene, not the objects themselves.

Because objects and places can change over time, we need to determine whether or not objects or places in repositories update to match the objects or places they represent. On the one hand, updating these objects and places will ensure that they match the actual object or depicted destination so that the user is not surprised if he, for example, travels to a place via a repository copy. On the other hand, if the object or place changes too much the user might not recognize it in the repository. Plus if objects and places in repositories maintain their initial state then we can use these repositories as a form of selected undo: users can retrieve previous versions of an object from a repository. In addition, by capturing a moment in time a place in a repository could provide a mechanism for traveling to both a particular place and a particular time.

If we provide system-maintained repositories for multi-user worlds then we raise another design decision. In particular, different users might have copies of the same objects in their repositories. Assuming that retrieving an object from a repository retrieves the actual object from the world, we need to provide a mechanism for locking repository copies (and communicating to the user that a copy is locked). We could borrow the idea I discussed above of copying objects and locking by operation. In this case whether or not a user could pull an object from a repository could depend on what he intends to do with it.

6.1.18 Symbolic linking

3D widgets [Conner92] are a standard approach to controlling objects and affecting parameters in a virtual world. One of the drawbacks to 3D widgets, though, is that they are generally restricted to controlling a single object. If a widget must be able to control multiple objects, then we need a method of specifying what object the widget is currently affecting.

Consider the example of a resize widget for changing the size of objects in the world. Because resizing objects by grabbing them and stretching them with two hands is problematic for very large or very small objects, a designer might want to use a resize widget instead. We could create the resize widget as a handle attached to the object, but then the object would have to be in reach for the user to resize it. We could instead link a widget to the currently selected object (which the user selects by either using raycasting or image plane selection). However, if the user forgets what the widget is affecting or accidentally selects a different object the widget does not provide no visual cue to remind him what object it is currently affecting. This approach also prevents widgets from simultaneously affecting more than one object.

I believe that a way around this problem is to extend the concept of symbolic linking to virtual worlds. The symbolic link takes the form of a container and a copy of a 3D object. The container is attached to what the user is linking from. In the case of 3D widgets, the container is attached to the widget. The copy is of the object the user wants to link to. By placing the copy into the container the user can establish a symbolic link so that the widget can then affect the object. Symbolic linking allows any widget that previously needed to be attached to an object to be handheld, so that users can work on objects from a distance. In addition, symbolic linking provides a visual cue of what the widget is controlling, and allows a widget to link to multiple objects, either with multiple containers or with a single container that can contain multiple copies.

We can implement this method of symbolic linking using two of the ideas I previously discussed in this thesis. First, we can use the Voodoo Dolls technique to dynamically create a copy of an object. Rather than using the copy to position the object, however, the user places it near the opening of the container. At this point we can use the idea of resizing an object to fit inside a container to allow the user to place the copy inside the container. The copy lasts as long as it is inside the container; if the user lets go of the copy outside the container or replaces it with another object then the system destroys the copy and breaks the symbolic link.

FIGURE 6-5: A resize widget with containers for the target object and reference object.

With symbolic linking a resize widget could link to both the object the user wants to resize and an object that the user wants to use as a reference. Consider a designer making a 3D King Kong world. The designer would like to resize his monkey model to make it approximately one quarter the size of the Empire State Building. The designer creates copies of both objects and places them in the correct container. Once both containers are full the widget displays copies of the monkey and building at the correct relative sizes. The user can then reach in and stretch the monkey to be about the right size, or he can manipulate knobs on the widget to resize the monkey more accurately. Providing local copies of the object to resize and the reference object makes it easier for the user to judge the relative size of the objects, particularly if the objects are not near each other in the virtual world.

With symbolic linking we can implement position and orientation widgets similar to the resize widget. These widgets would also have containers for both the object to control and a reference object, and would display local copies of these objects arranged as they are in the virtual world and at a suitable scale. Both widgets would allow the user to reach in and directly move or turn the object. Both widgets would also provide controls (sliders or knobs) to set the position or orientation of the controlled object more accurately.

We can also implement the WIM as a widget with a container. The object in the container would determine the focus of the WIM. We could also use the object in the container to determine the scale of the WIM. For example, the WIM could cover a volume with a radius equal to four times the longest dimension of the object in the container. Alternately we could add a second container to the WIM to determine the depicted volume. The object in the first container would then set the WIM's focus, while the object in the second container would set the radius: the WIM's radius would equal the distance between the two objects.

.

FIGURE 6-6: A WIM widget with a container that sets the middle of the WIM and a container that sets the edge of the WIM.

We can also create a portal for teleportation or scrying using symbolic linking. The portal would have a single container that would determine the destination that the portal depicts. The user could merely watch through the portal to monitor events in that location, or he could step through the portal to enter that area. This provides a more flexible mechanism of implementing teleportation portals; rather than having fixed destinations the user could adjust the destinations of portals to make them more suitable to his needs. Souvenirs of a place would then be more than a sentimental reminder of a place, because a user could use a souvenir to access that place again in the future.

As a final example, we can use symbolic linking to create a remote control widget. The widget would have a single container for specifying what object the widget affects. The widget could have a small steering wheel and throttle for moving the object around. In addition, if objects in the scene inform the widget of the commands that they accept the widget can add a button for each command to allow the user to invoke those commands.

We can enhance the containers attached to widgets by borrowing the previous idea of system-maintained repositories. We can have the system keep track of what objects the user has recently or frequently placed in the container and display those objects around the container on demand. Rather than creating a new copy of an object using an image plane selection technique the user could then just grab a copy out of the repository and place it in the container. We can also add a user-maintained repository to a container so that the user can explicitly choose which objects to associate with it. Adding this capability to a WIM widget with a container would allow users to quickly access spaces that they visit often by storing souvenirs of those places with the WIM.

This approach is not just limited to linking 3D widgets to objects. Symbolic linking provides a general purpose method of answering “what / which object.” We can establish parent-child and owner relationships with symbolic links. Parents could have containers that allow users to insert copies of multiple objects to make those objects children of that parent. A user's avatars could have a container that holds copies of objects that the user owns. We can also use symbolic links to assign constraints for objects. For example, a character's head might have a container attached to it. By placing a copy in that container we can constrain the character's head to look at a particular object.

6.2 Summary

In this chapter I presented a handful of other ideas that I generated by breaking assumptions about the real world. Although these ideas are less developed and arguably smaller than my three core contributions to the interaction lexicon, I believe that they are interesting in their own right. Some of them may be useful when combined with existing interaction techniques, and others could provide a starting point for future work on interaction techniques.

We don't know who discovered water, but we are pretty sure it wasn't a fish.
Marshall McLuhan

Successful innovation in a particular work can feed the genre and cause it to evolve.
Timothy Oren

CHAPTER 7 *Analysis and Contributions*

The techniques I present in this thesis constitute an existence proof of the ability to generate new interaction techniques that improve on the state of the art for 3D interaction by identifying and breaking assumptions. While the techniques I created prove the approach works, my experience using it suggests that it has certain strengths and weaknesses that designer should consider. As I hypothesized, I found this approach very effective for generating lots of new, unusual ideas. The primary weakness of this approach, however, is that many of the generated ideas are of questionable utility.

I believe that part of this difficulty lies with a problem I mentioned at the beginning of the last chapter: the paucity of proven applications for VR make it difficult to argue that a new technique is valuable if it does not focus on the general problems of selection, object manipulation, or navigation. As a result, this approach is most valuable when it helps generate ideas that augment existing interaction techniques or address one of these problems. Note that all of the primary techniques in this thesis fall into the latter category, while some of the secondary techniques fall into the former. Another part of the difficulty is that, by itself, the approach does not focus on generating ideas to solve particular problems. I often found myself metaphorically holding a hammer and searching for a nail.

I believe that the true value of this approach is best seen in the context of the strengths and weaknesses of other approaches to generating new interaction techniques. The two primary alternatives are Bowman's approach based on a technique-based taxonomy [Bowman99b] and Tan and Robertson's approach based on a task-based taxonomy [Tan]. Bowman's approach essentially encourages a "brute force" exploration of the design space. The focus is on combining existing components of interaction techniques in new ways. This approach is unlikely to yield breakthrough new techniques (it lacks any explicit method for generating new components), but may still yield valuable techniques.

Tan and Robertson's approach focuses on identifying the components of the user's task instead of the components of interaction technique. Their approach helps designers focus on task components that are not adequately supported by existing interaction techniques. While this approach helps designers focus on important problems, it lacks a generative method to help designers create techniques that address those problems.

I believe that the true value of my approach lies in combination with these two existing approaches. A designer can generate a number of new technique components by identifying and breaking assumptions, and then apply Bowman's approach to see if combinations of those components with each other and with existing components yield valuable techniques. Similarly, a designer might apply Tan and Robertson's approach to identify a particular task component, and then use my approach to generate techniques that address that component.

7.1 Contributions

The contributions of this research include new, best-practice interaction techniques for manipulating objects, specifying what parts of an object users can interact with, and navigating large virtual environments. The contributions also include a number of smaller ideas that extend existing interaction techniques or provide starting points for further exploration. Collectively these techniques and ideas demonstrate the value of the final contribution of this research: a new approach to creating interaction techniques by identifying and breaking assumptions, along with a list of assumptions that researchers have broken productively in the past.

7.1.1 The Voodoo Dolls technique

I broke the assumptions that objects can only exist in one place at the same time, that objects in the world are persistent, and that the behavior of objects is independent of the hand holding them to create the Voodoo Dolls technique. The Voodoo Dolls technique is a new technique for manipulating objects in immersive 3D environments that works for a broad range of object sizes, allows users to manipulate both nearby and distant objects, and allows users to accurately manipulate objects when the target location is nearby or far away. Users dynamically create and destroy *dolls*: copies of objects whose functionality depends on the hand holding them. In a user's non-dominant hand a doll provides a frame of reference and provides a focus around which the system can create a context for work. In a user's dominant hand a doll controls the position and orientation of the object it represents. The position and orientation of the doll in the dominant hand relative to the doll in the non-dominant hand specifies the position and orientation of the manipulated object relative to the reference object.

