Design, Implementation and Evaluation of a 3D Magic Lens Interface utilizing a Handheld Device within an Immersive Virtual Environment

January 2010

Miguel MIRANDA MIRANDA

Design, Implementation and Evaluation of a 3D Magic Lens Interface utilizing a Handheld Device within an Immersive Virtual Environment

Submitted to

Graduate School of Information Science and Technology

Osaka University

January 2010

Miguel MIRANDA MIRANDA

Abstract:

Virtual reality is used to simulate phenomena and environments that are difficult or impossible to experience in the real world. In the early days of the field, because of its high cost, only a few applications took advantage of the possibilities offered by virtual reality, and most such applications were dedicated to specialty training. However, as the technology evolved, its costs have declined. Other devices such as video games, cellular phones, three dimmensional (3D) movies and television have begun including virtual reality concepts. This tendency can be expected to increase in the future.

A key aspect of virtual reality is the interaction between the system and the user. That is the modification of the virtual environment according to input information provided by the user. Real-time interaction enhances the sensation of immersion in the virtual environment. The user interacts with the virtual environment with the help of various interfaces. 3D interfaces present problems that are differents to those experienced with two-dimensional (2D) interfaces, such as imperfect depth cues, unstable mid-air hand placement, and others.

The present study investigates the problem of interaction in 3D. Specifically, it addresses the selection and manipulation of virtual objects. A number of methods have been developed for interacting in virtual environments. However, it is difficult to provide a unique, generic interaction method that can be used under different conditions and in different environments. Previous interfaces have shown their effectiveness for particular tasks and applications. The study also addresses the implementation of a magic lens interface using a handheld device. Magic lens interfaces are widely used in many virtual reality systems. An effective implementation of magic lens uses a transparent plate as a prop for providing a real object representing the lens and giving tactile feedback information. The study introduces a new implementation method based on a handheld device and streaming software technology that aims at enhancing interaction by providing an active prop that can store and manipulate virtual objects through a previously captured view. Learning to operate the interface is easy because the user is able to interact with the 3D environment using the same resources normally provided by 2D interfaces, which are present on the handheld device. Furthermore, it is flexible, because the 2D interfaces are programmable. Besides the magic lens interface, an architecture for implementing other 3D interfaces were developed in order to offer the posibilities of interaction methods.

In order to evaluate the usability of the proposed interface, three users studies were conducted. The results showed that the magic lens interface provides advantages for selecting objects under conditions where other interfaces experience difficulties. These include moving objects, or objects separated from the user by long distance. During manipulation, the user found our interface very intuitive, but its size and weight presented a serious drawback that must be resolved in order to increase its performance.

Keywords: Immersive virtual environments, Magic-lens interface, Virtual ob-

ject manipulation

Acknowledgments

This work was completed under the supervision of Professor Haruo Takemura of the Graduate School of Information Science an Technology at Osaka University. I would like to express profound gratitude to my advisor, Professor Haruo Takemura, for giving me the opportunity of studying in his laboratory. Without the continuous encouragement and advice of Professor Takemura, I could never have completed this thesis. I would also like to thank Professor Kiyoshi Kiyokawa, who has also been a wonderful source of support and guidance throughout the development of this work.

I am grateful to Professor Fumio Kishino, Professor Takao Onoye and Professor Yasushi Yagi, as members of the thesis committee, for their insightful comments on this research.

I express my sincere gratitude to the Ministry of Education, Culture, Sports, Science and Technology of Japan (MEXT) for providing scholarship and academic support throughout the doctoral program.

I have also benefited greatly through being able to meet and work with a number of people during my doctoral work at the Osaka University. I want to thank to all the members -staff and students- of Takemura Laboratory for their help and support. I also express my gratitude to those who have indirectly helped me in the completion of this work.

I would like to acknowledge the enormous support from my family and to express my sincere thanks to my parents for their understanding and support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The number of available virtual reality applications has gradually increased, along with the range of activities that make use of them. In the early days of the field, because of high implementation costs, only users, such as the military [Furness 1986, McCarty 1994] and space exploration [Fisher 1987] fields, were able to develop applications for providing realistic training in simulated extreme, high-risk conditions.

However, overtime, as computer systems increased in power and new input and output device technologies emerged, the prices related to implementing virtual reality systems declined considerably [Lawton 2006]. Lower prices, and the attraction of experiencing an enhanced level of immersion in 3D virtual worlds, has made possible the applications of virtual reality to new fields. In addition to traditional scientific, medical, industrial, and academic areas, virtual reality concepts have diversified to art and entertainment [Burdea 2003]. An example of the wider presence of virtual reality technologies is the Wii[Nintendo 2009] system, which is a video game console that includes a controller for sensing 3D motion, that provides similar functions to those offered by tracker devices in immersive virtual reality systems. In the future, this expansion can be expected to continue and become further diversified. The integration of there new technologies into new apparatuses, such as mobile devices, has resulted in a great variety of new applications integrating virtual and augmented reality concepts [Schmalstieg 2002].

Working within 3D spaces is often considered to be more appealing than working in 2D desktop environments. The experience of seeing 3D images and interacting in 3D space resembles our real world, while adding the additional advantage of the providing the capability of simulating environments and phenomena that are not always possible to experience in reality. However, the creation of virtual reality applications is a complex task that requires the coordination of different hardware and software components. A virtual reality system captures and processes interaction information provided by the user, after which, it provides simulated stimulus perceptible by human senses. In order to provide a pleasant and credible experience, the overall process must be accomplished in real-time.

Communication between human users and computers is established by means of interfaces. The desktop metaphor offers an effective and easy to understand interaction model for 2D environments. Using a desktop, the user provides information, by means of the mouse and a keyboard, to manipulate the controls and graphics

elements presented on the desktop screen. Since the user is able to see and "feel" the activity she/he is performing, the tactile feedback information provided by the mouse and keyboard enhance the interaction, The desktop provides a set of generic interactions such as mouse clicks, drag-and-drop actions, and key presses, which are interpreted in different ways depending on the context in which they are performed. The level of sophistication of 2D interfaces is a fundamental element that has influenced the broad acceptance of computer systems in our society. Newcomers intuitively learn the metaphor in a short period of time. Then, after the basic interactions have been assimilated, these new users are able to interact with almost any application, by means of the same set of interactions.

In virtual reality, the creation of interfaces for 3D spaces that have a level of effectiveness and sophistication equivalent with the 2D desktop metaphor is a problem that has yet to be fully resolved. The diversity of input devices, technologies for sensing movement in 3D space with six degrees of freedom, problems related to depth clues, occlusion and other factors, all contribute to the complexity of 3D interface development.

Several 3D interaction techniques have being proposed [Poupyrev 1998b, Bowman 2006, Bowman 2001, Bowman 2008, Bowman 2005] along the evolution of virtual reality technologies. However these interfaces have only proven to be effective in applications under specific and controlled conditions or environments, and there is no current generic standard 3D interaction metaphor, that provides a level of flexibility that is sufficient for use under different environments and conditions. In other words, an equivalent to the desktop metaphor for 3D environments does not yet exist.

Interaction can be defined as all the activities taking place between a user and a computer system that are necessary to accomplish a common task. For simplification, the interaction tasks are studied separately here. Object selection, manipulation and navigation are the most basic tasks. Selection provides the user the ability to distinguish an individual object from among several other objects for the purpose of manipulating it. Manipulation is defined as the ability to change the position of an object. While navigation provides the ability to explore the environment. By combining these three operations, an interface is able to offer a basic set of interactions that allow the user to explore and modify a virtual world. The interest of the research described in this dissertation is focused on the selection and manipulation of virtual objects in immersive virtual environments.

1.2 Research Goal and Challenges

The difficulty in establishing universal interaction methods in 3D spaces has lead to the development of various interfaces, each with its own particular characteristics. Each interface is designed to be suitable for specific tasks and/or conditions in a virtual environment. However, the proliferation of interfaces has a potential drawback, users must learn how to handle a specific individual interface for each

separate activity. As activities become more complex and increase in number, users can easily become confused since they are required to memorize different techniques.

The research described in this dissertation conforms to the proposal of an interface that can simplify interaction within 3D virtual environments. The proposed interface is inspired by a metaphor called a magic lens, [Bier 1993, Pierce 1997, Schmalstieg 1999] which is used extensively in virtual and augmented reality applications. However, this interface suffers from limitations inherent to its implementation. The interface proposed in this dissertation not only tries to retain the advantages offered by previous works, it also seeks to enhance its functions and alleviate its limitations. The conditions the interface tries to fulfill include:

- The interface is expected to be easy to learn and use: It should integrate interaction metaphors that resemble familiar methods used in daily life, while simultaneously considering the extension of capabilities necessary to overcome the limitations imposed by reality. For example, a see-through lens metaphor resembles a transparent glass window. If it is extended with an X-ray view, the lens allows the user to explore the inner characteristics of objects. Therefore, the adoption of familiar interaction techniques would reduce the time the user expends learning how to use the interface and at the same time, would encourage concentration on the task being performed, instead of manipulation of the interface.

- The interface is expected to improve haptic sense feedback: As more senses receive stimulus simultaneously, the feeling of immersion increases. Most virtual reality systems offer visual and sound immersion. If a handheld device equipped with buttons and a touch screen is integrated into the system, the tactile feedback information produced by physical contact with those elements would provides additional clues to enhance interaction.

- The proposed interface is expected to improve manipulation accuracy: Some previous 3D interaction techniques were noted for problems related to selection and manipulation of small and distant objects [Poupyrev 1998b, Bowman 1999a]. As the separation between the object and the user increases visual clues tend to decrease, making the selection difficult. Moving objects to positions far away from the user results in the same problem. Capturing snapshots of the environment, which are then used to select objects such as this, reduces the interaction space and increases accuracy. This idea is explored in [Pierce 1997]. The user, wearing a glove, uses hand gestures for capturing snapshots. Then, by manipulating the objects contained in the snapshot, the user is able to manipulate the 3D object indirectly. However, since the manipulation is performed in a 2D space, the difficulty of controlling the distance relative to the user is a drawback of this technique. The partial occlusion of a section of the environment by the snapshot is a minor disadvantage as well.

- The interface is expected to provide high resolution for detailed exploration: On projector-based virtual reality systems such as the CAVE, small objects

are prone to lose detail due to the size of individual pixels, even though high-resolution projectors are used. Developing a magic lens with a high resolution display would provide close up details that are difficult to perceive by most projection technologies.

- The proposed interface is expected to offer high flexibility: Handheld devices are equipped with a processor, a storage unit, and a graphical display. Thus, they provide a complete miniature computer system with characteristics similar to modest laptop computers. Such handheld devices are normally launched with operating systems adapted to their hardware components, but they often include programmable libraries for developing applications that are similar to those developed for desktop computers. Therefore, developers should be able to program sophisticated visual 2D interfaces and take advantage of all the functions provided by the handheld devices by using familiar development tools.

- The interface shall be developed with the goal offering set of interaction techniques: By providing a set of basic generic functions that can be to be applied in a wide range of environments and conditions, the proposed interface attempts to simplify 3D interaction through the extension of 2D metaphors and by combining 2D and 3D approaches.

- The interface is expected to offer great extensibility: Since the handheld device is programmable, it will be possible to implement new functions. Moreover, because new handheld devices often include additional electronics components such as accelerometers or the Global Positioning System (GPS), which could be integrated for the creation of new interactions, it is possible to program the new capabilities offered by handheld devices through software libraries. As new 2D interactions are integrated into mobile devices, and as users become accustomed to them, it will be desirable to extend their application into 3D space.

- The interface is expected to be portable: Physical portability refers to the size of the devices. The interface considers implementations for handheld computers, Personal Digital Assistants (PDAs), and mobile devices that can be carried in one hand. Software portability refers to the hardware and operating system's ability to support the interface. One important characteristic the interface design would offer is independence with respect to hardware infrastructure and software platform.

- The interface is assumed to offer real-time interaction response: An important requirement of virtual reality systems is real-time interactivity which contributes to the sensation of immersion. As new components are integrated into the virtual reality infrastructure, the overall performance of the system becomes prone to degradation. Therefore, mechanisms for reducing inter-

communication delays and synchronizing activities among components are essential design features.

- The interface is expected to offer high adaptability: Since new handheld devices appear continuously, an interface that can be migrated onto different devices while preserving the same basic interaction mechanism, is an advantageous. At the same time, if the interface is capable of modifying individual aspects of interaction through 2D interface programming, such interaction could be adapted to new environments or conditions.

- It is assumed that the interface would be inexpensive: The equipment required for implementing the interface would consist of a handheld device and an attached tracker system sensor. Thus, the price of a handheld system would be just a fraction of the equipment costs of a complete virtual reality system. Additionally, as the increasing use of powerful devices with graphic interfaces and abundant processing power become more common in daily life, any handheld device could be integrated into a virtual reality system to support interaction.

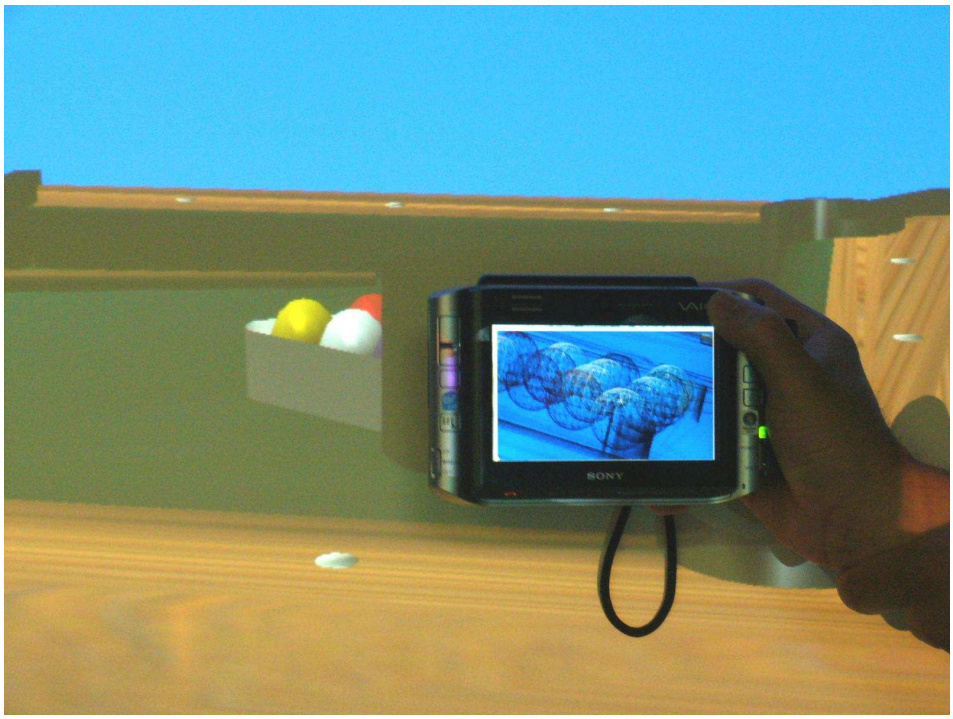

Figure 1.1: The magic-lens showing the see-through the lens effect.

Since the use of isolated interaction techniques limits the usability of the applications, a combination of several techniques are normally integrated to provide a wide variety of interactions. The combination of interaction techniques stimulates

creativity in developers searching for novel forms of interaction. However, offering an excessive number of interaction techniques tends to confuse the user. Therefore, a set of fundamental interaction techniques that can be applied under different contexts, while offering the same level of functionality, is desirable. In this research, the use of the magic lens metaphor in combination with other interactions that minimize its limitations is explored. Magic lenses have been extensively used in virtual and augmented reality applications [Bier 1993, Brown 2006, Looser 2004, Stoev 2002]. The magic lens implementation proposed in this research enhances its functionality.