In my formal study comparing the Voodoo Dolls technique with the Indirect HOMER technique, I demonstrated that Voodoo Dolls users were able to both position and orient objects more accurately than HOMER users. The exact difference in accuracy depends on the particular task, but in general the smaller the object and the farther the placement distance the greater the difference. On average, across all the experiment tasks, Voodoo Dolls users positioned objects 88.28% more accurately and oriented objects 72.86% more accurately. In addition to establishing the Voodoo Dolls technique as a best-practice manipulation technique, this result suggests that in the future researchers should concentrate on improving feedback for manipulating techniques.

Other advantages of the Voodoo Dolls technique are:

- The technique overcomes the trade-off between range and accuracy or placement time.
- The technique allows user to manipulate both visible and occluded objects.
- The technique takes advantage of the asymmetric role human hands play when manipulating objects.

7.1.2 Navigating with Places and Landmarks

I broke the assumptions that distant objects are occluded by nearby objects, that objects get smaller and smaller as they get farther away until they are too small to see, that all objects get smaller at the same rate, that objects' appearance remains constant (except for size) at all distances, and that the abstract concept of place is not represented by a concrete object in order to create a new technique for navigating large virtual worlds using place representations and visible landmarks.

Place representations are miniature representations of regions of the world. These representations may be *actual* or *symbolic*. An actual representation is essentially a world in miniature that represents a leaf node in the world's *place hierarchy.* Users can travel using an actual representation by positioning it to achieve the desired viewpoint of the represented place and then instantaneously teleporting to it. By contrast, a symbolic representation is an "artist's rendition" that represents an internal node in the place hierarchy. Users can reach into a symbolic representation and pull out actual or symbolic representations of the places it contains. The system determines which representations are visible based on the user's current location and the place hierarchy. The system displays the actual representations for nearby places, while for farther places it displays the symbolic representation of a place that contains them; the farther the place, the higher the containing place. These place representations float around the user at waist level along the vector to the place they represent. In addition to helping users maintain their orientation relative to other places in the world, they allow users to quickly travel large distances with a small number of discrete gestures.

Visible landmarks enhance local navigation by making it possible for users to see local landmarks even if other objects would normally occlude them. Users can image plane navigate to a visible landmark to quickly traverse short distances, and the visibility of landmarks helps users maintain their orientation.

I also created a panning and zooming 3D WIM that incorporated residue and semantic zooming. In a formal study I demonstrated that users navigate faster through a large virtual world using Places and Landmarks than using a panning and zooming WIM. On average Places and Landmarks users completed the within-place tasks 21.67% faster

(29.813 seconds vs. 38.063 seconds) and the between-place tasks 37.77% faster (36.403 seconds vs. 58.495 seconds). The results of this study suggest that both the visible landmarks and the place representations play a part in allowing users to navigate more efficiently.

Other advantages of navigating with Places and Landmarks are:

- The technique scales smoothly from a world the size of a small town to a world the size of planet.
- Designers can present the illusion of a continuous world even if they only implement discrete places.

7.1.3 Painting Interaction Surfaces

I broke the assumption that a 3D model's interaction semantics should depend on its geometric structure to create a new process for specifying which sections of the model can respond to user input. Instead of assigning responses to groups of polygons, 3D interaction designers instead paints interaction surfaces directly onto the model. The system stores the painted surfaces in an interaction map. This process is both simpler and more flexible. In my evaluation of this process I showed that designers, regardless of artistic ability, can trace interaction surfaces on models. By contrast very few designers can modify the geometric structure of a 3D model. Making the process simpler also made it more flexible. 3D interaction designers can quickly paint new or modify existing interaction surfaces to change a model's interaction semantics.

Other advantages of interaction surfaces and interaction maps are:

- Interaction surfaces allow designers to distinguish and assign a reaction to any section of a model.
- Designers can use the same process regardless of the model's geometric representation.
- Designers can create interaction surfaces using existing commercial tools.
- Interaction surfaces work in combination with existing methods for specifying responses.
- Designers can create overlapping interaction surfaces.
- Designers can pass parameters to response functions.
- Interaction surfaces can change at run-time.

7.1.4 Other promising ideas

I also broke a variety of assumptions to contribute ideas that I hope will become integrated with existing interaction techniques, be developed into full-fledged interaction techniques, or inspire researchers to the creation of new interaction techniques. These ideas include:

- Interaction techniques that depend on frame rate
- Providing orientation indoors
- A heuristic extension to Object Associations
- Resizing to fit
- Creating copies to aid selection
- Culling objects to aid searching
- Locomotion by selecting two objects
- Moding locomotion: indoors vs. outdoors
- New dimensions for privacy in multi-user worlds
- Equating logical and physical restrictions
- Adding memory to object manipulation
- Ordering a space to "Spread Out"
- Duplication instead of locking in multi-user worlds
- Reversing notions of indication and control
- Reifying the abstract
- Multiple light-weight views of an object
- System-maintained repositories

- Symbolic linking

7.1.5 Creating techniques by breaking assumptions

Taken together, these contributions form the existence proof for the primary contribution of this dissertation: a new method for creating interaction techniques by identifying and breaking our assumptions. The techniques this dissertation contributes and the evaluations of them demonstrate that this approach can yield valuable interaction techniques. In addition, this dissertation contributes a list of assumptions about the real world that I and other researchers have broken, intentionally or not, to generate new techniques. These assumptions are.

- Space is linear and continuous.
- The visual properties of a space match the physical properties of the space.
- Appearance does not necessarily reflect reality.
- There is only one world.
- Everyone in the world occupies the same point in time.
- The world is persistent even when not in view.
- The world is spherical.
- Gravity exists.
- Objects do not arbitrarily attract each other.
- The intrinsic and extrinsic properties of objects are inviolate and distinct.
- Objects are persistent: we cannot create or destroy them on command.
- Objects can only exist in one location.
- “Passive” objects cannot affect other objects.
- Objects work the same in all contexts.
- Objects have no built-in knowledge of their properties or functionality.
- Objects (and the world) have no high level semantic memory or history.
- The actions the user can perform and their effects are independent of the current context or task.
- We control our viewpoint and our actions.
- We move through the world, the world does not move around us.
- Abstractions are not represented by physical objects.
- Effects follow causes.
- Objects work the same in the physical and virtual worlds.
- Nearby objects occlude distant objects.
- Objects get smaller and smaller as they get farther away until they are too small to see.
- All objects get smaller at the same rate as they move farther away.
- Objects’ appearance remains constant (except for size) at all distances.
- The abstract concept of place is not represented by a concrete object.

7.2 Future Work

Having addressed specific opportunities for future work in the preceding chapters, I would like to briefly consider the potential for future work on 3D interfaces in general. Even if 3D interaction does not become a popular interaction paradigm, I believe that continued expansion of the 3D interaction lexicon is worthwhile. Creating new ways for people to interact with computers in 3D may help us design better interfaces for people to interface with computers in 2D. Parts of 3D interfaces may transfer to 2D, or they might inspire new designs in 2D. In the process of constructing and studying 3D interfaces we may also learn more about people, and that in turn might help researchers develop better 2D interfaces. 3D interfaces might also impact 2D designs as we move away from a desktop model of computing and toward an emphasis on mobile, networked computing. Interfaces to mobile computers may draw on both position and motion in the real world, and blur the lines between real and virtual worlds (e.g. [Fitzmaurice]). The lessons we learn from interaction with virtual 3D worlds may influence the development of these interfaces.

I believe that the smaller and less developed ideas mentioned in the previous chapter are promising starting points for continuing the developing of the 3D interaction lexicon. I continue to believe that the approach of identifying and breaking assumptions is a valuable method for creating new techniques, although my belief is tempered with an awareness that creating relevant, important techniques with this approach takes time and effort. Researchers working on creating new techniques must bear in mind that there is no one, true process for creating these techniques. Considering the complementary strengths and weaknesses of Tan's, Bowman's, and my approaches, rotating between these approaches might be more effective than using a subset of them. We must also explore new methods of adding to the lexicon. For example, closer collaboration between computer scientists, artists, and designers might prove valuable, considering that collaboration between artists and technologists has historically lead to a breakthrough in the use of a medium.

APPENDIX A Voodoo Dolls Study Data

This appendix contains the raw data, descriptive statistics (mean and standard deviation), and questionnaire results from the Voodoo Dolls user study.

TABLE A-1: Distance Accuracy for Indoor Condition (meters)

Technique	Sex	Cat Bookend	Alice Bookend	Tome	Comfy Chair	Desk	End Table
Voodoo Dolls	Female	0.0221	0.0190	0.0236	0.0733	0.0323	0.0122
Voodoo Dolls	Male	0.0314	0.0306	0.0153	0.0287	0.0405	0.0227
Voodoo Dolls	Male	0.0219	0.0274	0.0091	0.0142	0.0220	0.0243
Voodoo Dolls	Male	0.0116	0.0085	0.0100	0.0890	0.0515	0.0433
Voodoo Dolls	Male	0.0156	0.0061	0.0109	0.0064	0.0322	0.0137
Voodoo Dolls	Male	0.0450	0.0197	0.0287	0.0350	0.0257	0.0185
HOMER	Male	0.0446	0.0375	0.1356	0.0683	0.0295	0.0096
HOMER	Male	0.0605	0.0524	0.0259	0.1141	0.0538	0.0263
HOMER	Female	0.1313	0.1100	0.0359	0.1155	0.1257	0.2342
HOMER	Male	0.0204	0.1676	0.0693	0.0555	0.0482	0.0170
HOMER	Male	0.0354	0.0487	0.0298	0.0700	0.0346	0.1102
HOMER	Female	0.0306	0.0397	0.0873	0.0408	0.0217	0.0579

TABLE A-2: Angle Accuracy for Indoor Condition (degrees)

Technique	Sex	Cat Bookend	Alice Bookend	Tome	Comfy Chair	Desk	End Table
Voodoo Dolls	Female	3.8860	5.1488	4.6146	4.5473	2.6197	0.9675
Voodoo Dolls	Male	4.1445	3.6560	2.9412	2.2199	1.0821	2.5924
Voodoo Dolls	Male	4.5732	2.8840	4.7870	5.6723	1.2322	2.7228
Voodoo Dolls	Male	4.0019	4.8353	1.4304	3.7806	1.2790	1.3062
Voodoo Dolls	Male	5.3462	2.8594	3.3160	1.1170	1.7431	0.5948
Voodoo Dolls	Male	3.0109	4.4329	9.9730	5.1003	1.5577	5.1377
HOMER	Male	7.0175	10.4626	17.2324	3.7901	0.7242	1.6404
HOMER	Male	8.2450	3.2856	1.4730	4.5113	0.2741	1.3660
HOMER	Female	5.8354	8.6914	12.6316	8.2667	1.1521	1.6594
HOMER	Male	30.2696	13.0842	2.5195	1.2978	0.0000	0.7843
HOMER	Male	2.5654	10.1098	4.6140	9.0303	0.7220	0.9177
HOMER	Female	7.9002	7.4198	1.5303	0.0000	0.7252	6.4406

TABLE A-3: Distance Accuracy for Outdoor Condition (meters)

Technique	Sex	Octopus	Coaster	Ferris Wheel	Ring Toss Booth	Hot Dog Cart	Popcorn Cart	Taz	Tweety Bird	Marvin Martian
V. Dolls	Female	1.3250	1.5372	.3601	0.1403	0.0243	0.0614	0.3878	0.0750	0.1362
V. Dolls	Male	0.3787	1.2700	0.0883	0.0557	0.0146	0.0684	0.0415	0.1096	0.0123
V. Dolls	Male	0.6493	1.1112	0.2955	1.1996	0.1646	0.0778	0.0398	0.0437	0.0176
V. Dolls	Male	0.7304	0.5465	0.5376	0.1567	0.0293	0.0659	0.0336	0.0535	0.0710
V. Dolls	Male	0.2861	0.2106	0.4881	0.1857	0.0796	0.0649	0.0286	0.0789	0.0196
V. Dolls	Male	0.6831	1.7156	1.2344	0.1028	0.1121	0.1913	0.0870	0.1300	0.0803
HOMER	Male	2.4453	0.5411	2.3513	4.2016	0.0667	7.2580	0.5091	0.7393	0.4933
HOMER	Male	1.7509	1.8432	0.5800	0.2357	0.0873	0.2426	0.9814	1.0833	0.6035
HOMER	Female	0.3759	2.8337	2.0242	0.9229	0.1938	1.9341	0.2008	5.0740	10.0324
HOMER	Male	1.1873	2.5987	1.4826	18.6384	0.1026	8.8522	7.7446	6.5991	0.5555
HOMER	Male	1.3536	4.0516	0.8377	13.4205	0.0663	1.5541	0.8986	1.0874	7.3139
HOMER	Female	1.4847	2.9397	0.8161	3.5315	0.2783	18.0386	0.5216	0.5296	0.5441

TABLE A-4: Angle Accuracy for Outdoor Condition (degrees)

Technique	Sex	Octopus	Coaster	Ferris Wheel	Ring Toss Booth	Hot Dog Cart	Popcorn Cart	Taz	Tweety Bird	Marvin Martian
V. Dolls	Female	1.3825	1.7663	1.8633	2.8022	1.3270	3.7930	13.6438	16.6140	34.6407
V. Dolls	Male	2.6236	1.3865	3.7297	0.6205	0.5264	12.7223	8.6610	5.5017	6.1475
V. Dolls	Male	2.3091	2.7910	1.1232	2.2490	6.4411	8.8598	4.6792	3.2994	3.6211
V. Dolls	Male	3.0135	0.5609	6.2163	0.9764	2.6035	5.9037	7.8406	7.4627	10.4430
V. Dolls	Male	.6155	0.8214	1.7712	1.7182	5.6111	0.9561	1.3551	13.2909	4.7593
V. Dolls	Male	0.9512	2.5210	2.7326	8.9802	8.5661	2.8487	7.6525	47.8071	29.1540
HOMER	Male	5.6368	3.2238	19.4962	7.3296	18.6764	11.6456	39.7018	45.8100	104.0676
HOMER	Male	2.8258	2.8564	5.0137	8.1354	1.7237	13.3956	31.4133	22.9798	61.9331
HOMER	Female	5.4940	1.5929	37.9432	2.1721	31.4769	19.8568	50.2879	28.7179	101.8971
HOMER	Male	13.0916	1.2524	11.2823	79.8385	2.1453	55.1957	44.6017	50.7540	97.9566
HOMER	Male	7.5500	0.0000	26.0834	4.6155	2.6032	12.6886	50.1406	10.3945	111.2581
HOMER	Female	1.4634	3.1774	18.2971	5.1872	3.9990	11.3111	50.5692	46.5139	101.6554

TABLE A-5: Mean and Standard Deviation for Each Task by Technique.

Task	Voodoo Dolls Distance Mean	Voodoo Dolls Distance Std. Dev.	Voodoo Dolls Angle Mean	Voodoo Dolls Angle Std. Dev.	HOMER Distance Mean	HOMER Distance Std. Dev.	HOMER Angle Mean	HOMER Angle Std. Dev.
Cat Bookend	0.0246	0.0120	4.1604	0.7744	0.0538	0.0403	10.3055	9.9930
Alice Bookend	0.0186	0.0098	3.9694	0.9863	0.0760	0.0522	8.8422	3.3211
Tome	0.0163	0.0081	4.5104	2.9429	0.0640	0.0427	6.6668	6.6630
Comfy Chair	0.0411	0.0330	3.7396	1.7566	0.0774	0.0308	4.4827	3.6254
Desk	0.0340	0.0107	1.5856	0.5599	0.0523	0.0379	0.5996	0.4042
End Table	0.0225	0.0113	2.2202	1.6703	0.0759	0.0859	2.1347	2.1404
Octopus	0.6754	0.3648	1.8159	.9702	1.4330	0.6801	6.0103	4.0924
Coaster	1.0651	0.5812	1.6411	0.8962	2.4680	1.1819	2.0171	1.2915
Ferris Wheel	0.5007	0.3929	2.9061	1.8558	1.3487	0.7232	19.6860	11.5075
Ring Toss Booth	0.3068	0.4397	2.8911	3.0885	6.8251	7.4705	17.8797	30.4267
Hot Dog Cart	0.0708	0.0594	4.1792	3.1740	0.1325	0.0856	10.1041	12.3076
Popcorn Cart	0.0883	0.0508	5.8473	4.3239	6.3132	6.6859	20.6822	17.1950
Taz	0.1031	0.1411	7.3054	4.1164	1.8094	2.9216	44.4524	7.6998
Tweety Bird	0.0818	0.0329	15.6626	16.5114	2.5188	2.6234	34.1950	16.0194
Marvin Martian	0.0562	0.0489	14.7943	13.5597	3.2571	4.2826	96.4613	17.4800

TABLE A-6: Most Frequently Mentioned of the “3 Easiest Things About HOMER”

Facet	Votes
Picking Large / Nearby Objects	5
Moving object at constant distance	3
Easy to rotate objects	2
Roughly equal distance placement	2
Dropping Object	2

TABLE A-7: Most Frequently Mentioned of the “3 Hardest Things about HOMER”

Facet	Votes
Picking distant objects	5
Lack of feedback, especially for rotation	5
Waiting for reeling	5
HMD	2

TABLE A-8: Most Frequently Mentioned of the "3 Easiest Things about Voodoo Dolls"

Facet	Votes
Fast, easy rough placement	4
Feedback	4
Picking	3
Selecting small object using context	3

TABLE A-9: Most Frequently Mentioned of the "3 Hardest Things About Voodoo Dolls"

Facet	Votes
Picking correct context for distant object	4
Rotating context in non-dominant hand	3
Tracker noise	2
HMD	2

TABLE A-10: Responses to the Question "Did your arms get tired during the study?" by Technique

Choice	HOMER	Voodoo Dolls
Not at all	0	1
A little bit	3	4
Moderately	3	1
Extremely	0	0

TABLE A-11: Responses to the Question "During the study, I felt..." by Technique

Choice	HOMER	Voodoo Dolls
No dizziness	3	3
A little dizziness	3	3
Very dizzy	0	0

TABLE A-12: Responses to the Question "Did you feel nauseous at all during the study?" by Technique

Choice	HOMER	Voodoo Dolls
Not at all	4	3
A little bit	2	2
Moderately	0	1
Extremely	0	0

TABLE A-13: Model Dimensions and Placement Distance (meters)

Model	Width	Height	Depth	Placement Distance
Cat Bookend	0.195	0.300	0.301	7.8
Alice Bookend	0.194	0.300	0.532	8.1
Tome	0.275	0.300	0.075	7.9
Comfy Chair	1.140	1.750	1.280	5.5
Desk	2.005	0.936	1.070	6.1
End Table	0.976	1.094	0.670	5.3
Octopus	37.911	14.052	40.000	162.9
Coaster	132.825	50.007	92.946	200.1
Ferris Wheel	15.714	19.768	11.249	118.8
Ring Toss Booth	4.180	3.249	4.180	113.8
Hot Dog Cart	1.872	2.169	1.799	11.0
Popcorn Cart	1.163	1.754	0.634	73.7
Taz	0.589	0.688	0.3726	66.9
Tweety Bird	0.401	0.953	0.437	67.4
Marvin Martian	0.564	0.907	0.392	66.6

APPENDIX B

Places & Landmarks Study Data

This appendix contains the raw data, descriptive statistics (mean and standard deviation), and questionnaire results from the Places & Landmarks user study.