The overall goal of this research is to implement an interface based on the magic lens metaphor using a handheld device. This goal is divided into the following three sub-goals.

- The first short term sub-goal is to study the communication mechanism between the virtual reality system and the handheld device necessary for implementing the magic lens effect [Bier 1993]. A previous study related to computer graphics cluster technologies analyzes the possible rendering models for distributing rendering processing among cluster nodes [Chen 2001]. Although the study does not consider heterogeneous environments composed of devices with varying performance or the capture of interaction information, it provides important insight into the model type more suitable for the interface. The distributed pixels model was selected for exchanging information between the rendering computer and the handheld device, implemented as a video stream. A study on distributing graphics by a stream is described in [Lamberti 2003]. The interface proposed in this dissertation adds a synchronization mechanism for providing real-time interaction.

- The second short term sub-goal is to create interactions with the "active" magic-lens implemented on a handheld device. The term "active" refers to the flexibility needed for attaching different functions to the magic-lens through programmed 2D interfaces displayed on the mobile display. Previous works considered the use of dumb tablets [Szalavári 1997] or Plexiglas sheets [Schmalstieg 1999] for implementing the magic lens effect. However, these implementations only provide tactile feedback information to the user and separate the tablet and 2D graphic interface, resulting in limited interaction.

  One of the most remarkable limitations of previous interfaces, including ray-casting [Bowman 1997b], Go-Go [Poupyrev 1996] and HOMER [Bowman 1997b], is the loss of accuracy for small and distant objects [Poupyrev 1998b, Bowman 1999a]. Since the interaction occurs in 3D space, a hand movement used to manipulate an object will result in different distance displacements depending on the distance between the object and the user. For example, in ray casting, the user handles a ray. If the object is near the user, a hand movement represents a small displacement. However, as the distance between the user's hand and the object increases, the displacement distance increases as well. For small object selection, the problem experi-

enced with previous interfaces is that the visual cues become more difficult to distinguish. The interface proposed in this work provides a more accurate method for selection and manipulation by using snapshots.

- Finally, the third short term sub-goal is to build a set of generic interaction techniques implemented around the magic lens metaphor. Obtaining maximum flexibility and extensibility are two concepts the interface considers in its design.

1.3 Contributions.

The originality of the study is the implementation of the magic-lens metaphor [Bier 1993] on a handheld device. The integration of several metaphors in the interface make possible a novel exploration of 3D interfaces. Handheld devices have been employed for providing text input, stroke-drawing [Ayatsuka 2000], control information [Kruijff 2003, Hartling 2005, Watsen 1999, Kukimoto 2005], and for augmented reality [Schmalstieg 2002]. The personal interaction panel (PIP) study [Szalavári 1997] provides a rich set of interaction techniques implemented with "dumb" elements, such as a flat panel and a pen. The same study states that by using a "pressure-sensitive flat display with pen", such as handheld palmtops, the interaction capabilities can be improved even more. However the study does not expand on that concept.

Integration of a mobile device into a virtual reality system infrastructure provides several advantages such as:

- Providing a programmable or "active" prop: The tactile feedback information provided by a prop [Schmalstieg 1999, Szalavári 1997] enhances the interaction since its adds additional stimulus. The proposed interface extended this idea since the prop does not act passively as a transparent surface. It can also display graphics, capture and process interaction information, and even provide other stimulus, such as vibration.

- Improving accuracy: The interface provides for the selection and manipulation of small and distant objects. It extends the concept proposed in the image-plane interaction technique [Pierce 1997]. The selection is performed with the support of snapshots. The original image-plane technique uses gestures to capture snapshots, which are then presented in virtual reality in front of the user, occluding the area covered by the snapshot. Implementing the snapshot on the handheld device occludes the back part as well, but it is easier to manipulate because it is contained inside a real object, not inside the virtual space. Conceptually, the handheld device operates like a magic photographic camera. The user takes a picture and the elements contained in the picture are then subject to manipulation. Additionally, a camera is capable of storing numerous pictures to be reviewed later. The same metaphor is used by the handheld device that stores snapshots in a clipboard for future reference.

- Extending 3D interactions with the support of a handheld device: Previous studies have explored the introduction of handheld devices for creating 3D interaction [Ayatsuka 2000, Kruijff 2003, Hartling 2005]. However, the flexibility offered by handheld devices have not been totally explored. The graphic display, touch screen, sound, vibration, and networking, can all be combined to build richer interaction techniques.

1.4 Organization of the dissertation

The rest of this dissertation is structured as follows. Chapter 2 presents the technologies used to implement immersive virtual reality systems, with a special emphasis on input and output devices. A more detailed description of previous 3D interface studies is provided as well.

Chapter 3 presents the proposed approach for implementing a magic-lens effect with a handheld device. It starts with a description of different models used for distributing graphics processing between handheld device and the virtual environment rendering computer. Then, a description of an architecture used to provide the magic-lens effect and capture the user's interaction information is presented. The final part presents the evolution of an implementation with performance testing.

Chapter 4 describes an architecture used to create interaction metaphors for virtual environments with the handheld device. The requirements for a software layer that encapsulates a generic interaction task is discussed. Then, the architectural design of a layer for creating interaction is presented. Finally, the implementation of pevious well-known 3D interfaces, and the combination of interaction around the magic-lens, are presented.

Chapter 5 comprises the evaluation of the interface using the developed prototyped, its discussion, and results.

Chapter 6 presents an overall discussion of the interface, its evaluation and possible extensions.

Finally, conclusions and future work are summarized in Chapter 7.

Chapter 2

Background

2.1 Introduction

With virtual reality, it is possible to explore environments and phenomena that are difficult or even impossible to experience in the real world. The main appeal of virtual reality is the possibility of experiencing alternative realities simulated by computer systems. Hardware and software in collaboration with special devices stimulate the user's sensory channels in order to emulate similar sensations which the human brain is able to interpret as authentic stimulus obtained from the real world. Similar to desktop computer systems, the special devices are classified into two categories. Input for devices capturing information to be processed by the computer system and output for devices able to provide information to the user. Input devices capture information to provide for the interaction of the user with the virtual environment. The information is processed to modify the environment accordingly and simultaneously generate new stimuli, produced by the output devices, which reflect the new environmental condition after the interaction took place. In order to offer a pleasant and realistic experience, the time elapse between the interaction and the modification of the environment must be near zero, instantaneously. Real-time interactivity is a key element in virtual reality systems.

In addition to the input devices, the user requires supplementary communication channels which let them manipulate and control the simulated objects and environment represented in the virtual reality system. The communication between computer systems and humans is assisted by interfaces. Since there are more variables and factors to consider, the development, implementation, and evaluation of interfaces for 3D environments is more complex and sophisticated than their 2D counterparts. Although several 3D interfaces have been proposed, the definition of a generic set of interfaces or methods of interaction does not exist. The creation of 3D interfaces is still a complex process.

This chapter describes the technologies employed for implementing immersive virtual environments and their drawbacks. The methods proposed for interacting in the environments represented by them. The efforts for creating interfaces designed for three dimensional spaces. And the kind of interactions that are possible with those interfaces and their limitations.

2.2 Immersive Virtual Reality

Virtual reality is the computer technology which creates responsive synthetic worlds that look like real ones. The simulated worlds are dynamic since the user is able to modify the condition of the world based on interaction information provided through input devices. The modifications in the artificial world occur in accordance with the input information instantly, inducing on the user the sensation of being inserted in the environment. The final goal of virtual reality technology is to produce on the user the sensation of being totally immersed in a synthetic world [Sutherland 1965]. For virtual reality applications, the term immersion refers to the state of consciousness that the user experiences when he or she loses the self reference with the real world to be substituted by a sense of presence in a artificially created world. To offer total immersion is still a goal virtual reality has not reached completely [Brooks 1999]. However, the ideas and concepts developed by virtual reality researchers have been applied successfully regardless of its limitations. For example, most video games rely on the real time interaction techniques employed in virtual reality. The player feels like being part of the game, even though the user is seeing images displayed on a TV screen or computer monitor, and using a joystick or other handheld control. Other examples are the VRML [Carey 1997] and X3D [Brutdzman 2007] specifications, which represent attempts of creating virtual reality application through the Internet. Although the two above mentioned examples are considered virtual reality applications, since the user sees 3D images on a flat screen monitor and interacts with the system through keyboard, mouse or joystick, the user still feels as if he/shes is interacting with a computer, limiting the immersion feeling.

A system is considering immersive if it provides depth clues that stimulates the perception of 3D spaces, if it is able to track 3D positions and create a correspondence between virtual and real spaces, and if it provides supplementary stimulus on others senses such as sound, touch, smell, and flavor. In addition to the technological aspects, the level of immersion is influenced by psychological and subjective factors. But in this dissertation, the term immersion is restrained to systems capable of offering visual immersion and able to track position in virtual space.

2.3 Input Devices for Implementing Immersive Virtual Environments

Keyboards and mice are not restricted to 2D interaction. They can be used within immersive virtual environment. However, their use is restricted to systems that offers a flat surface where these input devices can be placed. They are not suitable for immersive systems such as a head mounted display or CAVE which offer freedom to move and walk inside the system boundaries. Therefore, a different kind of input device is recommended for these systems. There are plenty of input devices providing different types of information such as joysticks, gloves, styluses, buttons and so on. This diversity carries a drawback. Since each input device can be used to

implement a different set of interactions, if the application is complex and requires more interactions than are included in a set, the user is forced to change the input device, and a place where unused devices can be placed becomes necessary. A flexible device able to alleviate this problem is a handheld device. It can be handled in one hand, provides pen-tablet interaction, provides many functionalities, and with the introduction of 2D interaction, some 3D task become easier to perform.

An important input device in immersive virtual environments is the tracking system. This device senses the location and position of objects in 3D space.

2.3.1 Handheld Devices

An input device able to modify its properties according to the performed interaction offers great flexibility for the implementation of interactions techniques. PDAs and handheld PCs and other handheld devices provide a graphical touch screen, physical buttons, sound, and networking. On top of that, they are able to perform computation and can store information by themselves. Previous works have introduced handheld devices for virtual reality systems [Ayatsuka 2000, Kruijff 2003, Hartling 2005, Watsen 1999, Kukimoto 2005], and for augmented reality [Schmalstieg 2002]. However new interactions are able to be implemented using handheld devices in novel ways. The work described in this dissertation explores the use of a handheld device as a magic lens [Bier 1993, Viega 1996], extending the functionality of the original proposal with the combination of the capabilities that mobile devices offer.

2.3.2 Tracker Devices

The tracking system devices are equipment able to sense spatial three dimensional positions and orientations. For their implementation several technologies have been employed such as magnetic, acoustic, mechanics and optical.

After fixing an origin position, the tracking system is calibrated to set up a controlled 3D coordinate system. Then, sensors register their position and orientation with respect to the coordinate system at regular periods of times. Positions and orientations are represented mathematically as 3D vectors and matrices respectively. This information is sent to a computer in order to evaluate the future changes the environment will suffer based on the information provided by the sensors. The problems and limitations of tracking devices depend on the technology employed, and they will be discussed in the interface implementation.

2.4 Output Devices for Implementing Immersive Virtual Environments

The Head Mounted Display [Sutherland 1968] and the CAVE [Cruz-Neira 1993] are two examples of systems able to produce visual immersion.

2.4.1 Head Mounted Displays

There are several implementations of the head mounted display (HMD). The most widely used consists of two miniature independent screens placed one in front of each human eye. The screens are mounted in a structure similar to glasses or a helmet in order to keep them in place. A computer system is in charge of generating separate images for the left and right eyes simultaneously. With both images, the human brain is able to generate three dimensional perception through a process called stereopsis. The area of the brain specialized on vision compares the position of the objects contained in both images. Then based on the differences, the brain is able to make an interpretation of the distance or depth the object is located at with respect to the position of the eyes. Therefore, near objects have a big difference between both images, while far objects have small difference between both images.

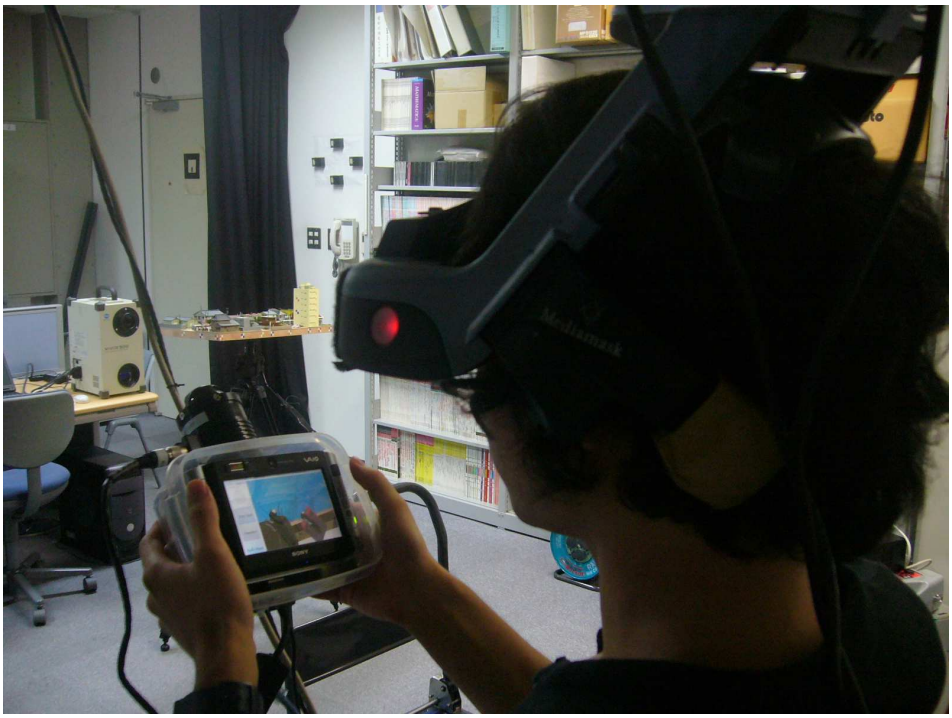

Figure 2.1: A user wearing a head mounted display.

One tracking sensor is attached to the head mounted display in order to sense the user's point of view and render the images accordingly. Rendering is the process of generate synthetic three dimensional computer graphics images, generally employing special graphics cards in combination with graphics libraries such as OpenGL [Shreiner 2004] or DirectX [Gray 2003].

Head mounted displays present some drawbacks which negatively impact the design of interactions. Visual clues produce discomfort and sometimes lead to depth perception errors. The images are presented on both displays. The displays are

separated from the eyes at a constant distance. Virtual objects are rendered according to their location in the virtual reality which rarely coincides with the displays position. Therefore, instead of focusing on to the object position, the eyes tend to focus on the display. Another problem present in most head mounted displays is narrow field of view. Field of view (FOV) refers to the angle of aperture that can be seen on a display. The maximum field of view human eyes are able to perceive is around 200 degrees, while a head mounted display offers a FOV between 30 and 60 degrees. Therefore, the head mounted display limits spatial perception, as only a narrow portion of the environment can be seen at a time. With head mounted displays the user is able to experience complete visual immersion. But, some head mounted display occlude the real environment. Therefore, they are not suitable to be used with a handheld device. See-through head mounted display designed to be used for mixing real and virtual worlds represent an alternative that can work with handheld device interaction.

2.4.2 The CAVE

The CAVE [Cruz-Neira 1993] is another technology which accomplishes the above mentioned visual and spatial immersion. It consists of a small cube-shaped room. Most implementations employ only four faces. The faces corresponding to the roof and one wall are omitted. Projectors located on the back of every wall generate the images which the user inside the CAVE is able to see. The floor is the only face where projection is generated from the front. For that reason the roof is omitted. The other omitted face corresponds to the front wall which is used as an open door for entering the room.

A tracker system is used inside the CAVE in order to sense the user's head and objects. The total number of projectors depend on the technology employed and the images' resolution. Two kind of projection technologies are widely used for creating stereopsis: active and passive stereo. Active stereo projectors display right and left images alternately at high refresh rate. Refresh rate is the speed a visual display device updates the image. LCD shutter glasses are used for blocking and letting pass light into the eyes in synch with which eye image the projector is displaying, so when the left image is displayed only the left eye can see through the glass while the right eye is blocked and vice versa. Passive stereo projectors use polarization or spectral multiplexing for separating left and right images. In polarization, special filters mounted on the projectors are employed to modify the oscillations of light in horizontal or vertical orientations. Then, special glasses let light pass through them according to the kind of orientation each eye is assigned to. In spectral multiplexing, the images are displayed simultaneously with different colors. Color filter glasses cut out different colors according to the filter sensitivity.

Similar to the head mounted display, the CAVE has problems with visual depth clues. Although the objects are projected according to their virtual position, the user tends to focus on the CAVE's walls. Another problem is the resolution. Since CAVEs are projector-based technologies, the area to be projected on is big compared,

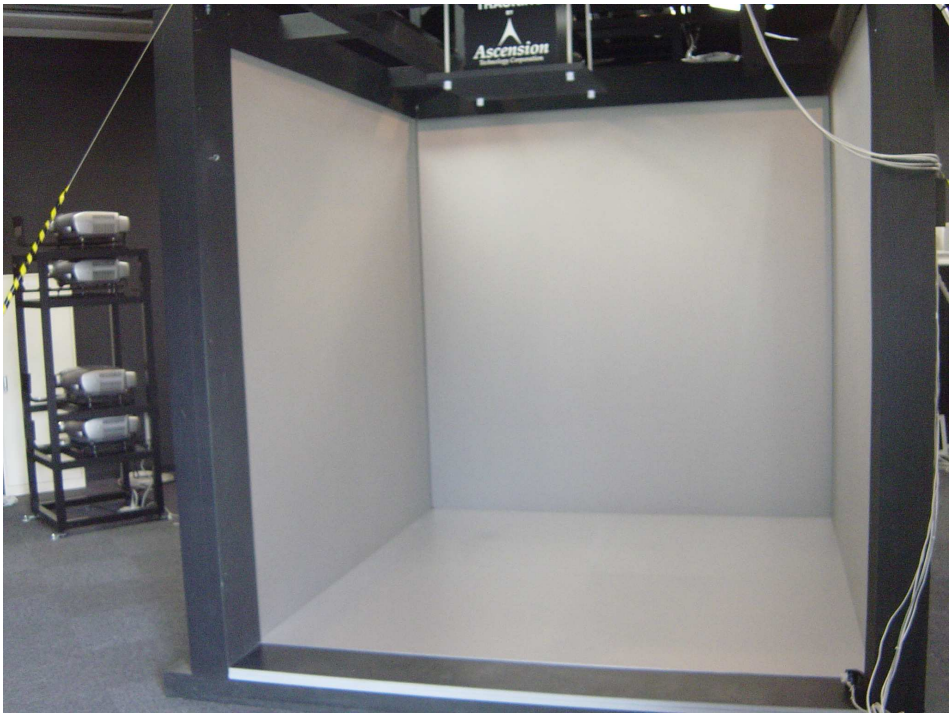

Figure 2.2: The physical setting of a CAVE system.

for example, to a monitor screen. In order to cover the entire wall area, the pixel size is increased, making them noticeable. With active stereo technologies, the shutter glasses requires a synchronization signal in order to determine the open/close sequence of each lens. If the signal is interrupted, occluding the sensor or the lens, the 3D effect is lost.

2.4.3 Others Technologies

Workbench systems [Kruger 1994] provide an immersive experience with the aim of offering augmented interaction. As the user is working on a table, the combination of virtual and real objects is more natural. Its use is recommended for tasks where interaction with real elements is essential. Hemispherical display systems [Shibano 2001] use special software and optics to display images onto curved surfaces. The images are adjusted or distorted by the system. The curvature of the projection surface enhances the user's depth perception.

2.5 Interaction in an Immersive Virtual Reality System

An immersive virtual reality system requires special methods to provide interaction. The system requires methods to define how the user moves and walks in the environment, how to perform a selection, movement and modification of objects,

Figure 2.3: The CAVE system.

Figure 2.4: On the left, a spherical screen display. On the right, a personal-scale spherical display

and how to provide commands. The most intuitive approach is to emulate the interaction methods employed in our real world. However, this approach can not be applied most of the time due to the limitations imposed by the input and output devices and the lack of perceptual information. For example, a user in a CAVE is able to see objects in three dimensional space, but it is not possible to grab them because they are merely projections. They do not have a physical existence. There are research efforts trying to simulate touch and force. However, haptic technologies have not been reached miniaturized enough to implement in light, small size devices able to be carried with comfort, without obstructing the interaction in the virtual environment.

Even if the technology would emulate haptic forces, a system that only replicates the interaction methods present in our real world would be limited to the same restrictions. It is desirable to create "magic" interaction methods to augment limitations present in reality. An interface offering good interaction techniques will impact positively on the quality of the virtual environment applications. 3D interfaces represent an essential element for creating successful virtual reality systems.

2.5.1 3D Interfaces

There is no standard mechanism for interacting in a virtual reality system. Researchers have proposed several methods, proving their effectiveness in particular applications. However, there is no one set of interaction methods considered to be fundamental. Researchers who design 3D interfaces are continuously searching for flexible metaphors which provide the same generalization offered by a 2D desktop in a 3D space. However, 3D spaces present more complexities. The user moves in three dimensions, with six degrees of freedom. This freedom of movement represents an advantage but on the other hand, it carries some inconveniences specifically when manipulating an object with precision. The diversity in input systems is another problem. The plethora of input information determines and in some cases limits the kind of possible interactions, making their standardization difficult. Depth perception and occlusion are other problems not present in 2D but they comes up in 3D spaces.

2.5.2 Selection and Manipulation

The most basic operations to interact with an virtual environment are selection, manipulation and navigation. Selection gives users the power of picking up objects contained in the environment. It is the operation that identifies and grab a specific object from a set of objects contained in an environment. Manipulation gives the possibility to modify the object's position, its orientation or even its geometric properties. In the real world, manipulation consists of handling the physical object by hand. The properties of the object influence what kind of manipulation can be applied on it. So a liquid or gas object requires a container while soft material objects requires taking into account shape variation. In this dissertation, we narrow

the manipulation to rigid objects, or other words, objects that preserve their shape. Navigation provides the capability of moving freely around the environment. Manipulation and navigation are closely related. Some navigation techniques can be built using manipulation techniques. For example, grabbing the air and pulling with the hand can represent the action of moving the view position toward the direction of the hand. Some researchers consider manipulation a more fundamental task.

2.6 Interaction Techniques for 3D Manipulation

The creation of interfaces for 3D virtual environments has been a very active research area. In order to study the similarities and difference among interfaces, several classifications have been proposed. In [Poupyrev 1998b], interfaces are classified in two groups based on the position of the user respect to the virtual environment. In the exocentric metaphor group, the user perceives the environment from the outside like the view of God. In the egocentric metaphor group, the user interacts from inside as being an element contained inside the environment.

2.6.1 Exocentric Metaphors

The most representative interface in this group is Worlds in Miniature (WIM) [Stoakley 1995]. This interface provides a small scale 3D map view of the entire virtual world (See Figure 2.5. Selection and manipulation are performed using the scaled down object contained in the map. The WIM performance decrease as the size of the environment increases. In other words, when the virtual environment is large the scale map is so small that it is difficult to distinguish small objects, precision of manipulation is also reduced because a small distance in the map represent a long distance in the entire environment.

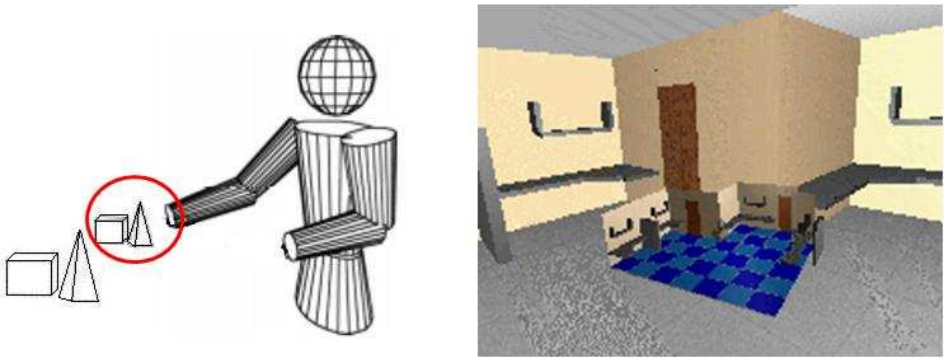

Figure 2.5: On the left, the World-in-miniature interaction technique. On the right an example applied to a virtual room [Stoakley 1995].

2.6.2 Egocentric Metaphors

Two subgroups conforms to this metaphor: the virtual hand metaphors and the virtual pointer metaphors. Go-Go hand [Poupyrev 1996] is the most significant example of the former. In Go-Go, the hand position is extended using a logarithmic function. The interface predefines a threshold distance, around half the complete extension of the arm. The Figure 2.6 shows this interaction technique.

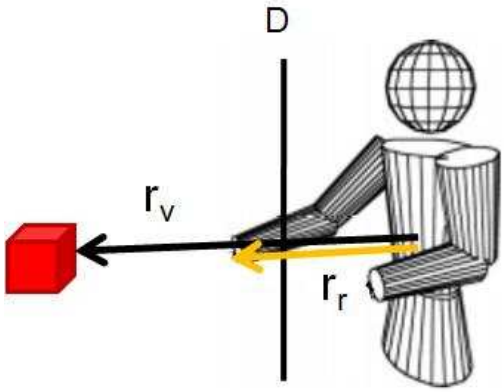

Figure 2.6: The Go-Go interaction coordinate system [Poupyrev 1996].

When the user's hand is located inside this threshold the virtual hand moves as a real hand does. But when the hand is extended outside the threshold distance, a logarithmic factor is multiplied to the real hand position in order to extend the virtual hand further. The main problem of the Go-Go interface is that its precision decreases as the distance is elongated.

The mathematical expression of this behaviour is defined as

$$r_v = F(r_r) = \begin{cases} r_r & \text{if } r_r \leq D, \\ r_r + \alpha(r_r - D)^2 & \text{otherwise.} \end{cases} \tag{2.1}$$

Ray-casting [Bowman 1997b] and Image-plane [Pierce 1997] are two examples of the later sub group of metaphors. In ray-casting, a ray starting at the hand position is extended along the tracker device orientation, representing a 3D pointer. When this ray intersects an object the user is able to select the object using a button or a gesture. The Figure 2.7 shows this interaction technique. Ray-casting is an effective selection interface. However, it does not show good performance when high-precision selection is required, for small or far away objects [Bowman 1999a, Poupyrev 1998b]. Manipulating objects with ray-casting is difficult due to the restricted range of movement provided by the interface.

The principle of the Image-plane interaction metaphor is to restrict the interaction space from 3D into 2D space by using snapshots.

With this interface, the user is able to capture 2D views or snapshots from the environment using gestures. Selection and manipulations are performed indirectly

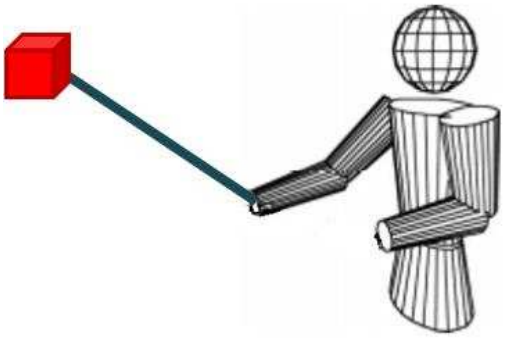

Figure 2.7: The Ray-casting interaction technique [Bowman 1999a].

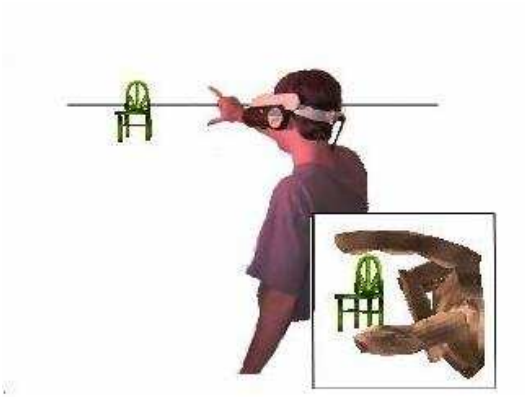

Figure 2.8: The user is able to select 3D objects using gestures sensed by a globe [Pierce 1997].

using those snapshots. Selecting objects from a 2D picture is less prone to ambiguities than selecting directly from a 3D space because all movements are restricted to two dimensions. Manipulations such as translations and rotations are easy as well. A movement on the 2D projected object produces an equivalent movement in 3D space. Because the Image-plane technique simulates direct touch, it is easy to use and intuitive. However, it is difficult to control the distance between the user and the 3D object.

An interface which combines the interactions in the two above-mentioned subgroups is HOMER [Bowman 1997b]. HOMER stands for hand-centered object manipulation extending ray-casting, and it is a two step interaction technique. First ray-casting is used for selection, then, the interface switches to a hand metaphor mode for manipulating the object.

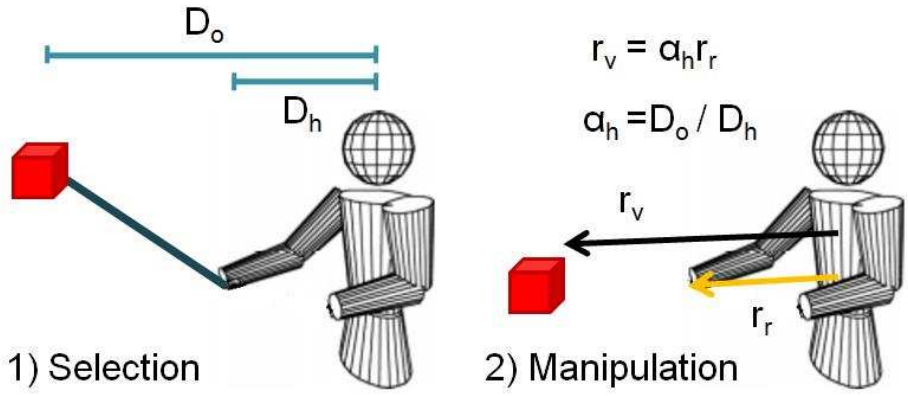

Figure 2.9: On the left, the user selects object using ray casting. On the right, for the manipulation the interaction switch to a virtual hand with a linear factor modifying the movement range.

On manipulation mode the hand's movement is determined by a linear factor, proportional to the object distance divided by the hand distance with respect to the user.

$$r_v = \alpha_h r_r \tag{2.2}$$

where:

$$\alpha_h = \frac{D_o}{D_h} \tag{2.3}$$

Although selection on HOMER is very effective since it relies on ray-casting, the manipulation suffers clutching problems. When the object selected is near, the manipulation factor is small, producing a short range of manipulation, and in the opposite case, if the object is far, the factor would be big, and it is difficult to manipulate the object with precision. To overcome this problem, the user is required

to progressively manipulate the object two, three, or more times in order to modify the manipulation factor and produce a manipulation movement that helps reach the final position.