TABLE B-1: User Data

Technique	User ID	Sex	Sense of Direction	Absorption Rating	Spatial Ability
Places & Landmarks	1	Male	2	5	26
Places & Landmarks	2	Male	5	17	38
Places & Landmarks	3	Female	5	13	4
Places & Landmarks	4	Male	4	21	33
Places & Landmarks	5	Male	4	28	27
Places & Landmarks	6	Male	5	20	33
Places & Landmarks	7	Male	5	26	25
Places & Landmarks	8	Male	4	22	36
Places & Landmarks	9	Male	6	15	25
Places & Landmarks	10	Male	7	15	24
Pan & Zoom WIM	11	Male	5	10	31
Pan & Zoom WIM	12	Male	6	26	40
Pan & Zoom WIM	13	Male	4	16	22
Pan & Zoom WIM	14	Male	4	8	31
Pan & Zoom WIM	15	Male	5	18	31
Pan & Zoom WIM	16	Male	7	27	22
Pan & Zoom WIM	17	Female	6	26	25
Pan & Zoom WIM	18	Male	5	19	18
Pan & Zoom WIM	19	Male	6	13	38
Pan & Zoom WIM	20	Male	4	17	16

TABLE B-2: Task Performance Time for Beach and Farm Within-Place Tasks (seconds)

Technique	User ID	B-BHtoPT	B-LCtoBu	B-PTtoLC	F-RStoBS	F-FHtoGS	F-GStoRS
P&L	1	20.250	20.406	18.891	22.390	26.781	31.203
P&L	2	47.610	25.000	22.375	24.985	31.703	24.812
P&L	3	23.640	40.000	23.203	28.109	32.000	35.202
P&L	4	14.875	17.859	26.422	18.453	18.750	20.187
P&L	5	66.828	22.218	47.989	31.953	45.219	32.469

TABLE B-2: Task Performance Time for Beach and Farm Within-Place Tasks (seconds)

Technique	User ID	B-BHtoPT	B-LCtoBu	B-PTtoLC	F-RStoBS	F-FHtoGS	F-GStoRS
P&L	6	18.671	32.954	20.969	24.703	19.875	23.141
P&L	7	17.078	29.312	17.812	36.157	27.641	26.453
P&L	8	16.578	31.282	15.844	27.453	18.968	32.641
P&L	9	14.266	20.906	22.969	30.734	37.125	17.594
P&L	10	35.406	23.767	20.531	16.563	21.735	33.062
WIM	11	28.140	62.500	44.640	28.453	41.515	53.344
WIM	12	31.938	54.234	22.609	56.203	65.704	38.438
WIM	13	19.312	32.641	53.828	30.594	29.594	29.187
WIM	14	41.297	49.531	27.453	31.000	34.547	33.156
WIM	15	86.438	50.110	24.078	20.547	43.844	21.718
WIM	16	32.297	31.765	38.141	25.141	37.203	23.641
WIM	17	27.406	44.375	35.391	22.000	42.625	50.125
WIM	18	58.187	25.469	51.032	21.203	40.907	35.484
WIM	19	20.078	20.469	22.047	19.891	48.890	26.203
WIM	20	23.265	79.594	152.657	33.141	36.937	25.250

TABLE B-3: Task Performance Time for City and Amusement Park Within-Place Tasks (seconds)

Technique	User ID	C-CHtoCu	C-CutoMu	C-MutoCH	A-SwtoCo	A-FWtoSw	A-OtoFW
P&L	1	35.000	47.156	17.609	35.937	23.188	28.078
P&L	2	30.219	35.781	29.344	31.797	43.188	36.000
P&L	3	34.922	59.484	36.171	42.078	43.063	26.750
P&L	4	21.812	36.844	30.437	64.610	22.079	20.922
P&L	5	37.172	68.125	33.297	56.265	44.844	53.515
P&L	6	18.656	29.797	16.766	34.687	22.922	41.609
P&L	7	25.906	26.156	24.266	28.312	19.063	25.672
P&L	8	26.438	31.610	30.641	32.313	38.047	30.348
P&L	9	21.594	31.922	26.516	40.484	33.156	23.312
P&L	10	20.406	25.391	21.282	26.734	49.672	34.250
WIM	11	34.969	45.375	21.484	23.203	28.469	20.297
WIM	12	100.781	62.375	43.485	34.719	36.141	34.234
WIM	13	30.922	77.032	32.265	61.921	32.844	25.156
WIM	14	41.079	58.750	38.157	44.969	31.391	42.688
WIM	15	24.562	26.672	24.140	23.500	26.250	34.984
WIM	16	35.125	46.094	21.015	21.531	35.203	30.890
WIM	17	24.719	23.109	34.672	33.078	33.266	18.078
WIM	18	29.953	34.235	41.875	60.344	56.244	32.016
WIM	19	35.610	132.859	19.187	21.781	25.234	21.609
WIM	20	23.828	28.125	50.797	33.984	41.907	21.016

TABLE B-4: Task Performance Time for Between-Place Tasks (seconds)

Technique	User ID	CtoB-LH	AtoB-PT	FtoB-BH	FtoB-Bu	FtoB-LC	BtoF-BS
P&L	1	37.062	30.578	39.750	36.578	32.219	35.000
P&L	2	32.828	35.500	47.672	38.282	41.812	45.547
P&L	3	44.703	47.016	58.609	70.875	57.960	83.578
P&L	4	38.680	25.406	25.063	39.672	32.607	35.390
P&L	5	48.360	36.859	88.172	60.656	41.640	37.391
P&L	6	27.797	29.890	34.578	39.719	27.828	31.859
P&L	7	22.735	27.047	32.578	25.781	30.266	32.313
P&L	8	34.391	36.704	28.937	27.594	35.016	33.485
P&L	9	29.688	25.312	33.062	39.344	31.344	49.953
P&L	10	24.656	31.781	25.265	49.844	23.156	45.469
WIM	11	73.656	60.531	74.125	81.234	54.266	61.687
WIM	12	49.469	45.390	97.000	77.109	57.687	73.641
WIM	13	31.110	30.188	43.969	58.031	44.609	49.719
WIM	14	100.313	81.594	64.907	61.610	50.375	80.485
WIM	15	42.318	118.656	42.375	36.969	42.985	39.172
WIM	16	71.750	33.641	144.484	48.266	61.218	68.172
WIM	17	39.672	40.141	44.875	55.250	32.985	47.750
WIM	18	44.656	36.203	43.453	50.813	41.344	39.485
WIM	19	71.859	31.188	38.656	32.204	30.219	40.625
WIM	20	64.750	103.202	75.984	50.000	84.938	33.453

TABLE B-5: Task Performance Time for Between-Place Tasks (seconds)

Technique	User ID	AtoF-GS	CtoF-Wi	CtoF-FH	FtoC-Ho	AtoC-Cu	AtoC-Mu
P&L	1	33.422	36.078	35.062	34.328	25.531	54.031
P&L	2	40.063	35.063	33.359	33.813	43.329	34.735
P&L	3	38.172	40.609	29.515	48.156	51.890	49.063
P&L	4	33.891	25.890	25.484	32.375	33.985	53.969
P&L	5	37.844	35.937	38.781	35.578	41.797	38.688
P&L	6	18.485	27.187	23.125	26.968	28.203	38.641
P&L	7	27.859	20.688	21.828	62.281	25.609	25.062
P&L	8	29.750	28.875	32.328	26.578	29.031	32.594
P&L	9	23.687	32.469	35.719	31.422	52.094	56.703
P&L	10	21.859	20.875	28.547	46.156	28.188	33.781
WIM	11	53.219	79.656	44.437	60.343	157.359	104.469
WIM	12	34.704	47.969	53.656	91.688	63.500	110.250
WIM	13	55.093	70.765	52.859	47.547	93.734	36.047
WIM	14	36.906	45.375	53.266	57.109	60.859	66.375

TABLE B-5: Task Performance Time for Between-Place Tasks (seconds)

Technique	User ID	AtoF-GS	CtoF-Wi	CtoF-FH	FtoC-Ho	AtoC-Cu	AtoC-Mu
WIM	15	39.297	28.031	29.219	47.938	41.625	42.016
WIM	16	48.719	46.906	45.922	78.938	113.782	127.719
WIM	17	48.985	41.125	43.828	67.656	72.235	88.750
WIM	18	71.734	80.828	41.687	80.485	49.890	55.031
WIM	19	64.515	67.391	33.297	119.188	43.359	48.578
WIM	20	75.281	91.187	70.078	68.703	79.891	62.391

TABLE B-6: Task Performance Time for Between-Place Tasks (seconds)

Technique	User ID	AtoC-St	BtoC-CH	BtoA-Ca	BtoA-Sw	FtoA-FW	CtoA-Ca
P&L	1	23.563	29.063	34.359	32.672	36.172	28.109
P&L	2	25.172	36.641	44.859	28.968	65.281	38.984
P&L	3	48.563	46.719	65.093	45.172	35.125	60.657
P&L	4	23.594	20.750	52.531	31.656	29.297	65.875
P&L	5	30.579	55.250	64.969	86.141	35.766	44.547
P&L	6	20.735	32.687	36.625	29.422	50.984	66.671
P&L	7	22.453	41.984	34.875	28.062	28.437	27.828
P&L	8	31.218	26.313	32.734	27.828	24.421	28.260
P&L	9	25.109	33.515	36.718	43.656	22.094	28.437
P&L	10	18.828	27.703	28.719	39.984	38.000	21.313
WIM	11	88.188	54.500	103.890	82.156	33.063	42.187
WIM	12	66.016	70.937	74.469	66.688	61.172	58.000
WIM	13	45.313	47.453	50.219	50.890	86.235	50.547
WIM	14	37.875	53.359	57.875	44.828	36.813	42.234
WIM	15	32.500	53.141	37.469	45.516	54.157	53.391
WIM	16	47.672	95.578	65.188	52.640	41.125	46.125
WIM	17	38.094	46.953	36.516	43.625	55.391	85.141
WIM	18	66.532	32.569	39.750	48.062	36.610	33.984
WIM	19	38.422	25.062	34.984	65.172	85.016	58.094
WIM	20	107.234	34.375	61.250	75.594	31.094	31.344