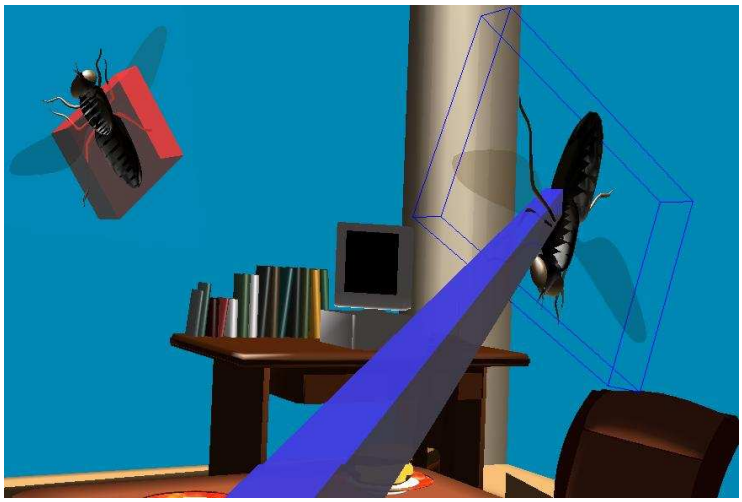

Figure 2.10: The selection in HOMER relies on the ray-casting metaphor.

2.6.3 The Magic Lens Metaphor

The Magic lens metaphor [Bier 1993, Viega 1996] does not fall in one of the above mentioned classes, however, it has a central role in our development. The magic lens metaphor offers the capability of exploring and manipulating virtual objects in a very natural and intuitive form, emulating the interaction employed in the real world with an augmented lens. With the lens the user is able to see different view of the environment. The Figure 2.11 shows two applications of magic lens.

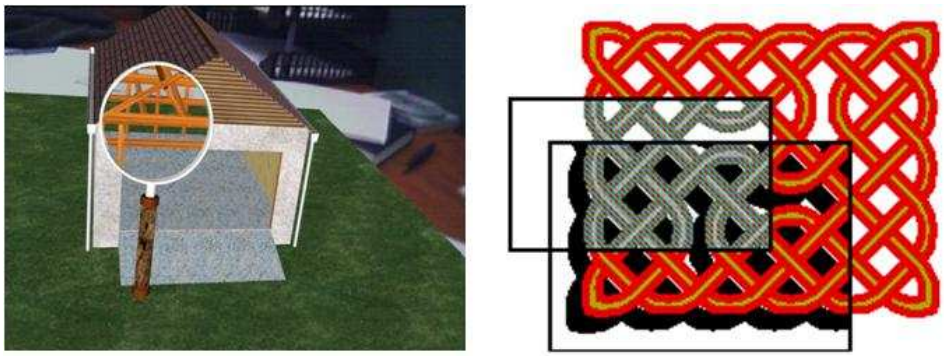

Figure 2.11: On the left a magic lens is used for exploring the structure of a house, on the right combination of magic lenses are used to generate new effects[Bier 1993].

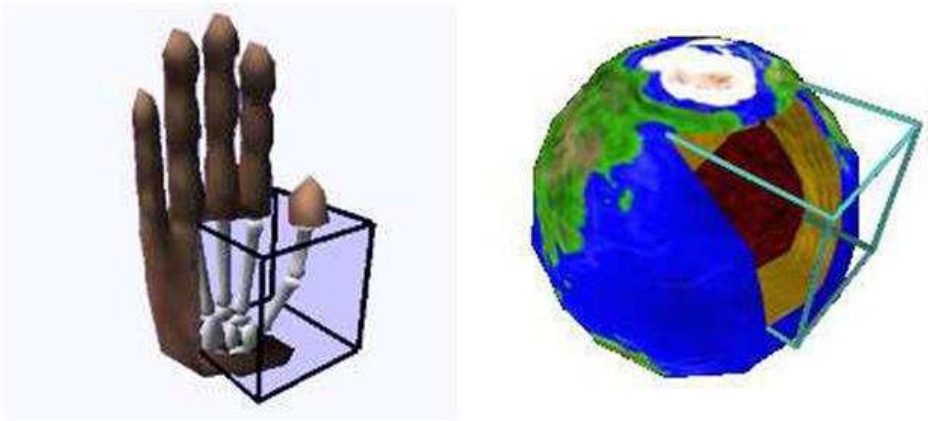

Figure 2.12: Two applications using a 3D Magic lens [Viega 1996].

The interface comprises two elements: a transparent glass-like window and a tool palette.

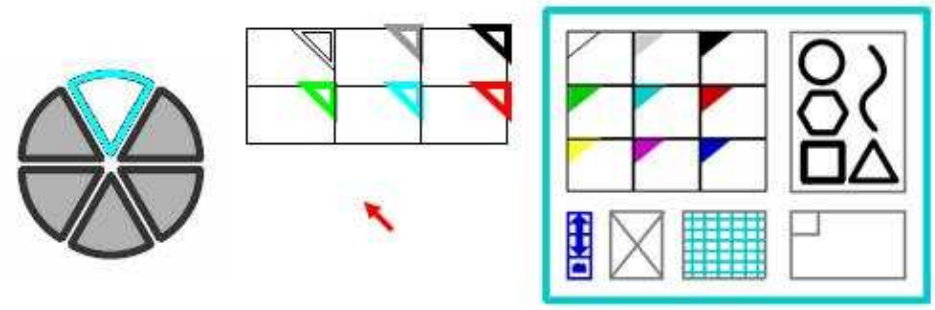

Figure 2.13: On the right, the tool palette. In the center, the window. The window is divided into cells. In the small triangle in the upper part of every cell, the border color indicates how the window is able to change the properties of the object. The red pointer helps to validate the modification of the object. On the left, the result of applying the operation on a portion of the circle [Bier 1993].

Users explore or modify virtual objects by covering them with the window. The window transparency lets the user see through it. The tool palette contains a set of operations that can be applied onto the virtual objects. In order to perform a modification to some objects, the user first selects the desired operation from the tool palette. Then, the glass-like window gets the capability to apply the selected operation over the objects that are covered by it.

The magic lens metaphor, as is described in [Bier 1993], was originally implemented as a virtual tool, in other words, the elements conforming the interface are computer simulated objects contained inside the virtual world with the same limitations such as visual clues, occlusion problems an so on. The usability of the

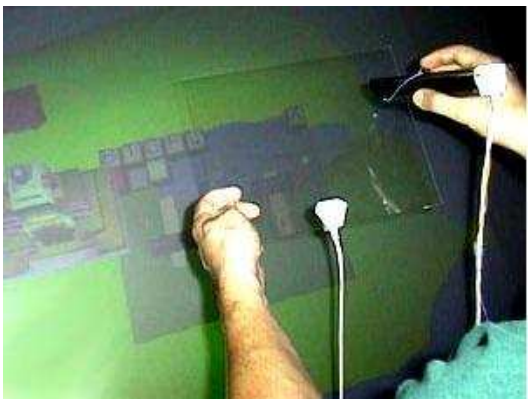

Figure 2.14: The transparent prop consist of a plastic square with a tracking sensor attached to it [Schmalstieg 1999].

magic lens metaphor can be extended by implementing the glass-like window using a real transparent plastic window with a tracking sensor attached to it. This method is called passive haptic feedback [Hoffman 1998] or props and consists of matching virtual objects with real object counterparts that have similar shape and appearance. A virtual reality system implemented with props induce on the user the illusion of both seeing and "feeling" virtual objects. The personal interaction panel [Szalavári 1997] is an interface employing a prop in combination with a Virtual Table. A plastic tablet and a pen, both tracked by individual sensors, are used for providing the haptic feedback. Controls and interaction information is projected on the tablet surface, and the interaction is performed through the movements of the pen over the controls. A similar approach employs a physical transparent glass as a prop [Schmalstieg 1999]. Controls are displayed on the Virtual Table. The limitation of both approaches is the use of dumb props. The tablet is designed to provide exclusively passive haptic feedback. A much richer set of interactions is possible replacing the prop with a programmable device.

2.7 Summary

The creation of computer simulated worlds opens up the opportunity for exploring phenomena which are difficult to accomplish in reality. Virtual reality is a powerful tool that stimulates human curiosity and creativity.

This chapter describes the most widely used technologies for implementing immersive virtual environments with the aim of identifying the drawbacks that impact the design of interfaces.

Although these technologies are not able to produce a totally immersive experience, advances in visual 3D technologies and 3D tracking systems have made possible the development of applications which the support activities of simulation, research, training, and education.

The possible forms of interaction in an immersive virtual environment are more diverse than their equivalents in 2D desktop systems. Movement in 3D space, visual clues, occlusions, and the lack of physical touch are factors which make the design of interfaces complex. The chapter mentions previous works that implement interfaces for selection and manipulation of virtual objects, along with their advantages and limitations.

The interfaces presented in this research attempt to alleviate the drawbacks and limitations found in the works mentioned in this chapter.

Chapter 3

Architecture for Implementing a Magic lens Interface Using a Handheld Device

3.1 Introduction

Interaction between user and computer system requires the use of interfaces. Such interfaces provide the communication channels through which the user and computer exchange actions and reactions. To perform their role efficiently, interfaces must be stable, reliable, and have a rapid response time. To some degree, they should be imperceptible in the sense that they must not interfere with the activity that the interface is supporting. In other words, they should promote the concentration of the user on the tasks she/he performs.

For 3D immersive virtual environment systems, the interfaces must face the problem of handling a large set of complex input and output information. On early virtual reality systems, this was the main restriction for obtaining a "realistic" effect [Brooks 1999]. However, with the evolution of tracking technologies, capturing motion for real-time interaction in 3D has been made possible. A number of problems remain unresolved. These include the presence of physical wires, the size and weight of tracking sensors and input devices, and limitations in input/output technologies.

This dissertation explores the use of a magic lens that can enhance 3D interaction. A handheld device is used for producing transparent window effect, displaying the portion of the image that it covers on the screen. At the same time, the handheld device captures the user's interaction information, which is send back to the virtual reality computer system to be processed, all in real-time. With the introduction of a handheld device, the lens becomes an active element. It takes advantage of the full set of 2D interactions implemented on the handheld device, giving them new purposes oriented toward 3D interaction.

This chapter describes the design and architecture of the magic lens effect used for implementation of the interface.

3.2 Interface Requirements

The implementation of a magic lens interface using a handheld device requires the integration of several technologies: computer graphics rendering, virtual reality frameworks, networking, and mobile interfaces design. The challenge is to integrate all

these components with a high level of stability and reliability, so that they can operate in real-time. The requirements the interface should fulfill are:

- Graphic resolution: The handheld device requires a graphics card powerful enough to display color images at the highest possible resolution (480*640 and above). Lower resolutions would produce poor interface-environment integration.

- Reliable and low latency tracking system: The interface requires the continuous and accurate sensing of the handheld device's position with respect to the virtual environment. Ultrasonic, magnetic, optical or other technology can be employed. However, it is important to consider the restrictions that the tracking system imposes on the handheld device. For example, a magnetic tracker can suffer from distortions produced by the metallic parts of the handheld device itself.

- Wireless networking with low latency: Wired technologies do not provide a good solution for implementing the networking layer because they restrict freedom of movement. A technology that uses a software-streaming layer should also provide low latency in order to produce the real-time streaming and command exchange required for natural interaction.

- A touch screen with a flexible graphical interface development platform: The commands issued by the user require a graphical interface implemented on the handheld device. Familiar and natural interaction is possible by employing a touch screen. Physical contact with the display surface of the mobile device provides feedback information that enhances interaction tasks, such as selection.

- A flexible and extensible software layer for 3D interaction: The design of interfaces is a process of successive stages of implementation and testing. A software layer that synthesizes and offers a set of pre-programmed interaction mechanisms would be a valuable tool for implementing and testing new interfaces.

- Long battery life: Battery life is a secondary point of the design, but a primary when the interface is used. For long interaction sessions, good battery support avoids interruptions needed for recharging batteries.

3.3 Interface Conceptual Design

The design philosophy of the interface follows the concept design of a magic lens [Bier 1993, Viega 1996] with several extensions. The user is able to see-through the lens. The lens can modify its effect based on the object it covers. The group of interactions the interface comprises are:

- The lens can modify visual properties of the environment, allowing for exploration based on different attributes, such the inner layer characteristics of the objects.
- Selection of virtual objects through lens contact: Object selection is performed on 2D images. If objects can been seen through the lens, the user can select them using a "pinch" action.
- Selection using snapshots: Much like a came, the lens can capture snapshots of the environment, after which, the user can select objects from the snapshots.
- Clipboard functionality: The snapshots are retained in memory storage for future reference.
- A flexible 2D control interface: Because the lens offers several interactions, a mechanism for changing functionality is necessary. This implementation of the magic lens integrates 2D control together with the lens and allows for the modification of available functions depending on the interaction status.
- Intuitive object manipulation: Manipulation is performed in 3D space. After an object is selected, the interface can manipulate the object in a wider area than possible with actual manipulation. An interaction metaphor that adds elongation is integrated to provide this characteristic.
- Additional tactile and audio feedback: Handheld devices normally provide vibration and audio. These elements can be added to the interaction to provide additional clues.
- Provides input text and annotations. Like [Poupyrev 1998a] the lens is able to capture text and handwriting annotations.
- Attach and review additional non-graphical information: Small marks located in the virtual space indicate the presence of additional information. By selecting such marks with the device, the information is displayed on the screen.
- Multiple viewing angles: The lens can select strategic camera positions and assign a virtual camera position. Then, the lens can switched to and remotely manipulate any of the predefined cameras.

3.4 Communication Model

The lens and the virtual reality projection system are separate components. They require compatible methods for communication and task assignment. In terms of implementation, both elements are computers with different capabilities. The lens is mounted on the handheld device (which has limited resources) while the virtual reality system is implemented on a high performance graphic computer.

3.4.1 Peer to Peer Model

In the peer-to-peer model, computers are grouped in order to share resources. Every participant in a group can establish direct communication with all other members belonging to the group, and, can become either a producer or consumer of resources. The model has advantages for pervasive applications. In virtual and augmented reality, it is used for collaborative environments [Nobuyuki 2005, Henrysson 2005] that address communication among devices with similar capabilities.

3.4.2 Client-Server Model

In this model, computers are separated into two groups, servers and clients. Servers are computers able to provide services, while clients are computers that make use of those services. Due to the asymmetric capabilities of the handheld device and virtual reality computer equipment, this model is most suitable for implementing the magic lens.

Figure 3.1 shows a diagram of the magic lens interface implemented on a handheld device. The tracking sensor attached to the mobile device senses its position and orientation relative to the virtual environment.