TABLE B-7: Leg Information

Leg	Start Place	End Place	Distance (meters)	Nearest Landmark	P&L Mean (sec)	P&L Std. Dev. (sec)	WIM Mean (sec)	WIM Std. Dev. (sec)
CtoB-LH	City	Beach	758492	Lighthouse	34.090	8.332	58.955	21.105
B-BHtoPT	Beach	Beach	1149	Picnic Table	27.520	17.389	36.836	20.863
AtoB-PT	A. Park	Beach	385840	Picnic Table	32.609	6.658	58.073	32.187
FtoB-BH	Farm	Beach	373406	Beach House	41.369	19.443	66.983	33.373

TABLE B-7: Leg Information

Leg	Start Place	End Place	Distance (meters)	Nearest Landmark	P&L Mean (sec)	P&L Std. Dev. (sec)	WIM Mean (sec)	WIM Std. Dev. (sec)
B-LCtoBu	Beach	Beach	1201	Buoy	26.370	6.877	45.069	18.094
FtoB-Bu	Farm	Beach	374806	Buoy	42.834	14.015	55.149	15.501
B-PTtoLC	Beach	Beach	1218	Lifeguard Chair	23.701	9.049	47.188	38.830
FtoB-LC	Farm	Beach	373475	Lifeguard Chair	35.385	9.759	50.063	15.839
F-RStoBS	Farm	Farm	639	BScarecrow	26.150	6.047	28.817	10.764
BtoF-BS	Beach	Farm	374151	BScarecrow	42.999	15.598	53.419	16.441
F-FHtoGS	Farm	Farm	485	GScarecrow	27.980	8.715	42.177	9.845
F-GStoRS	Farm	Farm	869	RScarecrow	27.677	6.089	33.655	10.926
AtoF-GS	A. Park	Farm	15479	RScarecrow	30.503	7.433	52.845	14.143
CtoF-Wi	City	Farm	429908	Windmill	30.367	6.759	59.923	20.741
CtoF-FH	City	Farm	430152	Farm House	30.375	5.652	46.825	11.548
FtoC-Ho	Farm	City	430724	Hospital	37.766	11.169	71.960	36.022
C-CHtoCu	City	City	168	Church	27.213	6.757	38.155	22.722
AtoC-Cu	A. Park	City	427695	Church	35.966	10.493	77.623	36.022
C-CutoMu	City	City	362	Museum	39.227	14.486	53.4626	32.892
AtoC-Mu	A. Park	City	427970	Museum	41.727	10.910	74.163	31.637
BtoC-CH	Beach	City	757633	City Hall	35.063	10.400	51.393	20.340
C-MutoCH	City	City	287	City Hall	26.633	6.552	32.708	10.955
AtoC-St	A. Park	City	428039	Statue	26.981	8.515	56.785	24.855
A-SwtoCo	A. Park	A. Park	371	Roller Coaster	39.321	12.265	35.903	15.241
BtoA-Ca	Beach	A. Park	385267	Carousel	43.148	13.312	56.161	21.633
CtoA-Ca	City	A. Park	427665	Carousel	41.388	17.397	49.784	15.553
A-FWtoSw	A. Park	A. Park	435	Swings	33.922	11.312	34.695	9.035
BtoA-Sw	Beach	A. Park	385206	Swings	39.356	17.678	57.517	13.871
A-OtoFW	A. Park	A. Park	286	Ferris Wheel	32.046	9.776	28.097	8.050
FtoA-FW	Farm	A. Park	14965	Ferris Wheel	33.336	8.142	55.289	15.553

TABLE B-8: Time to Teleport to Place For Task (seconds)

Technique	User ID	CtoB	AtoB	FtoB	FtoB	FtoB	BtoF	AtoF	CtoF	CtoF
P&L	6	9.937	13.140	17.812	8.797	8.609	11.484	4.641	12.344	7.844
P&L	7	13.828	18.453	16.437	14.641	10.172	12.891	6.265	8.078	12.140
P&L	8	15.125	14.047	13.016	12.734	23.703	12.813	5.500	16.219	12.313
P&L	9	17.141	10.016	22.766	13.391	16.844	29.000	6.422	17.735	9.516
P&L	10	15.172	18.219	9.750	25.453	12.016	15.282	7.218	12.312	11.391
WIM	16	37.766	14.453	118.328	22.719	33.500	39.547	31.953	31.266	28.875
WIM	17	19.781	22.313	26.781	21.766	15.156	26.531	28.985	24.672	22.234
WIM	18	15.75	19.281	18.250	31.297	22.187	13.016	49.609	58.766	24.125

TABLE B-8: Time to Teleport to Place For Task (seconds)

Technique	User ID	CtoB	AtoB	FtoB	FtoB	FtoB	BtoF	AtoF	CtoF	CtoF
WIM	19	28.187	19.406	27.094	19.141	13.234	24.906	24.328	25.172	24.859
WIM	20	22.063	75.453	48.609	29.391	58.938	22.437	38.906	65.250	53.172

TABLE B-9: Time to Teleport to Place For Task (seconds)

Technique	User ID	FtoC	AtoC	AtoC	AtoC	BtoC	BtoA	BtoA	CtoA	FtoA
P&L	6	12.109	6.391	11.781	7.844	19.078	12.297	9.953	11.171	7.062
P&L	7	16.937	11.437	17.890	15.328	26.078	14.141	12.812	11.766	7.843
P&L	8	13.813	13.172	13.234	23.562	13.485	19.703	15.140	9.265	8.953
P&L	9	14.031	18.703	19.672	13.765	18.953	20.812	16.547	11.047	19.328
P&L	10	27.031	12.672	8.422	13.641	15.515	13.610	19.765	8.234	5.234
WIM	16	52.969	48.235	64.703	32.860	73.688	43.266	27.140	66.578	27.578
WIM	17	45.687	56.125	57.891	27.282	29.141	22.157	25.953	19.859	15.344
WIM	18	51.422	27.593	24.953	50.969	16.672	26.610	25.172	23.875	29.688
WIM	19	21.907	21.421	30.938	25.860	17.125	17.156	27.547	18.531	19.406
WIM	20	51.547	55.547	31.750	94.500	24.657	33.344	52.813	19.375	41.422

TABLE B-10: Mean Teleportation Times and Standard Deviations by Technique

Leg	P&L Mean	P&L Std. Dev.	WIM Mean	WIM Std. Dev.
AtoF	6.0092	0.978	34.756	9.846
FtoA	9.684	5.559	26.688	10.107
AtoB	14.775	3.579	30.181	25.464
BtoA1	16.113	3.863	28.507	10.173
BtoA2	14.843	3.717	31.725	11.826
FtoB1	15.956	4.930	47.812	40.983
FtoB2	15.003	6.237	24.863	5.216
FtoB3	14.269	6.114	28.603	18.724
BtoF	16.294	7.234	25.287	9.536
CtoF1	13.338	3.786	41.025	19.465
CtoF2	10.641	1.917	30.653	12.819
FtoC	16.784	5.985	44.706	13.047
AtoC1	12.475	4.399	41.784	16.223
AtoC2	14.200	4.575	42.047	17.930
AtoC3	14.828	5.653	46.294	28.745
CtoA	10.297	1.484	29.644	20.749
CtoB	14.241	2.681	24.709	8.575
BtoC	18.622	4.795	32.257	23.748

TABLE B-11: Most Frequently Mentioned of the "3 Easiest Things About WIM"

Facet	Votes
Image plane navigation	9
Teleporting	6
Zooming in once crosshair positioned	3

TABLE B-12: Most Frequently Mentioned of the "3 Hardest Things about WIM"

Facet	Votes
Positioning crosshair for zooming	6
Finding small landmarks	4
Arm fatigue	2
Zooming felt reversed	2
Fine tuning position	2

TABLE B-13: Most Frequently Mentioned of the "3 Easiest Things about Places & Landmarks"

Facet	Votes
Image plane navigation	9
Using landmarks to travel	6
Teleporting	5
Maintaining orientation in world	3

TABLE B-14: Most Frequently Mentioned of the "3 Hardest Things About Places & Landmarks"

Facet	Votes
Fine tuning position	5
Finding landmarks in city	4
Crosshair occluded by handheld representation	4
Finding place representations	3
Hands colliding when pulling out representation	2
Cables tangling	2
Not seeing landmarks in actual place representations	2

TABLE B-15: Responses to the Question “Did your arms get tired during the study?” by Technique

Choice	WIM	Places & Landmarks
Not at all	3	7
A little bit	5	3
Moderately	2	0
Extremely	0	0

TABLE B-16: Responses to the Question “During the study, I felt...” by Technique

Choice	WIM	Places & Landmarks
No dizziness	4	9
A little dizziness	6	1
Very dizzy	0	0

TABLE B-17: Responses to the Question “Did you feel nauseous at all during the study?” by Technique

Choice	WIM	Places & Landmarks
Not at all	3	5
A little bit	6	3
Moderately	1	2
Extremely	0	0

TABLE B-18: Landmark Information

Landmark	Abbrev.	Length	Width	Height	Place
Lighthouse	LH	7.027	6.912	25.971	Beach
Lifeguard Chair	LC	0.729	1.082	3.416	Beach
Beach House	BH	11.634	12.585	10.595	Beach
Picnic Table	PT	3.814	2.362	1.210	Beach
Buoy	Bu	2.198	2.165	3.254	Beach
Farm House	FH	10.573	21.069	13.389	Farm
Windmill	Wi	4.852	7.487	10.457	Farm
BScarecrow	BS	0.413	1.948	2.318	Farm
GScarecrow	GS	0.440	2.076	2.470	Farm
RScarecrow	RS	0.438	2.067	2.460	Farm
City Hall	CH	15.353	33.061	13.725	City

TABLE B-18: Landmark Information

Landmark	Abbrev.	Length	Width	Height	Place
Statue	St	1.071	0.769	3.181	City
Hospital	Ho	30.733	30.733	120.000	City
Museum	Mu	25.970	58.030	23.185	City
Church	Cu	20.148	9.968	23.228	City
Octopus Ride	O	28.485	27.000	8.580	Amusement Park
Coaster	Co	87.791	52.955	50.007	Amusement Park
Ferris Wheel	FW	11.100	10.000	14.267	Amusement Park
Carousel	Ca	19.984	10.166	6.271	Amusement Park
Swings	Sw	21.186	21.186	7.960	Amusement Park

A man will turn over half a library to make one book.
Samuel Johnson

References

[ActiveWorlds]For more information on ActiveWorlds.com, Inc. visit http://www.activeworlds.com.