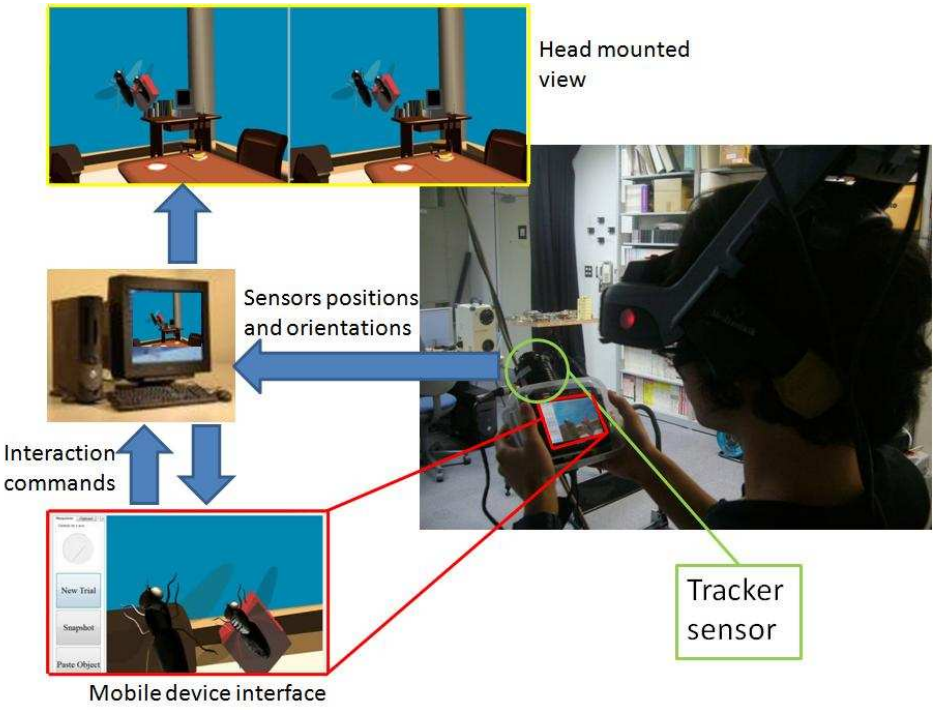

Figure 3.1: Architecture of the system

The position and rotation captured by the sensor is transformed into vector and matrix representation. Then, this mathematical information is transmitted to a computer that has the assigned role of a tracking system provider or server. The tracker server runs a background process that receives sensor information at predefined regular time intervals. The process is capable of opening and managing network connection ports in order to distribute all the sensor information to client computers. Using the matrix and vector describing sensor information, the render process is able to determine the handheld device position and render the environment accordingly. In addition to its position, the system require the physical dimensions of the handheld device. Therefore, its size must be predetermined in order to determine the area covered by the interface. The rendering system simultaneously creates two different rendered images, one for the virtual environment and the other for the handheld device display. A dedicated connection is established between the rendering system and the handheld device in order to exchange images and interaction information captured by the handheld device. These include touch screen selection, button pushing, and screen interface interaction.

3.5 Graphics Processing Distribution

The magic lens implemented on the handheld device provides an example of a distributed system. This can be defined as two computers with different processing, storage and display capabilities working together to produce a unified application. The processing load balancing and task synchronization are characteristics that impact the overall performance of distributed system. The division and assignment of such activities can be seen as the first step in the implementation of the interface. In [Chen 2001], a taxonomy of graphical data distribution strategies is presented. The study describes the possible strategies for distributing graphics work among nodes in a computer cluster. The taxonomy of the study provides the design guidelines for implementation of the graphics layer of the interface. Figure 3.2 shows a diagram of the strategies. The division in this taxonomy was established based on the assignment of rendering processing among nodes, and on the data to be transmitted between the components.

The following subsections describe the distribution methods in greater detail.

3.5.1 Control Distribution (Synchronized Execution)

In this model, there is a unique application stored in the memory of both computers, together with the virtual reality geometry database, that allows the execution to be performed simultaneously. The client captures events and sends the interaction information to the server. Because the interchange of

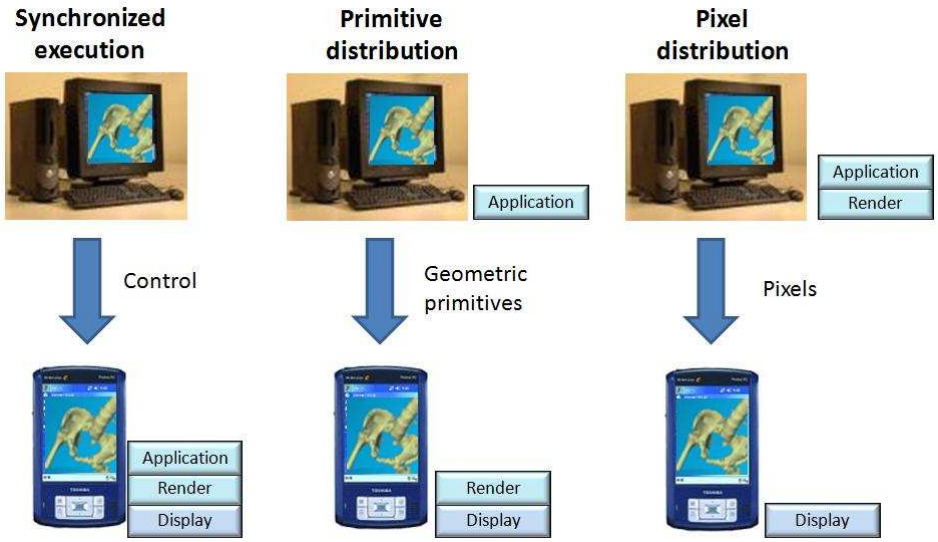

Figure 3.2: Distribution data models

information in this model is minimal, it does not require high network bandwidth. However, the storage and processing activities consume significant resources that limited handheld can not handle.

3.5.2 Geometric Primitives Distribution

In this model, the application is split into two sub-applications and the geometry database is stored only on the server. The server application extracts a portion of the geometry based on client demands, and the geometry description is transmitted over the network. The client (handheld device) stores a light version of the graphical application so that when a graphical modification is performed the client requests new geometry from the server. Then the server sends 3D graphics primitives (vertex, matrices) over the network to be processed and displayed by the client. The primitive distribution models offer a good method for graphic rendering load balancing. If the distribution is performed among PCs, it is possible to use OpenGL [Shreiner 2004]. Handheld devices incorporate light versions of the OpenGL graphics standard called OpenGL/ES [OpenGLES 2009], but, there are incompatibilities between the two. Although new OpenGL/ES implementations on handheld devices can render 3D geometry with textures, light sources, stencils and so on, the polygons are not considered to be primitive elements in OpenGL/ES. Therefore, it is necessary to preprocess the geometry in order for it to be presented exclusively as triangles before its transmission to the handheld device. As the environment increases in complexity, the bandwidth requirement increases as well. Chromium [Humphreys 2002] implements a graphic distri-

bution framework among nodes in a cluster. The bottleneck in Chromium is the networking layer, a situation that is alleviated by employing fiber optics for increasing bandwidth. However, bandwidth is limited when using wireless technologies.

3.5.3 Pixels Distribution

In this model, the server performs all of the graphic rendering and distributes pixels through the network using a compression technology, such as JPEG or MPEG. The model is simple and requires uniform bandwidth. In [Chen 2001], the author reports that the best trade-off between networking and processing corresponds to the pixels distribution model. Another important consideration is that handheld devices have evolved to the point of becoming complete miniaturized computer systems. This makes possible the creation of more complex applications, so that with time, it could be feasible to implement a primitive or even a control distribution model on a mobile device. Furthermore, as handheld devices become more powerful, software technology tends to unify the development process, making possible the transparent migration of an application among various platforms. On the other hand, it is important to implement technologies applied to limited resource devices because miniaturization promotes the integration of processors into a broader category of devices. After the appearance of PDAs, cellular telephones, and digital music players, technology advancements worked to integrate daily life devices with the aim of offering a more comfortable lifestyle to users. With hardware integration proceeding into even smaller devices, software must be adapted to the context and limitations of those devices. These include, the low memory, low power and limited processing capacity that are characteristics present in the smallest computer devices. In conclusion, implementing technologies oriented to devices with limited computer and memory resources increments the portability of the application onto a wider number of handheld devices. Because of this, the prototype presented in this dissertation follows a pixel distribution model.

3.6 Prototype Implementation

Three prototype devices were built with the purpose of validating our concept model. The difference between them is the technology used for implementing video streaming. In the first, an easy JPEG based streamer was implemented that produced high delay results. In the second, the JPEG and graphics support were modified with the aim of speeding up the application. Finally, in the third, a MPEG based architecture that provides a low-delay video transmission was created.

3.7 Design of the Server

During implementation of the magic lens interface, the server performs most of the intensive processing activities. It reeives and processes input events, generates renders for the virtual environment and handheld device, and synchronizes the overall application. The server design is shown in Figure 3.3. The design follows a layered structure in order to provide flexibility, and each layer can be implemented with different software technologies. (The technologies used for this implementation are shonw in parenthesis within the figure). On the bottom is the networking layer that handles the low level communication aspect between the server and client. Using networking support, the stream encoding layer generates the video that is transmitted to the client. The application layer consists of three basic components. The tracking system layer receives positions and orientations from sensors and provides that information to the graphics layer. The graphics is a component that relies on OpenGL for rendering all the images in the system. And finally, the application component is the module that handles the synchronization and management of all the elements.

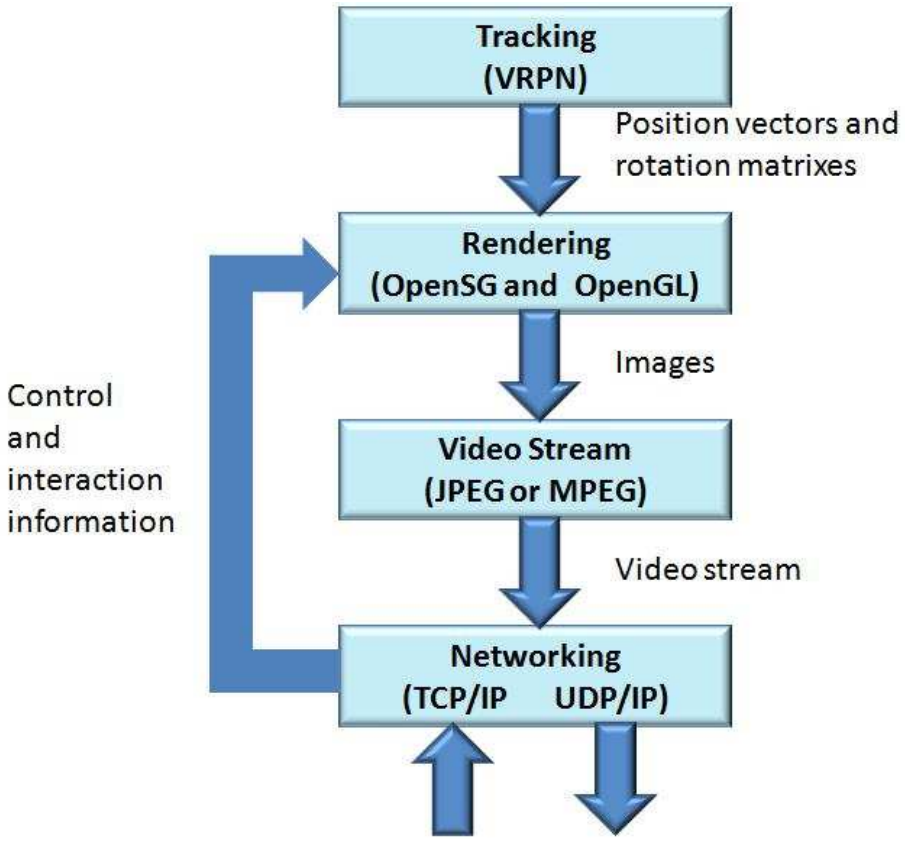

Figure 3.3: Block diagram of the server

Every layer in the architecture was implemented using a specialized software framework or library. The modular design allows for the use of various software technologies for each layer. The selection of the software libraries was made according to:

- Performance: libraries with efficient algorithm and modular design, showing good bench march results.
- Availability: open source projects offering ready to use libraries that are available for most operating systems.
- Flexibility: library source code modification and extension makes the development process easier and more robust.
- Portability: a library or framework that can run on different hardware architectures without code modifications, offers the possibility of executing applications on a more extended range of computer systems.

The following subsections explain the implementation of each layer in the architecture. Object oriented programming was the programming paradigm employed.

3.7.1 Networking layer

The use of distributing programming abstractions such as Remote Procedure Calling (RPC) [Stevens 1998] or Common Object Request Broker Architecture (CORBA) [Henning 1999] provide a unified layer oriented to integrating an application distributed across a computer network, as well as handling and hiding the low level details of the implementation. CORBA is a complex technology that offers mechanisms for distributing objects through the network, and is capable of combining various programming languages in the same application. The use of these technologies is attractive for the programmer because it reduces development time, and the differences among hardware and network technologies are handled by the distribution layer. On the other hand, the hardware requirements for CORBA implementations make it difficult to use on devices with limited resources. For handheld devices, the Fnorb [Fnorb 2009] python implementation of CORBA offers support for the WindowsCE [Boling 2003] operating system utilized on Window-based PDA's and mobile phones. Its performance is limited to implementing distributed objects. However, graphics support is not considered in the implementation. A standard Internet socket stack is present on most handheld devices. An advantage of this approach is that the application gains control over the transmission/reception process, which is important to maintaining a low latency over a network and, a key element to accomplishing real-time interaction. Therefore, the interface was implemented by offering basic services for UDP/IP and TCP/IP port creation and administration.

With the support of the network layer, the server applications open two ports in anticipation of the handheld device connection. One port handles video streaming while the other handles interaction commands. Outgoing communication transmits images and incoming communication receives control and interaction information. UDP/IP is a fast and unreliable protocol that is suitable for transmitting information that does not require confirmation on the reception side. The images displayed on the handheld device refresh constantly at high speed. Although some images are lost by the network layer, reducing latency in image refresh is the priority. Therefore, the application can tolerate a certain level of loss on image transmission. When a small number of images are lost, they can be replaced by previous successfully delivered images. In contrast, to guarantee the correct reception of interaction commands, the server uses a reliable TCP/IP connection. Commands are defined by small packets that are not sent as frequently as the images. Therefore, even though it is not as fast as UDP, the reliability of delivery is the priority. As a result, TCP/IP is considered most suitable. After a handheld device establishes a connection to the server, the network layer assigns two threads that handle all communication.

3.7.2 Streaming Layer Based on JPEG

For the image streaming the first prototype implemented was based on JPEG [JPEG 2009] technology. Figure 3.4 shows a diagram of the architecture.

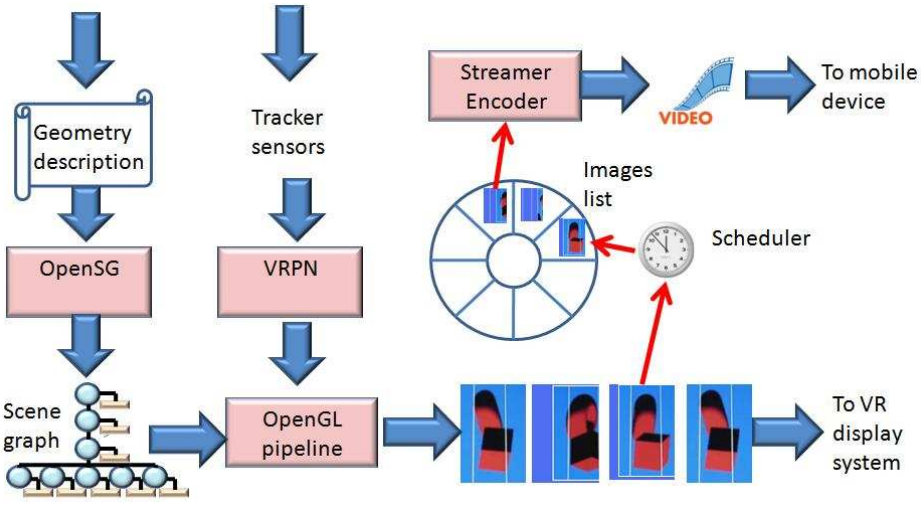

Figure 3.4: Detail of the server architecture.

A range between 20 and 30 frames per second is sufficient to produce a pleasant feeling of movement [Brooks 1999]. Recent developments in graphics card design have been marked by increasing frame rates and increased complex-

ity continuously rendered geometry. However, technology for mobile devices is generally not equal to technology for PCs. Thus, in order to produce a stream that can be displayed on the handheld device, a mechanism for reducing or controlling the frame rate employed to create the sequence of images is necessary.

The system implements a scheduler that is fixed against the computer clock. At regular intervals of 1/24 seconds, the scheduler executes a thread that captures the last rendered frame for storage in a circular list. A separate thread takes the images contained in the list and feeds them to a JPEG compression software module. The implementation of the circular list employs the producer-consumer method to synchronize the operation of both threads. The JPEG compression module is implemented using the open source Independent JPEG [IJEP 2009] library along with a function that stores compressed images directly into memory. The compressed images are organized in a second circular list. This second list is shared in a producer-consumer relationship where the compression module acts as the producer and the thread with access to the network transmission module takes the role of consumer. The design of the server is based on a pipeline architecture. Rendering, compression and transmission are performed in parallel.

3.7.3 Streaming Layer Based on MPEG

The JPEG implementation is simple and easy to use. However, because the compression is a frame by frame process, and due to the low speed decompression of some handheld devices, a delay time of about one second results. With the intention of reducing the compression/decompression time in the architecture, a second prototype based on MPEG technology was implemented. The MPEG specification is a set of standard methods for compressing and combining audio and video digital data into one stream. It is widely employed in DVD production, Internet video on demand, and multimedia applications.

The FFmpeg [FFmpeg 2009] library is an open source MPEG compliant implementation that was selected for the interface development because it is flexible, extensible, and has the ability to handle several open source codecs.

The interface employs the MPEG TS (Transport Stream) container over a UDP/IP protocol as is described in [Lamberti 2007, Lamberti 2003]. Figure 3.5 shows the architecture of the implementation.

The previously described scheduler is used to support the capture and storing of information produced by the OpenGL rendering module. At regular intervals of 1/25 second a thread takes the image produced by the render module and inserts it into an image circular list. Information regarding the frustum and the handheld device position are stored in a separate data list. The image circular list is read by the MPEG encoding module, implemented using FFmpeg. The use of two lists facilitates the synchronization of interaction between

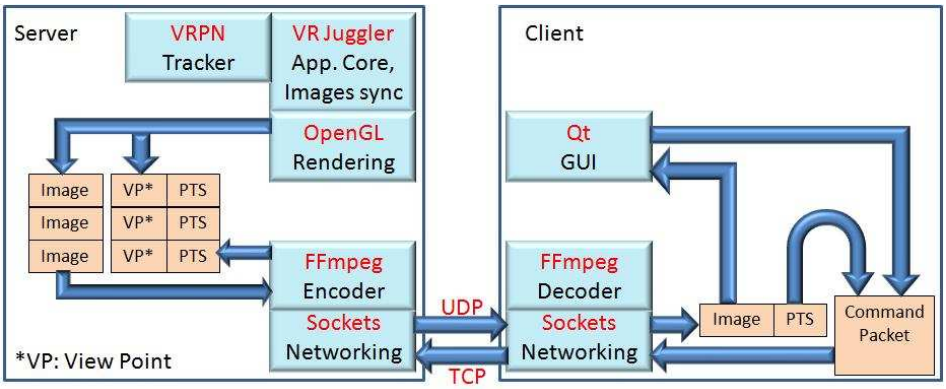

Figure 3.5: Architecture of the MPEG-based prototype

server and client. Images and data are stored in both lists, but they share a time stamp. With the time stamp, the server is able to render the same frame where the interaction took place again, thus avoiding the storage of every frame in memory. In order to assign a consistent and reliable time stamp, the implementation relies on the MPEG structure. To offer a flexible trade-off between data transmitted and processing time, the MPEG format specification defines three different types of frames; P frames (Post), I Frames (Intra) and B Frames (Bi-directional). The I-frames are complete frames corresponding to one image. These I-frames are employed as base frames for calculating the B-frames and P-frames, which are incomplete frames containing position differences among elements present between frames. By increasing the intervals between I-frames, or in other words, increasing the number of B and P frames, we are able to reduce the bandwidth, with the negative effect of increasing the buffering and processing time. Because the interface priority is to offer a close real-time experience, our application avoids the creation of B and P frames and only produces I-frames. This allows our aplication to receive the frame representing the exact position of every element contained in the virtual environment, as well as to reduce the buffering and processing time on the client.

The synchronization mechanism between the server and client consists of relating every image with the time stamp produced by the FFmpeg library. Every frame in MPEG includes two time stamps; the Decoding Time Stamp (DTS) and the Presentation Time Stamp (PTS). The former stamp registers the sequence of time each frame should be decoded and the later registers the time every frame should be displayed or presented to the user. The implementation takes advantage of the PTS, which is the continual sequence of integer numbers assigned to every frame by the codifier. These numbers are recovered after decoding at the client and determine the sequence the frames are rendered in the handheld device. When the user performs a selection of

objects on the images displayed on the mobile device, the PTS is captured and sent back to the server to request re-rendering of the frame. The data circular list mentioned above registers the frustum and handheld device position for every image destined to the handheld device. The encoder assigns PTS to the images, and their corresponding frustum and handheld device position. On the client side, when the stream arrives, we decode and check the PTS for every frame. When the user performs an interaction, the client collects the image PTS, the command to be performed, and the touched screen position. Then, it builds a packet that is sent to the server using a TCP/IP connection. Finally, when an interaction packet arrives at the server, the server checks for the PTS contained in the packet. Then, using this time stamp, the server searches for the corresponding frustum and user's viewpoint on the data circular list to be used when rendering the image again in a background process, and performs the interaction requested.

The server is responsible of storing and handling the information related to the generation of the 2D projection based on the handheld device position. Then, when an interaction event is captured on the mobile display, the time stamp indicates what images should be re-rendered again in order to apply the interaction requested on the correct frame. The server does not store every frame produced by the render module, instead the system re-renders the frames where the interaction occurred.

3.7.4 Tracking Systems

The prototype was tested with four different tracker technologies, depending on the facilities offered by the immersive system where it was tested. On a CAVE system located at Osaka University a magnetic ascension Flock of Birds (FOB) system equipped with four sensors and a wand was employed. The magnetic tracker uses a low frequency magnetic field emitted by a transmitter device, after which small sensors receive and determine their position and orientation relative to the magnetic source. Metals and magnets distort the magnetic field generated by the transmitter, resulting in inaccurate information, so it is important to isolate the tracker components from materials of this type. The FOB tracking range is 3.05 m, and the system updates tracking information at 144 Hz. The tracker uses a serial RS-232 connection to the computer to send the information at a baud rate of 115,200. The latency, which is the time that the system takes to report a change in the sensors, is 11.5 ms.

Using a Head Mounted Display Olympus MW601 system and an optical HiBall-3000 tracker system [Welch 2001] equipped with four cameras that provide the positions and orientations. In the HiBall system, the cameras are calibrated by aligning their relative position with an array of LEDs mounted on the ceiling. The LEDs emit light in a high-speed predefined sequence.

When the camera captures the light from the LEDs, camera position can be calculated by triangulation. The disadvantage of HiBall is occlusion. It is not possible to rotate the camera in the x-axis more that 90 degrees because the camera will no longer be able to detect the LEDs. The size (7.3 cm tall and 5.4 cm diameter) and weight (about 300 grams) of the camera are factors that can lead to user fatigue. The tracking range is defined by the ceiling mounted LED array. In the laboratory where the prototype was tested, the tracking volume comprises an area 3 m square and 2 m high. The update rate is 500 Hz for each camera. The tracker employs an Ethernet network connection to provide information to the computer. The latency of the tracker is less than 1 ms.

An optional tracking system employed with the Head Mounted display was the optical OptiTrack FLEX 100 system that consists of six cameras. Each camera lens is surrounded by an infra-red light ring and small highly reflective spheres are used as markers. The cameras senses the infrared light reflected from the markers and by performing computer vision algorithms, is able to calculate the position and rotation of the markers. The cameras can capture 100 images per second and are connected to a USB 2.0 port. The tracking range depends on the size of the marker and placement of the cameras. The maximum range is 7 m. The latency this of tracker technology is 10 ms.

Finally, for a portable version of the interface, a magnetic Polhemus 3Space Fastrak tracker with three sensors was employed on a tiled display system. The update rate of this tracker system is 120 Hz with a latency of 4 ms. The system is connected to a computer using the RS-232 serial port or IEEE-488 parallel port at a maximum baud rate of 115,200. The tracking range is small, at around 76 cm, but it can be increased to 3 m with a reduction in accuracy.

3.7.5 Tracking Information Handling

Tracking systems are connected to computer system that collects the information provided by the sensors. A software driver is responsible for reading and organizing this information. If each tracker system utilizes its own driver, a common specification or software interface is necessary in order to define a unique development platform. Most tracking system manufacturers have relied on Virtual Reality Peripheral Network (VRPN), which is a flexible and extensible library created to handle information provided by different tracker technologies. VRPN [Taylor 2001] is an open source library that can be installed on most widely used operating systems. The library includes pre-installed drivers for many commercial tracker devices, including the four types listed in the previous section. Furthermore, if a driver is not available, the library offers easily extensible generic drivers that can provide the same function. VRPN offers a server background process (daemon) to accept connection requests and to distribute the tracking information to sev-

eral computers simultaneously through a network using the TCP/IP protocol. VRPN is implemented in C and the library offers a complete set of functions to integrate tracking information directly into C or C++ code. Applications using VRPN have the advantage of easy migration, flexibility, extensibility, and uniformity.

3.7.6 Virtual Reality Framework

The diversity of input and output devices when implementing virtual reality systems has contributed to the proliferation of drivers and libraries and that are difficult to integrate into unified applications. Moreover, after the creation of a successful application, migrating to a different hardware or software architecture can be a time consuming task. For these reasons, researchers have tried to develop frameworks that offer generic development foundations that minimizes the differences in hardware and operating systems, while providing a unique development platform to the developer. Examples of such frameworks are Cavelib [Mechdyne 2009], OpenTracker [Reitmayr 2005], DIVERSE [Kelso 2002], SVE [Kessler 2000] and VR Juggler [Bierbaum 2001].

3.7.7 VR Juggler Implementation

The development of the prototype magic lens interface was implemented in one such framework, VR Juggler. The selection was made on the basis of the flexibility and portability that VR Juggler offers. The platform is based on a set of independent software components that can be integrated according to the particular application requirements. VR Juggler is an object oriented framework programmed in C++. It is divided into eight modules.

VR Juggler kernel: Each application must be registered to and launched in this kernel in order to be executed. The kernel is an event-driven background process that organize the computation of the virtual reality applications. In this module the graphics subset is integrated as well. VR Juggler is able to interact with OpenGL [Shreiner 2004] or scene graphs such as SGI Performer, OpenSceneGraph [Martz 2007] and OpenSG [Reiners 2002] for rendering.

- Gadgeteer: This device management system provides the drivers, and the generic platform for managing all the information provided by the input devices.
- JCCL: Supports controlling the configuration files for defining the input and output devices.
- VPR: Provides platform-independent abstractions for low-level programming elements as threads, sockets, processes, clocks and so on.
- Sonix: Support audio. This is the module that integrates APIs such as OpenAL into VR Juggler.

- Tweek: CORBA, Java and JavaBeans provide the support necessary to build GUIs based on the distributed Model View Controller. This module allows the integration of Java and C++ applications in a distributed model under the CORBA specification.
- PyJuggler: A module for supporting Python programming.
- VRJ.NET: A module for integrating VR Juggler with .NET technologies.

Because we are developing a system for use in a heterogeneous environment, the prototype was tested on three different operating systems, Linux (Debian and Ubuntu distributions), Windows (XP and Vista) and SGI IRIX, all with different hardware architectures.

3.8 Design of the Client

Figure 3.6 shows the architecture for the client implemented on the mobile device. Like the server architecture, the client provides networking, and streaming layers. The actions of these layers are performed in inverse order to those of the server. While the server creates a video stream from the images captured from OpenGL and builds packets that are transmitted through the network, the client unpacks packets and decodes the video stream in order to extract the images that are displayed on screen. In contrast to the server, the client provides a 2D interface layer that presents the video on screen and captures interaction information. There is also a layer for transmitting the interaction information using a TCP/IP connection.

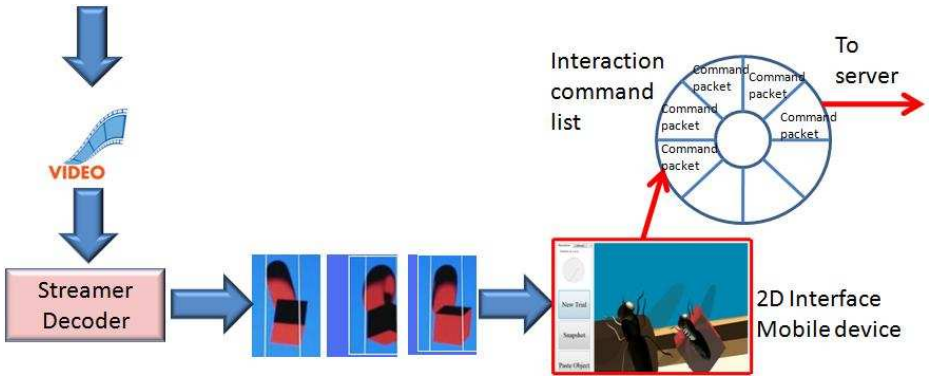

Figure 3.6: Block diagram of the client

3.8.1 Visual Interfaces for Handheld Devices

Handheld devices are provided with a preinstalled operating system. The controls and interaction events are determined by the set of libraries included in the operating system. Those libraries are programmed in a high level programming language such as C, or C++ while imposing a programming style. For the prototype, three different handheld devices were employed, a Windows Vista handheld computer and two PDAs, one with WindowsCE and the other with Linux Lineo. The libraries employed were all implemented in C or C++. The evolution of the interface implementation required the substitution of software technology in a specific layer according to the availability and performance of the handheld device employed. The result was a set of interchangeable layers with different performance characteristics.

3.8.2 Streamer on Handheld Device based on JPEG

The first client prototype was a WindowsCE based PDA. It was selected because of its size and processing characteristics. The libraries used for implementing the server were available for mobile technologies such as WindowsCE. Therefore, the Independent JPEG library and OpenGL/ES offered the development platform for implementing the decoder and the 2D interface layer respectively. Because the Independent JPEG library includes both compression/decompression algorithms, the library was simplified by eliminating the functions for compression in order to reduce memory footprint. Although the client only has to draw pixels, the prototype was implemented on OpenGL/ES to take advantage of the graphics driver provided by the PDA. The OpenGL/ES Utility Toolkit implementation for WindowsCE was used to capture the interaction events and provide the display window. OpenGL/ES is an OpenGL library specifically designed for handheld devices that does not offer all the functionalities present in OpenGL. For example, the commands ReadPixels and WritePixels for reading and writing directly from and to the frame buffer are not present in OpenGL/ES. The display of images is accomplished by creating textures from images and applying them on surfaces built with two triangle stripes that form a square.