[Angus] Ian G. Angus and Henry A. Sowizral. Embedding the 2D Interaction Metaphor in a Real 3D Virtual Environment. Stereoscopic Displays and Virtual Reality Systems II, SPIE Proceedings Vol. 2409, 1995, pages 282-293.

[Bajura] Michael Bajura, Henry Fuchs, and Ryutarou Ohbuchi. Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery Within the Patient. SIGGRAPH 1992 Proceedings, pages 203-210.

[Barrus] John W. Barrus, Richard C. Waters, and David B. Anderson. Locales and Beacons: Efficient and Precise Support For Large Multi-User Virtual Environments. Proceedings of VRAIS 1996, pages 204-213.

[Benford] Steve Benford, Chris Greenhalgh, and David Lloyd. Crowded Collaborative Virtual Environments. CHI 1997 Proceedings, pages 59-66.

[Bertram] Martin Bertram, James C. Barnes, et. al. Piecewise Optimal Triangulation for the Approximation of Scattered Data in the Plane. Computer Aided Geometric Design, vol. 17 no. 8, September 2000, pages 767-787.

[Bier] Eric Bier. Snap-dragging in Three Dimensions. Proceedings of the 1990 Workshop on Interactive 3D Graphics, pages 193-204.

[Bowman97] Doug Bowman and Larry Hodges. An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. Proceedings of the 1997 Symposium on Interactive 3D Graphics, pages 35-38.

[Bowman99a]Doug Bowman and Larry Hodges. Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments. Journal of Visual Languages and Computing, vol. 10 no. 1, February 1999, pages 37-53.

[Bowman99b]Doug Bowman. Interaction Techniques for Common Tasks in Immersive Virtual Environments: Design, Evaluation, and Application. Ph.D. Thesis, Georgia Tech, June 1999.

[Britton] Edward G. Britton, James S. Lipscomb, and Michael E. Pique. Making Nested Rotations Convenient for the User. Computer Graphics, vol. 12 no. 3, August 1978, pages 222-227.

[Brooks88] Frederick P. Brooks. Graphics Reality Through Illusion: Interactive Graphics Serving Science. CHI 1988 Proceedings, pages 1-11.

[Brooks99] Frederick P. Brooks. What's Real About Virtual Reality? IEEE Computer Graphics and Applications, vol. 19 no. 6, November/December 1999, pages 16-27.

[Bukowski] Richard W. Bukowski and Carlo H. Sequin. Object Associations: A Simple and Practical Approach to Virtual 3D Manipulation. Proceedings of the 1995 Symposium on Interactive 3D Graphics, pages 131-138.

[Butterworth]Jeff Butterworth, Andrew Davidson, Stephen Hench, and T. Marc Olano. 3DM: A Three Dimensional Modeler Using a Head-Mounted Display. Proceedings of the 1992 Symposium on Interactive 3D Graphics, pages 135-138.

[Butz] Andreas Butz, Clifford Beshers, and Steven Feiner. Of Vampire Mirrors and Privacy Lamps: Privacy Management in Multi-User Augmented Environments. Proceedings of UIST 1998, pages 171-172.

[Buxton] W. Buxton and B. A. Myers. A Study in Two-Handed Input. Human Factors in Computing Systems, pages 321-326, 1986.

[Canter] David Canter. The Psychology of Place. St. Martin's Press, New York, 1977.

[Card] Stuart K. Card, George G. Robertson, and William York. The WebBook and the WebForager: An Information Workspace for the World-Wide Web. Proceedings of CHI '96, pages 111-117.

[Carpendale] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending Distortion Viewing from 2D to 3D. IEEE Computer Graphics and Applications, July/August 1997, pages 42-51.

[Chalmers] Matthew Chalmers, Robert Ingram, and Christoph Pfranger. Adding Imageability Features to Information Displays. UIST 1996, pages 33-39.

[Chance] Sarah S. Chance, Florence Gaunet, Andrew C. Beall, and Jack M. Loomis. Locomotion Mode Affects the Updating of Objects Encountered During Travel: The Contribution of Vestibular and Proprioceptive Inputs to Path Integration. Presence, vol. 7, no. 2, April 1998, pages 168-178.

[Chase] William G. Chase. Spatial Representations of Taxi Drivers. In Don R. Rogers and John A. Sloboda, editors, Acquisition of Symbolic Skills. Plenum Press, New York, 1983, pages 391-405.

[Chen] Jui Lin Chen and Kay M. Stanney. A Theoretical Model of Wayfinding in Virtual Environments: Proposed Strategies for Navigational Aiding. Presence, vol. 8, no. 6, December 1999, pages 671-685.

[Chenney] Stephen Chenny and David Forsyth. View-Dependent Culling of Dynamic Systems in Virtual Environments. Proceedings of the 1997 Symposium on Interactive 3D Graphics, pages 55-58.

[Conway] Matthew Conway, Jeffrey S. Pierce, Randy Pausch et al. Alice: Lessons Learned from Building a 3D System for Novices. CHI 2000, pages 486-493.

[Conner92] D. Brookshire Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-Dimensional Widgets. Proceedings of the 1992 Symposium on Interactive 3D Graphics, pages 183-188.

[Conner97] D. Brookshire Conner and Loring S. Holden. Providing a Low Latency User Experience in a High Latency Application. Proceedings of the 1997 Symposium on Interactive 3D Graphics, pages 45-48.

[Cruz-Neira] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. SIGGRAPH 1993 Proceedings, pages 135-142.

[Cutler] Lawrence D. Cutler, Bernd Frolich, Pat Hanrahan. Two-Handed Direction Manipulation on the Responsive Workbench. 1997 Symposium on Interactive 3D Graphics, pages 107-114.

[Cult3D] Cult 3D is a product of Cycore. For more information visit http://www.cult3D.com.

[Cutting] James E. Cutting. How the Eye Measures Reality and Virtual Reality. Behavior Research Methods, Instruments, & Computers, vol. 29 no. 1, 1997, pages 27-36.

[Cyberware] For more information on Cyberware scanners visit http://www.cyberware.com.

[Darken93] Rudy P. Darken and John L. Sibert. A Toolset for Navigation in Virtual Environments. Proceedings of UIST 1993, pages 157-165.

[Darken96] Rudolph P. Darken and John L. Sibert. Wayfinding Strategies and Behaviors in Large Virtual Worlds. Proceedings of CHI 1996, pages 142-149.

[Darken99] Rudolph P. Darken and Helsin Cevik. Map Usage in Virtual Environments: Orientation Issues. Proceedings of IEEE VR 1999, pages 133-140.

[Debevec] Paul E. Debevec, Camillo J. Taylor and Jitendra Malik. Modeling and Rendering Architecture from Photographs. SIGGRAPH '96, pages 11-20.

[DeepPaint] Deep Paint 3D (Texture Weapons) is a product of Right Hemisphere, Ltd. For more information visit http://www.righthemisphere.com.

[Deussen] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomir Mech, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic Modeling and Rendering of Plant Ecosystems. SIGGRAPH '98, pages 275-286.

[Disney] For more information on DisneyQuest visit http://www.disneyquest.com.

[Donelson] William C. Donelson. Spatial Management of Information. ACM SIGGRAPH 1978, pages 203-209.

[Dornic] S. Dornic. Subjective Distance And Emotional Involvement: A Verification of the Exponent Invariance. Reports from the Psychological Laboratories, no. 237, University of Stockholm, 1967.

[Downs73] Roger M. Downs and David Stea, eds. Image and Environment: Cognitive Mapping and Spatial Behavior. Aldine Publishing Company, Chicago, 1973.

[Downs77] Roger M. Downs and David Stea. Maps In Minds: Reflections on Cognitive Mapping. Harper & Row, New York, 1977.

[Drucker] Steven M. Drucker and David Zeltzer. CamDroid: A System for Implementing Intelligent Camera Control. Proceedings of the 1995 Symposium on Interactive 3D Graphics, pages 139-144.

[EA] Ultima IX: Ascension is a product of Electronic Arts.

[Elvins] T. Todd Elvins, David R. Nadeau, and David Kirsh. Worldlets - 3D Thumbnails for Wayfinding in Virtual Environments. Proceedings of UIST 1997, pages 21-30.

[Feiner] Steven Feiner and Clifford Beshers. Worlds Within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds

[Fisher86] Scott Fisher, M. McGreevy, J. Humphries, and W. Robinett. Virtual Environment Display System. 1986 Workshop on Interactive 3D Graphics, pages 77-87.

[Fisher89] Scott Fisher. The AMES Virtual Environment Workstation (VIEW). SIGGRAPH 1989 Course #29 Notes.

[Fitzmaurice]George W. Fitzmaurice. Situated Information Spaces and Spatially Aware Palmtop Computers. Special issue on Augmented Reality and Ubiquitous Computing, Communications of the ACM, vol. 36 no. 7, 1993, pages 38-49.

[Forsberg] Andrew Forsberg, Kenneth Herndon, and Robert Zeleznik. Aperture-Based Selection for Immersive Environments. Proceedings of UIST 1995, pages 95-96.

[Fukatsu] Shinji Fukatsu, Yoshifumi Kitamura, Toshihiro Masaki, and Fumio Kishino. Intuitive Control of "Bird's Eye" Overview Images for Navigation in an Enormous Virtual Environment. ACM VRST 1998, pages 67-76.

[Furnas86] George W. Furnas. Generalized Fisheye Views. Proceedings of CHI 1986, pages 16-23.

[Furnas97] George W. Furnas. Effective View Navigation. CHI 1997 Proceedings, pages 367-374.

[Galyean] Tinsley A. Galyean. Guided Navigation of Virtual Environments. Proceedings of the 1995 Symposium on Interactive 3D Graphics, pages 103-104.