Threads and networking support were implemented using the native WindowsCE API. The networking layer was similar to the server layer described above. The pipeline mechanism was implemented in the same way as the server, but in the reverse order. First, the client requests a TCP/IP connection to the server and simultaneously opens a UDP/IP port for receiving JPEG compressed images. A first thread, which reads packets from the UDP port, stores the images in a short list. We restrict the storing space to only two compressed images in the list because of memory limitations. A second thread takes images from the list and feeds them to the decompression algo-

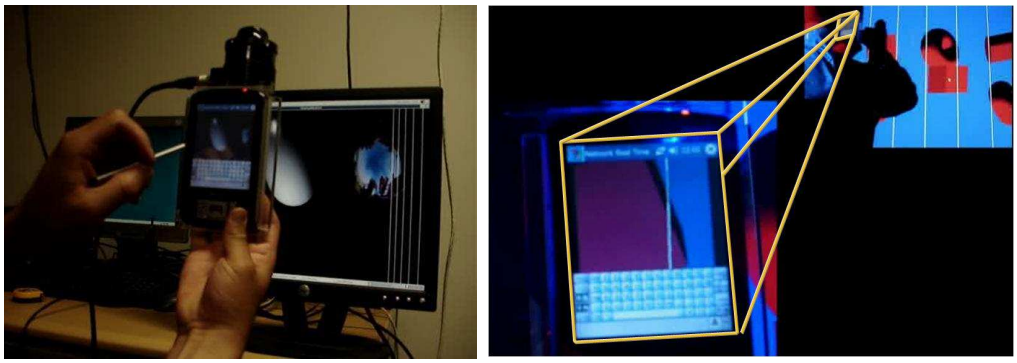

Figure 3.7: Two views of the first client prototype based on JPEG. On the left the PDA with the tracker sensor attached at the upper part. On the right a close up of the PDA's screen.

rithm that produces the image. Finally, the OpenGL/ES library transforms the image into a texture that is applied to the square described with two triangles. The creation of textures in OpenGL/ES is limited to sizes that are multiples of two. For that reason the system, only processes images with a resolution of 256*256 pixels. The OpenGL/ES interface detects interaction events that are inserted on an event list that is read by a third thread which transmits to the server using the TCP/IP connection previously established. Figure 3.7 shows the prototype.

A second prototype with the same PDA was implemented with the aim of reducing the delay and processing time. This prototype replaces the graphic support with a native graphics library that has direct access to the PDA's frame buffer. The name of the library is Game API (GAPI) and it is most commonly used for implementing video games on mobile architectures. GAPI makes it possible to switch the graphics operation of the PDA from a Windows environment to direct frame buffer memory access, thus giving the program full responsibility of the display creation. To access the GAPI library, the JPEG decompression layer requires a significant amount of modification in order to reduce the memory required for decompressing images. Compression and decompression algorithms in JPEG process one image line at a time. With direct access to frame buffer memory, the application has two advantages: First, the application is able to draw to screen immediately as a line is decompressed, thus saving time. Second, in terms of memory usage, it is only necessary to store one line of the image for decompression because after the line is decompressed, it is immediately moved into the frame buffer and the process is repeated for the next line. The Figure 3.10 shows the second prototype.

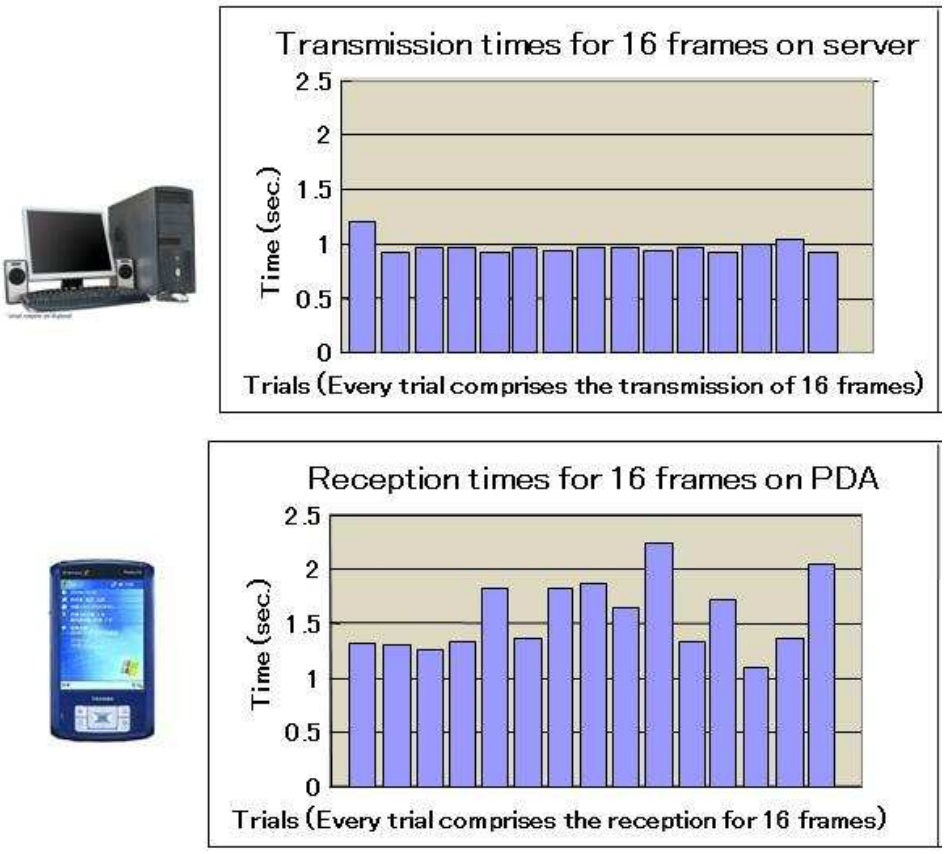

Figure 3.8: Graph of networking performance. The top shows the time for transmitting a group of 16 frames, while the bottom shows the time for receiving the same group of 16 frames on the PDA.

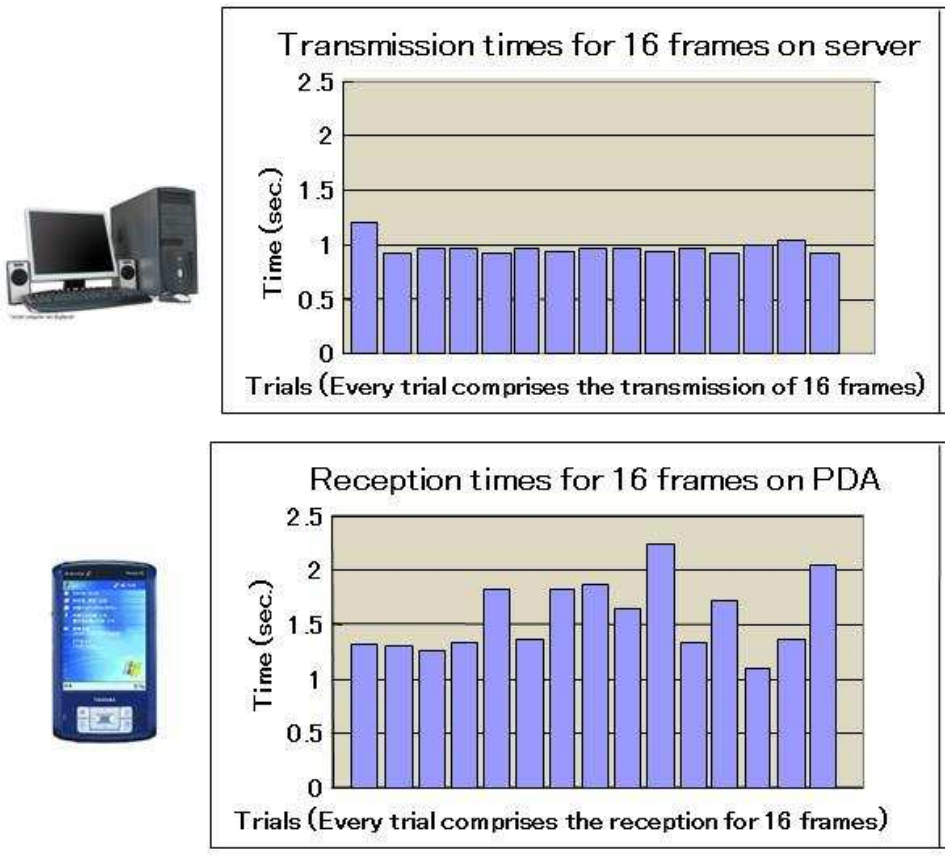

Figure 3.9: Graph of networking performance. The top shows the time for compressing a group of 16 frames, while the bottom shows the time for decompressing the same group of 16 frames at PDA.

Figure 3.10: Image of the second client prototype based on JPEG streaming.

3.8.3 Testing

For both prototypes, the server application was tested on Linux and Windows XP using a Pentium 4 3.2 GHz desktop computer with 1 GB memory. A Toshiba Genio e830 with an XScale PXA263 CPU with 128 MB memory was tested as the PDA device. IEEE802.11g was used for the wireless connection. For the tracking system, HiBall systems were used.

For the first prototype, the performance was poor. It displayed 256*240 24-bit color images at around six frames per second. The delay in the interface was about two seconds. This was not considered suitable for interaction.

The second prototype displayed 256*240 24-bit color images at around 12 frames per second with a delay of about 1 second. Figures 3.8 and 3.9 show graphs of the compression and decompression times and the transmission reception times between computer and handheld device. The decompression step was identified as the bottleneck of the application.

3.8.4 Streamer on Handheld Device based on MPEG

To alleviate the bottleneck on the client, the next prototype substituted the JPEG layer for MPEG streaming technology. Server and client were then re-implemented. For the client, the implementation was made according to [Lamberti 2007, Lamberti 2003]. At first, because of incompatibilities with WindowsCE based technology, the PDA was replaced by a Linux Lineo based PDA. The library FFmpeg was compiled and tuned for Linux. The FFmpeg

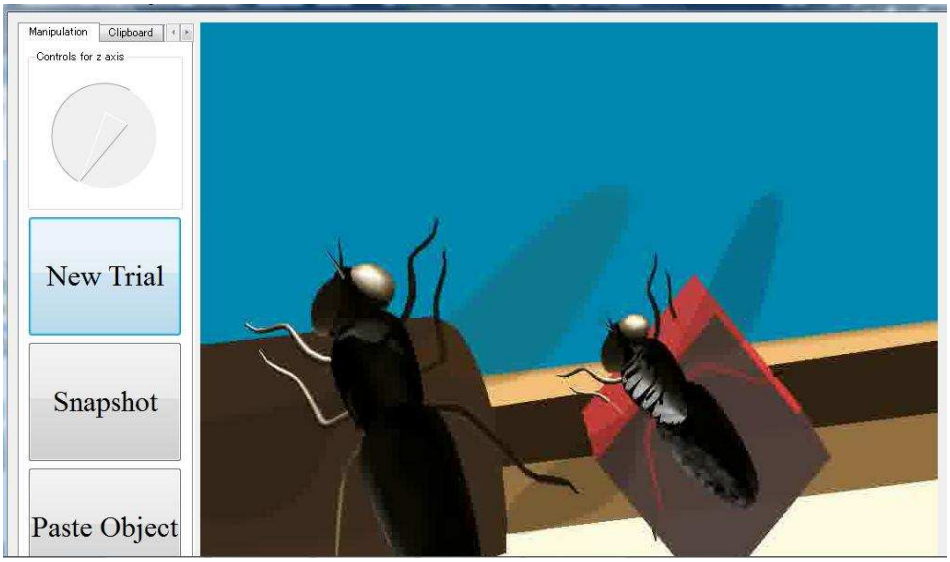

Figure 3.11: Image of the third client prototype based on MPEG streaming.

library itself includes networking support for video reception and synchronization. As images are decompressed and displayed on the PDA's screen, the time stamp number is verified. When an interaction event such as a screen touch is produced, the client application creates a network packet that includes the time stamp and the interaction information that is then transmitted to the server.

The interaction information is captured in the native PDA platform Qt [Qt 2009]. An additional advantage of implementing in Qt is the portability of the system. After a test with the Linux based PDA, it was found that the chip does not support intensive MPEG streaming in real-time so the application was migrated to a handheld PC without any modification in code. The handheld PC prototype increased the display area to 800*600 pixels. The Figure 3.11 presents a view of the prototype.

3.8.5 Interaction Information Channel

The interaction information is captured on the handheld device and sent to the server for interpretation, and finally to modify the environment accordingly. The architecture defines an interaction packet protocol that establishes a uniform mechanism of communication between the server and client. The client is responsible for generating packets following this protocol. The client 2D interface captures events such as pressing buttons or selection on the touch screen, generates the corresponding packet and then delivers it to the network layer for transmission. On the server side, when the packet arrives, the server relates the packet with a table of possible interaction events to determine the

action that must be performed. Finally, an event that is to be delivered to the internal VR Juggler kernel is generated in order to execute the operation required by the interaction information.

3.8.6 Testing

The server application has been tested on Linux and Windows using a Pentium 4 3.2GHz desktop computer with 1GB memory. A CAVE system using a Windows XP cluster with 17 nodes (Xeon 1.60GHz/1066MHz/Dual core, GPU ATI FireGL 7350/1GB Memory) has also been tested. A VAIO VGN-UX72 Intel Centrino Core 2 Solo, 1 GB memory computer with a display of 1024*600 running Windows Vista was tested for the client side. IEEE802.11g was used as the wireless connection. For the tracking system, FOB and Hi-Ball systems were used. With this equipment, the handheld device displayed 800*600 24-bit color images at 25 frames per second with a mean delay time of 127.8 ms. Figure 3.12 shows an example fluctuation of latency over about 20 minutes, measured as the time difference between frame transmission at the server and reception of the corresponding acknowledge packet.

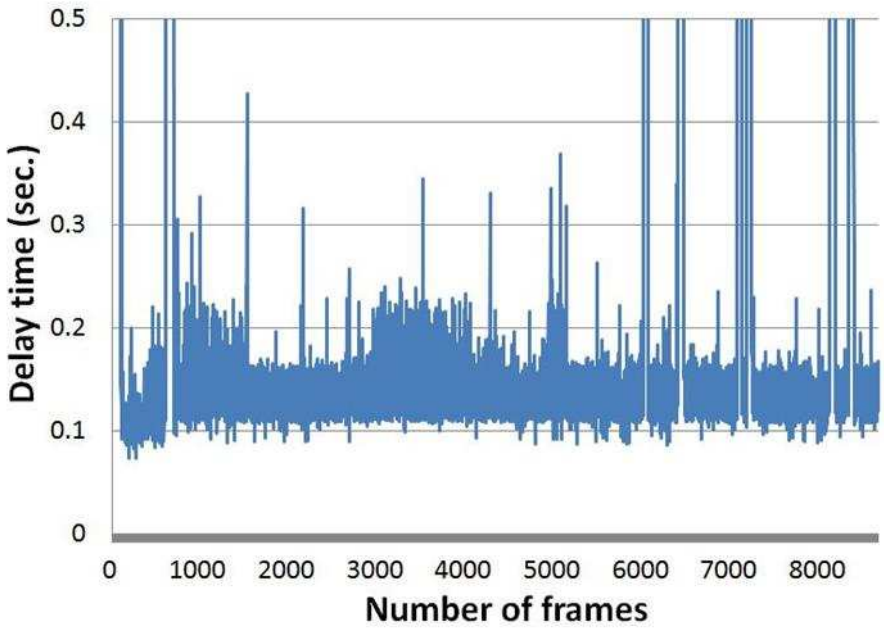

Figure 3.12: Performance of the MPEG-based implementation

3.9 Summary

A distributed system based on a client-server architecture was developed in order to implement the magic lens effect on a handheld device. A tracking sensor attached to the mobile device helps to determine its position relative to the virtual environment. The server generates all the images for the virtual environment, including the images corresponding to the portion of the area that the handheld device covers in the environment. The images corresponding to the handheld devices are collected for building a video stream that is transmitted over the network. Then, the mobile device displays the video stream and simultaneously captures interaction information provided by the user. In order to provide the illusion of immersion, real-time interaction is an important feature in virtual environments. For that reason, the architecture evolved from a JPEG based streaming technology to MPEG based streaming. Network delay led to this evolution. The architecture was implemented on a virtual reality framework that offers input-output device independence, scalability and configuration flexibility. The result is an implementation that is able to migrate from PC desktop computers to visualization cluster systems.