[Genau] Andreas Genau and Axel Kramer. Translucent History. CHI 1995 Conference Companion Proceedings, pages 250-251.

[Gleicher] Michael Gleicher and Andrew Witkin. Through-the-Lens Camera Control. SIGGRAPH 1992 Proceedings, pages 331-340.

[Gösele] Michael Gösele and Wolfgang Stuerzlinger. Semantic Constraints for Scene Manipulation. Proceedings of the Spring Conference in Computer Graphics 1999, pages 140-146.

[Gould] Peter Gould and Rodney White. Mental Maps, second edition. Allen & Unwin Inc., Boston, Massachusetts, 1986.

[Guiard] Yves Guiard. Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model. Journal of Motor Behaviors, vol. 19 no. 4, 1987, pages 486-517.

[Haeberli] Paul Haeberli and Mark Segal. Texture Mapping as a Fundamental Drawing Primitive. Fourth Eurographics Workshop on Rendering, 1993, pages 259-266.

[Hanrahan] Pat Hanrahan and Paul Haeberli. Direct WYSIWYG Painting and Texturing on 3D Shapes. SIGGRAPH '90, pages 215-223.

[He] Li-Wei He, Michael F. Cohen, and David H. Salesin. The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing. SIGGRAPH 1996 Proceedings, pages 217-224.

[Heckbert] Paul Heckbert. Survey of Texture Mapping. IEEE Computer Graphics and Applications, vol. 6 no. 11, 1986, pages 56-57.

[Herndon] Kenneth P. Herndon, Robert C. Zeleznik, Daniel C. Robbins, D. Brookshire Conner, Scott S. Snibbe, and Andries van Dam. Interactive Shadows. Proceedings of UIST 1992, pages 1-6.

[Herot] Christopher F. Herot. Spatial Management of Data. ACM Transactions on Database Systems, vol. 5, no. 4, Dec. 1980, pages 493-514.

[Hill] William C. Hill, James D. Hollan, Dave Wrobleski, and Tim McCandless. Edit Wear and Read Wear. CHI 1992 Proceedings, pages 3-9.

[Hindmarsh] Jon Hindmarsh, Mike Fraser, Christian Heath, Steve Benford, and Chris Greenhalgh. Fragmented Interaction: Establishing Mutual Orientation in Virtual Environments. CSCW 1998 Proceedings, pages 217-226.

[Hinckley94a]Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassel. Passive Real Worlds Props for Neurosurgical Visualization. CHI 1994 Proceedings, pages 452-458.

[Hinckley94b]Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. A Survey of Design Issues in Spatial Input. Proceedings of UIST 1994, pages 213-221.

[Hinckley97a]Ken Hinckley, Randy Pausch, and Dennis Proffitt. Attention and Visual Feedback: The Bimanual Frame of Reference. 1997 Symposium on Interactive 3D Graphics, pages 121-126.

[Hinckley97b]Ken Hinckley, Randy Pausch, Dennis Proffitt, James Patten, and Neal Kassell. Cooperative Bimanual Action. CHI 1997 Proceedings, pages 27-34.

[Hix] Deborah Hix and H. Rex Hartson. Developing User Interfaces: Ensuring Usability Through Product and Process. Wiley, New York, 1993.

[Hocking] John G. Hocking and Gail S. Young. Topology. Addison-Wesley, Reading, MA, 1961.

[Hoppe] Hugues Hoppe. Progressive Meshes. SIGGRAPH 1996 Proceedings, pages 99-106.

[Houde] Stephanie Houde. Iterative Design of an Interface for Easy 3-D Direct Manipulation. Proceedings of CHI 1992, pages 135-141.

[HTML] HTML 4.01 specification, www.w3.org/TR/html4/.

[Igarashi98] Takeo Igarashi, Rieko Kadobayashi, Kenji Mase, and Hidehiko Tanaka. Path Drawing for 3D Walkthrough. Proceedings of UIST 1998, pages 173-174.

[Igarashi99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. SIGGRAPH'99, pages 409-416.

[Igarashi01] Takeo Igarashi and Dennis Cosgrove. Adaptive Unwrapping for Interactive Texture Painting. 2001 ACM Symposium on Interactive 3D Graphics (to appear).

[Ingram] Rob Ingram and Steve Benford. Legibility Enhancement for Information Visualization. IEEE Visualization 1995, pages 209-216.

[Ishii] Hiroshi Ishii, Craig Wisneski, Scott Brave, Andrew Dahley, Matt Gorbet, Brygg Ullmer, Paul Yarin. ambientROOM. CHI 1998 Conference Companion, pages 173-174.

[JAlice] JAlice is a 3D authoring tool currently under development by the Stage 3 Research Group. For more information visit http://www.alice.org/jalice.

[Jul] Susanne Jul and George W. Furnas. Critical Zones in Desert Fog: Aids to Multiscale Navigation. Proceedings of UIST 1998, pages 97-106.

[Kabbash] P. Kabbash, W. Buxton, and A. Seken. Two-Handed Input in a Compound Task. Proceedings of CHI 1994, pages 417-423.

[Kallman] Marcelo Kallman and Daniel Thalmann. Direct 3D Interaction with Smart Objects. ACM VRST 1999, pages 124-130.

[Kay] Alan Kay. User Interface: A Personal View. In The Art of Human Computer Interface Design. Brenda Laurel editor, Addison Wesley, 1990, pages 191-207.

[Koller95] David Koller, Peter Lindstrom, William Ribarsky, Larry F. Hodges, Nick Faust, and Gregory Turner. Virtual GIS: A Real-Time 3D Geographic Information System. Proceedings of IEEE Visualization 1995, pages 94-100.

[Koller96] David R. Koller, Mark R. Mine, and Scott E. Hudson. Head-Tracked Orbital Viewing: An Interaction Technique for Immersive Virtual Environments. Proceedings of UIST 1996, pages 81-82.

[Kozlowski] Kozlowski, L. T. and Bryant, K. J. Sense of direction, spatial orientation, and cognitive maps. Journal of Experimental Psychology: Human Perception and Performance, vol. 3, no. 4, 1997, pages 590-598.

[Krueger] Myron W. Krueger, Thomas Gionfriddo, and Katrin Hinrichsen. VIDEOPLACE: An Artificial Reality. CHI 95 Proceedings, pages 35-40.

[Lamping] John Lamping, Ramana Rao, and Peter Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. Proceedings of CHI 1995, pages 1-8.

[Lampton] Donald R. Lampton, Bruce W. Knerr, Stephen L. Goldberg, James P. Bliss, J. Michael Moshell, and Brian S. Blau. The Virtual Environment Performance Assessment Battery (VEPAB): Development and Evaluation. Presence, vol. 3 no. 2, Spring 1994, pages 145-157.

[Liang] Jiandong Liang and Mark Green. JDCAD: A Highly Interactive 3D Modeling System. Third Conference on CAD and Computer Graphics, 1993, pages 217-222.

[Lieberman] Henry Lieberman. Powers of Ten Thousand: Navigating in Large Information Spaces. Proceedings of UIST 1994, pages 15-16.

[Lindeman] Robert W. Lindeman, John L. Sibert, and James K. Hahn. Towards Usable VR: An Empirical Study of User Interfaces for Immersive Virtual Environments. CHI 1999 Proceedings, pages 64-71.

[Loftin] R. Bowen Loftin and P. Kenney. Training the Hubble Space Telescope Flight Team. IEEE Computer Graphics and Applications, vol. 15 no. 5, September 1995, pages 31-37.

[Mackinlay90]Jock D. Mackinlay, Stuart K. Card, and George G. Robertson. Rapid Controlled Movement Through a Virtual 3D Workspace. SIGGRAPH 1990 Proceedings, pages 171-176.

[Mackinlay91]Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The Perspective Wall: Detail and Context Smoothly Integrated. Proceedings of CHI 1991, pages 173-179.

[Maciel] Paulo W. C. Maciel and Peter Shirley. Visual Navigation of Large Environments Using Texture Clusters. Proceedings of the 1995 Symposium on Interactive 3D Graphics, pages 95-211.

[Mapes] D. P. Mapes and J. M. Moshell. A Two-Handed Interface for Object Manipulation in Virtual Environments. Presence, vol 4 no. 4, 1995, pages 403-416.

[Maya] Maya Artisan is a product of Alias|wavefront. Visit http://www.aliaswavefront.com for more information.

[Martin] Kai Martin. Using Bitmaps for Automatic Generation of Large-Scale Terrain Models. Gamasutra, April 27, 2000, www.gamasutra.com/features/20000427/martin_pfv.htm.

[Mine] Mark R. Mine, Frederick P. Brooks, Jr., and Carlo H. Sequin. Moving Objects in Space: Exploiting Proprioception in Virtual Environment Interaction. SIGGRAPH 1997 Proceedings, pages 19-26.

[Murray] Craig D. Murray, John M. Bowers, Adrian J. West, Steve Pettifer, and Simon Gibson. Navigation, Wayfinding, and Place Experience within a Virtual City. Presence, vol. 9, no. 5, October 2000, pages 435-447.

[Niven] Larry Niven and Steven Barnes. *Dream Park*. Ace Books, 1988.

[OpenGL] OpenGL Architecture Review Board, Jackie Neider, Tom Davis, Mason Woo. OpenGL Programming Guide. Addison Wesley Publishing Co., Reading, MA., 1993, pages 389-390.

[Oren] Timothy Oren. Designing a New Medium. In The Art of Human-Computer Interface Design, Brenda Laurel editor. Addison Wesley Publishing Co., Reading, MA, 1990, pages 467-479.

[Parunak] H. Van Dyke Parunak. Hypermedia Topologies and User Navigation. ACM Hypertext 1989 Proceedings, pages 43-50.

[Passini] Romedi Passini. Wayfinding in Architecture. Van Nostrand Reinhold Company, New York, 1984.

[Patrick] Emilee Patrick, Dennis Cosgrove, Aleksandra Slavkovic, Jennifer Ann Rode, Thom Verratti and Greg Chiselko. Using a Large Projection Screen as an Alternative to Head-mounted Displays for Virtual Environments. Proceedings of CHI 2000, pages 478-485.

[Pausch91] Randy Pausch. Virtual Reality on Five Dollars a Day. Proceedings of CHI 1991, pages 265-270.

[Pausch95] Randy Pausch, Tommy Burnette, Dan Brockway, and Michael E. Weiblen. Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures. SIGGRAPH 1995 Proceedings, pages 399-400.

[Pausch96] Randy Pausch, Jon Snoddy, Robert Taylor, Scott Watson, and Eric Haseltine. Disney's Aladdin: First Steps Toward Storytelling in Virtual Reality. SIGGRAPH 1996 Proceedings, pages 193-203.

[Perlin] Ken Perlin and David Fox. Pad: An Alternative Approach to the Computer Interface. SIGGRAPH 1993 Proceedings, pages 57-64.

[Phillips] Cary B. Phillips, Norman I. Badler, and John Granieri. Automatic Viewing Controls for 3D Direct Graphics. Proceedings of the 1992 Symposium on Interactive 3D Graphics, pages 71-74.

[Photomodeler]Photomodeler is a product of Eos Systems. For more information visit http://www.photomodeler.com.

[Photoshop] Photoshop is a product of Adobe. For more information visit http://www.adobe.com.

[Pierce97] Jeffrey S. Pierce, Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. Image Plane Interaction Techniques in 3D Immersive Environments. Proceedings of the 1997 Symposium on Interactive 3D Graphics, pages 39-43.

[Pierce99a] Jeffrey S. Pierce, Brian C. Stearns, and Randy Pausch. Voodoo Dolls: Seamless Interaction at Multiple Scales in Virtual Environments. Proceedings of the 1999 Symposium on Interactive 3D Graphics, pages 141-145.

[Pierce99b] Jeffrey S. Pierce, Matthew Conway, Maarten van Dantzich, and George Robertson. Toolspaces and Glances: Storing, Accessing, and Retrieving Objects in 3D Desktop Applications. Proceedings of the 1999 Symposium on Interactive 3D Graphics, pages 163-168.

[Pirolli] Peter Pirolli and Stuart K. Card (1999). Information Foraging. Psychological Review, vol. 106, no. 4, pages 643-675.

[Poupyrev95]Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. Proceedings of UIST 1995, pages 79-80.

[Poupyrev97]Ivan Poupyrev, Suzanne Weghorst, Mark Billinghurst, and Tadao Ichikawa. A Framework and Testbed for Studying Manipulation Techniques for Immersive VR. ACM VRST 1997 Proceedings, pages 21-28.

[Poupyrev98]Ivan Poupyrev, Susan Weghorst, Mark Billinghurst, and Tadao Ichikawa. Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques. Computer Graphics Forum, vol. 17 no. 3, 1998, pages 41-52.

[QuickTime] QuickTime VR is a product of Apple Computer. For more information visit http://www.apple.com/quicktime/qtvr/.

[Rademacher]Paul Rademacher. View-Dependent Geometry. SIGGRAPH 1999 Proceedings, pages 439-446.

[Rennison] Earl Rennison. Galaxy of News: An Approach to Visualizing and Understanding Expansive News Landscapes. Proceedings of UIST 1994, pages 3-12.

[Robertson] George Robertson, Mary Czerwinski, Kevin Larson, Dan Robbins, David Thiel, and Maarten van Dantzich. Data Mountain: Using Spatial Memory for Document Management. Proceedings of UIST 1998, pages 153-162.

[Robinett] Warren Robinett and Richard Holloway. Implementation of Flying, Scaling, and Grabbing in Virtual Worlds. 1992 Symposium on Interactive 3D Graphics, pages 189-192.

[Samet] Hanen Samet. The Design and Analysis of Spatial Data Structures, Addison Wesley, 1989.

[Seitz] Steven M. Seitz and Kiriakos N. Kutulakos. Plenoptic Image Editing. International Conference on Computer Vision ‘98, pages 17-24.

[ShapeCapture]ShapeCapture is a product of ShapeQuest Inc., for more info visit http://www.shapequest.com.

[Sharkey] Paul M. Sharkey, Matthew D. Ryan, and David J. Roberts. A Local Perception Filter for Distributed Virtual Environments. Proceedings of VRAIS 1998, pages 242-249.

[Shaw] Chris Shaw and Mark Green. Thred: A Two-Handed Design System. Multimedia Systems Journal, 3(6), 1995.

[SmartScene]SmartScene is a product of Multigen Inc. More information is available at http://www.multigen.com.

[SmithG] Gareth Smith. Cooperative Virtual Environments: Lessons from 2D Multi-User Interfaces. CSCW 1996 Proceedings, pages 390-398.

[SmithR] Randall B. Smith. Experiences With the Alternate Reality Kit: An Example of the Tension Between Literalism and Magic. Proceedings of CHI + GI 1997, pages 61-67.

[Snyder] John M. Snyder and James T. Kajiya. Generative Modeling: A Symbolic System for Geometric Modeling. SIGGRAPH 1992, pages 369-378.

[Softimage] Softimage is a product of Avid Technology, Inc. For more information visit http://www.softimage.com.

[Spence] Jonathan D. Spence. The Memory Palace of Matteo Ricci. Viking Penguin, 1994.

[Steck] Sibylle D. Steck and Hanspeter A. Mallot. The Role of Global and Local Landmarks in Virtual Environment Navigation. Presence, vol. 9, no. 1, February 2000, pages 69-83.

[Stephenson]Neal Stephenson. *Snow Crash*. Bantam Books, 1992.

[Stevens] Albert Stevens and Patty Coupe. Distortions in Judged Spatial Relationships. Cognitive Psychology, Oct. 1978, vol. 10, no. 4, pages 422-437.

[Stoakley] Richard Stoakley, Matthew J. Conway, and Randy Pausch. Virtual Reality on a WIM: Interactive Worlds in Miniature. CHI 1995 Proceedings, 265-272.

[Strohecker] Carol Strohecker and Barbara Barros. Make Way for WayMaker. Presence, vol. 9, no. 1, February 2000, pages 97-107.

[Sung] Un-Jae Sung, Jae-Heon Yang, and Kwang-Yun Wohn. Concurrency Control in CIAO. IEEE VR 1999, pages 22-28.

[Sutherland] Ivan Sutherland. A Head-Mounted Three Dimensional Display. Fall Joint Computer Conferences, AFIPS Conference Proceedings vol. 33, 1969, pages 757-764.

[Tan] Desney S. Tan, George G. Robertson, and Mary Czerwinski. Exploring 3D Navigation: Combining Speed-Coupled Flying with Orbiting. CHI 2001 Proceedings, pages 418-425.

[Thibault] William C. Thibault and Bruce F. Naylor. Set Operations on Polyhedra Using Binary Space Partitioning Trees. SIGGRAPH 1987, pages 153-162.

[Thorndike] Perry Thorndike and Barbara Hayes-Roth. Differences in Spatial Knowledge Acquired from Maps and Navigation. Cognitive Psychology, vol. 14, no. 4, Oct. 1982, pages 560-589.

[Tognazzini] Bruce Tognazzini. Principles, Techniques, and Ethics of Stage Magic and Their Application to Human Interface Design. INTERCHI 1993 Proceedings, pages 355-362.

[Usoh] Martin Usoh, Kevin Arthur, Mary C. Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P. Brooks, Jr. Walking > Walking-in-Place > Flying, in Virtual Environments. SIGGRAPH 1999 Proceedings, pages 359-364.

[Venolia] Dan Venolia. Facile 3D Direct Manipulation. Proceedings of INTERCHI 1993, pages 31-36.

[Viewpoint] The Viewpoint Corporation sells 3D models on the Internet at http://www.viewpoint.com.

[Viega] John Viega, Matthew J. Conway, George Williams, and Randy Pausch. 3D Magic Lenses. Proceedings of UIST 1995, pages 51-58.

[Vinson] Norman G. Vinson. Design Guidelines for Landmarks to Support Navigation in Virtual Environments. Proceedings of CHI 1999, pages 278-285.

[Ware90] Colin Ware and Steven Osborne. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. Proceedings of the 1990 Symposium on Interactive 3D Graphics, pages 175-183.

[Ware93] Colin Ware, Kevin Arthur, and Kellogg S. Booth. Fish Tank Virtual Reality. INTERCHI 93 Proceedings, pages 37-42.

[Ware97] Colin Ware and Daniel Fleet. Context Sensitive Flying Interface. Proceedings of the 1997 Symposium on Interactive 3D Graphics, pages 127-130.

[Waterworth]John A. Waterworth. A Pattern of Islands: Exploring Public Information Space in a Private Vehicle. In Multimedia, Hypermedia, and Virtual Reality: Models, Systems, and Applications. P. Busilovsky, P. Kommers, and N. Streitz, Eds. Springer, Berlin, 1996.

[Weghorst] Hank Weghorst, Gary Greenberg, and Don Greenberg. Improved Computational Methods for Ray Tracing. ACM Transactions on Graphics, vol. 3 no. 1, 1984, pages 52 - 69.

[Welles] Citizen Kane was written by Orson Welles and Herman Mankiewicz, directed by Orson Welles, and produced by RKO Radio Pictures Inc, 1941.

[WorldUp] WorldUp is a product of Engineering Animation, Inc. For more information visit http://www.sense8.com.

[Xiao] Dongbo Xiao and Roger Hubbold. Navigation Guided by Artificial Force Fields. CHI 1998 Proceedings, pages 179-186.

[Yates] Frances A. Yates. The Art of Memory. University of Chicago Press, Chicago, 1966.

[Zeleznik] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Huges. SKETCH: An Interface for Sketching 3D Scenes. SIGGRAPH 1996 Proceedings, pages 163-169.

[Zhai] Shumin Zhai, William Buxton, and Paul Milgram. The “Silk Cursor”: Investigating Transparency for 3D Target Acquisition. CHI 1994 Proceedings, pages 459-464.