Chapter 4

Architecture for Designing 3D Interactions

4.1 Introduction

The characteristics of the input devices influence the design of the interaction techniques. The diversity of input devices for immersive virtual reality provides a rich set of possibilities for creating interactions. However, excessive diversity makes it more difficult to establish a set of generic interaction techniques. From the programmer's perspective, this diversity carries other problems. Each input device comes with a driver or a software module to read and provide input data to applications. This means a programmer must create code specific to every device. There are several research efforts offering libraries and frameworks that work to alleviate this problem, by offering a generic development platform: CaveLib [Mechdyne 2009], DIVERSE [Kelso 2002], SVE [Kessler 2000] and VR Juggler [Bierbaum 2001]. Applications programmed using these frameworks gain the advantage of easy migration onto different hardware platforms. However, 3D interface development does not take significant advantage of these software architectures because the interface programming occurs in an upper layer above input data handling. In other words, for an interface developer, it is not especially important to worry about whether the application will run on one hardware platform or another. Instead, it is more important to know what the characteristics of the input devices are. This includes, for example how many buttons they have, what kind of gestures they can identify, how many degrees of freedom they have, and so forth.

With a handheld device, a standardized set of 2D interfaces are introduced providing 2D interaction metaphors that can be extended into 3D. Moreover, as the handheld device is attached to a tracking sensor device, the device can also be used as a wand device capable of implementing 3D interactions. Numerous software layers are available to handle the available interaction techniques. Choosing the most suitable metaphor, according to the context the handheld device is applied in, provides more flexibility in the overall system. This chapter describes an architecture based on a set of generic modules that when combined, provides a full range of interaction techniques along with their management.

4.2 Implementation Goals

The architecture described in Chapter 3 accomplished the goal of offering an implementation of the magic lens metaphor on a handheld device. The interface was tested on different hardware and software configurations and showed similar performance on all of them. With the introduction of the handheld device, a full set of 2D interactions was modified to provide an extended version for 3D interactions. Since the tasks are implemented following the same mechanisms offered by the 2D version, one advantage of this approach is that the users can easily learn how to use the interface. Furthermore, in order to take full advantage of the handheld device potential, the system combines the magic lens metaphor with other 3D interactions techniques. For the correct combination of interactions, the system requires a software layer that handles the different metaphors, switching between them according to the context in which they are applied. This layer provides more flexibility and extensibility to the interface. The same layer can be used for implementing interfaces with different input devices during the interface evaluation process.

Interfaces implemented with this architecture take advantage of:

- Flexibility: Some interfaces extend the interaction of other interfaces. For example, HOMER uses ray-casting as a selection tool and later switches to a hand manipulation behavior like Go-Go, but with a different extension factor. Therefore, establishing the basic operations in the form of modules makes it possible to combine them freely. As a result, a complex interaction technique can be formed as a set of more basic interaction elements.
- Reusability: Basic building blocks can be reused, creating complex interaction by integrating basic interactions.
- Performance: Programming highly efficient building blocks would impact positively on the applications because all interactions techniques would rely on the same set of modules.

4.3 Architecture Conceptual Design

A previous study combining interaction techniques is Chasm [Wingrave 2008, Ray 2007], which describes an architecture for implementing several metaphors using different input devices. The architecture proposed in this study is a concept-oriented design, which presents interfaces as combinations of reusable function chunks (concepts) organized in tiers. Chasm has been used for implementing evaluation test beds that employ several interaction techniques in one application, thus offering greater flexibility. However, since the development of the interface proposed in this proposal focuses on a single

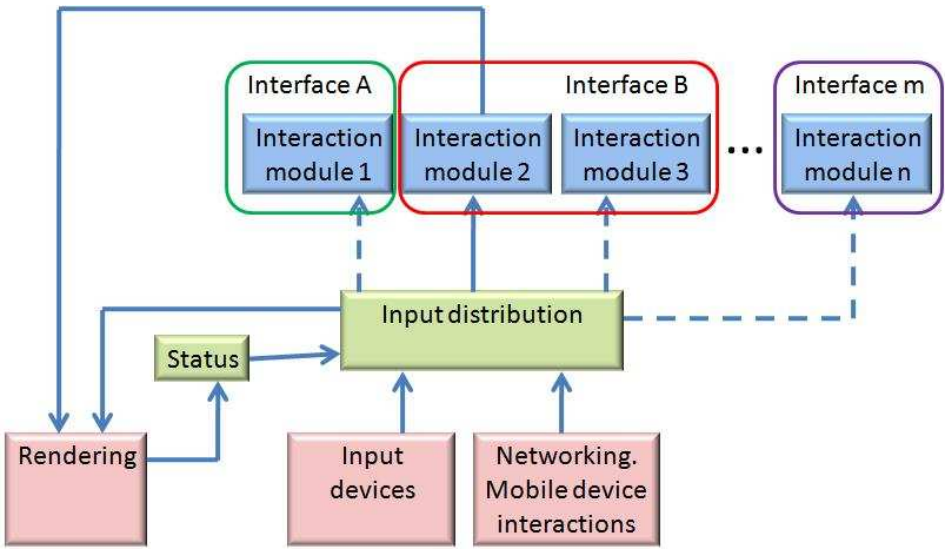

Figure 4.1: Diagram of the architecture.

input device (the handheld device), as opposed to the goal Chasm is pursuing, the use of Chasm was not considered in the development of our interface. Chasm represents a complex infrastructure focusing on describing interactions through a set of rules that are mapped onto input devices. Instead of using Chasm, a thinner, generic layer was built that provides direct access to input information and their efficient distribution to modules implementing the interactions. The modules worked as building blocks for creating interactions that are more complex.

Figure 4.1 shows a block diagram of the concept design. In this architecture, the input distribution module delivers the input information to the current active interaction. As illustrated in the figure by the solid arrow joining the input distribution module with interaction module 2 only one interaction can be active at a given time. The dot arrows show references to deactivated interaction modules. The input distribution module captures the input information and simultaneously checks for the current status. The status defines how the input information is interpreted, whether as control or as interaction information. Then, depending on the active interaction, the intermediate module delivers the information to the corresponding interaction module for processing. The interaction modules are able to call to the rendering module directly in order to provide visual support, such as a graphical representation of the interface, object intersection, etc. Since the rendering process employs the interaction information when creating the images, the rendering module is the best candidate to be the module responsible for updating the status information. The term input information refers to the information produced by the user as a result of its interaction. Input information can be generated

by the 2D interface or the tracking system. When the information is generated by the handheld device, actions such as selecting a button on the display screen are interpreted by the handheld device, which then builds a packet with the input information to be sent to the server. When the packet arrives at the server, the input distribution module delivers information captured at the handheld device to the corresponding interaction module. As such, 2D input information is processed by the handheld device. The layer is small, and easy to extend. Every time a new interaction is added, a new module is registered in the input distribution module. The new interaction module is then able to receive input information.

4.4 Interaction by Using the Handheld Device

The use of a handheld device provides additional input/output information to the virtual environment system, extending the possible forms of interaction. Most handheld devices are designed to be manipulated with both hands in an asymmetric interaction arrangement. The non-dominant hand grasps the device while the dominant hand manipulates the device controls and touch screen. A click on the touch screen can be performed with bare fingers or using a pen-like stylus. The interaction with a handheld device resembles the action of writing with a pen on a tablet while providing support with the other hand. This is known as the pen-tablet technique.

4.4.1 Pen-Tablet Technique

The pen-tablet technique offers several advantages to the design of interfaces for immersive virtual environments.

- Reduces the DOF of interaction from 3D to 2D.
- Offers passive haptic feedback.
- Manipulation task can be performed with more precision because the interaction is performed with a static image on a 2D limited surface.

The pen-tablet technique sets up constrains to 3D interaction, limiting the interaction space into a more comfortable and easy to handle 2D environment captured on the tablet display area. Since the user manipulates a real object (the handheld device) as the interaction tool, the user receives passive haptic feedback that enhance the immersion experience.

4.4.2 Selection using Handheld Device

The interface proposes two options for selecting an object:

- Direct pointing: When the object is visible on screen, the user is able to select it directly using the touch screen. This method is effective for selecting static objects that can be identified without ambiguity.
- Using a snapshot: It is possible to store a snapshot and then to select an object from it. This process is similar to taking a picture of the scene and then performing the selection on that picture. This selection method offers more stability and precision.

4.4.3 Manipulation with the Handheld Device

The pen-tablet technique is used for selecting objects. After selection, the role of the handheld device is modified and it becomes in a 3D manipulator device. During object manipulation, the handheld device becomes an actual proxy for the virtual selected object. Wherever the handheld device is moved, the virtual object will move to the corresponding position in the virtual environment. This easy-to-understand metaphor has the same limitations that real object manipulation has, so several extended approaches can be used:

- Ray casting for interaction. During object manipulation, the distance between the handheld device position and the selected object is preserved. This manipulation method offers a simple manipulation technique that is useful when the relationship between user and object does not change significantly.
- Using a fishing-reel [Bowman 1997a] approach. An additional 2D slide control included on the handheld device display helps in manipulating the distance between the handheld device and the virtual object. The drawback of this approach is that the manipulation action is separated into two different controls, decreasing the performance of the interface.
- Using HOMER-like manipulation. The handheld device is manipulated as a stylus on a HOMER interface. When an object is selected with HOMER, two distances are calculated, the distance between the user and handheld device and the distance between the mobile device and virtual object. Using both distances, the interface calculates a linear extension factor. Then, when the object is manipulated, the interface applies the calculated linear factor to the object in order to determine the final movement. This method presents the same problem as HOMER: That is, clutching. If after a near object is selected, the user wants to move it to a distant location, the factor is so small that it requires several steps to accomplish the task.
- Using a Go-Go manipulation style. After selection, the object is attached to the handheld device. If the handheld device is inside a threshold zone, the object will have the same position the handheld

device does. However, when the handheld device is outside the threshold, the object position is multiplied by a logarithm factor that extends the separation between handheld device and the object.

4.4.4 Clipboard

The interface stores all of the snapshots taken in a clipboard tool for future reference. Previous snapshots can be used to create a history view of the environment, or to refer to an old object arrangement. Advanced functions include the undo operation, which restores the environment to the state shown in the snapshot.

4.4.5 Virtual Cameras

The magic lens interface can be thought of as a virtual camera. The mobile device screen shows the user images that are generated from the handheld device view point just as a camera does. If the user finds a position in the 3D environment that offers a suitable point of view for a specific task, that position can be stored, and a virtual camera appears in the environment indicating its position. The screen interface on the handheld device allows the user to switch views to any of the virtual cameras, transfer such views to the real handheld device, and perform interaction tasks from those viewpoints.

4.4.6 Virtual Map

To support navigation, a view of the entire virtual environment is represented as a map. The user is able to change position by selecting a new location on the map. This interface will be extended to offering object manipulation as is possible in world in miniature interface.

4.4.7 Text and Annotations

Annotations, controls, and text can be integrated on the handheld device as shown in previous studies [Poupyrev 1998a].

4.5 Implementation of a Prototype

To verify the feasibility of the architecture, a prototype was developed. The architecture is programmed on top of the VR Juggler framework. Figure 4.2 shows the block diagram of the interaction architecture.

The next subsection gives an explanation of the blocks.

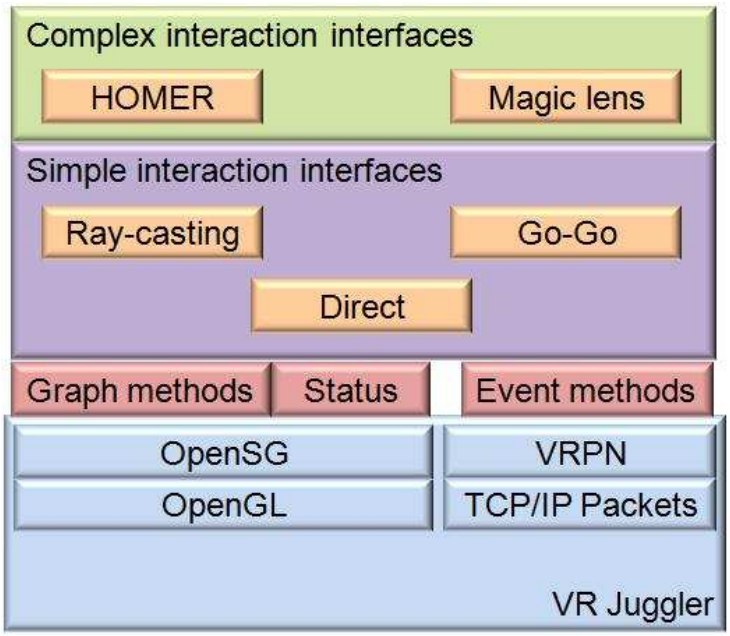

Figure 4.2: Block diagram of Interaction Architecture

4.5.1 Event Handling

Virtual reality frameworks are based on an event-driven programming paradigm. On VR Juggler, a kernel software module is responsible for controlling the loading and execution of applications. When an application is executed, the kernel provides a control loop that continuously calls software methods in a predefined sequence. In order to create an application, the developer must provide source code for the software methods that are called by the control loop. The number of software methods to be implemented depends on the graphics library employed. There are different software methods for OpenGL [Shreiner 2004], OpenSceneGraph [Martz 2007] and OpenSG [Reiners 2002]. Regardless of the graphics library employed, every application must implement three fundamental methods: preframe, draw and postframe. These correspond, respectively, to the software method called before, during, and after frame rendering.

In order to render the virtual environment according to the user's point of view, VR Juggler requires the user position and all the interaction information provided by input devices. For that reason, all input information is captured by the preframe method, before the rendering is performed.

Although the prototype uses a VRPN client module for receiving information from the tracking system, it is not implemented using the VRPN library. Instead, the prototype relies on VR Juggler to acquire input information. VR Juggler implements proxy [Gamma 1995] functions to create a uniform

and flexible mechanism for communicating with different device drivers. The proxies are classified into seven categories depending on the type of information the devices provides: analog, digital, command, glove, keyboard-mouse, position, and string. Classification is made depending on the type of captured information. For example a digital proxy registers an on/off event, position registers a vector containing a 3D position and a matrix containing rotation. The programmer integrates the proxy function into the program and the VR Juggler framework handles the connection with the corresponding input device driver. VR Juggler provides drivers for several input devices and libraries, one of which is VRPN. A VR Juggler layer called Input Manager attaches the proxy with its corresponding device driver at run time. The advantage of this separation is that proxies can be configured for connection with different drivers. The proxy configuration is performed with a flexible XML file format called JCCL.

The prototype extends the preframe method in order to capture the information provided by the proxy. Then, the prototype checks for the interaction status in order to propagate the input information to the corresponding interaction module. The propagation mechanism is implemented by using dynamic pointers. All of the interaction modules are registered to the application, which includes a stored pointer table. The application allows only one active interaction module at a time. When the interaction metaphor is changed, the interaction module is located in the pointer table and the application then switches to it. The interaction status determines the method called in the interaction module.

The interaction status determines the interpretation of the input events. For example, if a user is using a stylus with one button, the action performed by pushing the button changes according to the task the user is performing. If the user is selecting an object, when the button is pushed, the object in the current position is selected, but if the user is starting a trial, pushing the button starts a new trial. The possible interpretations of an input event are predermined, and the interaction modules provide methods for each possible interpretation. Then, when rendering is performed and the interaction status is checked, the correct method is called. In the prototype, the classification has two groups: control and interaction. Control refers to the events that provide commands or switch the interaction mode. Interaction refers to events that directly help the interaction.

4.5.2 Graphic Representation

All rendering activity relies on OpenGL. However, describing complex geometries in a structured representation with only raw OpenGL is cumbersome. A more suitable approach is to employ a scenegraph. Scenegraphs offer a hierarchical representations of geometry. They separate modeling from rendering

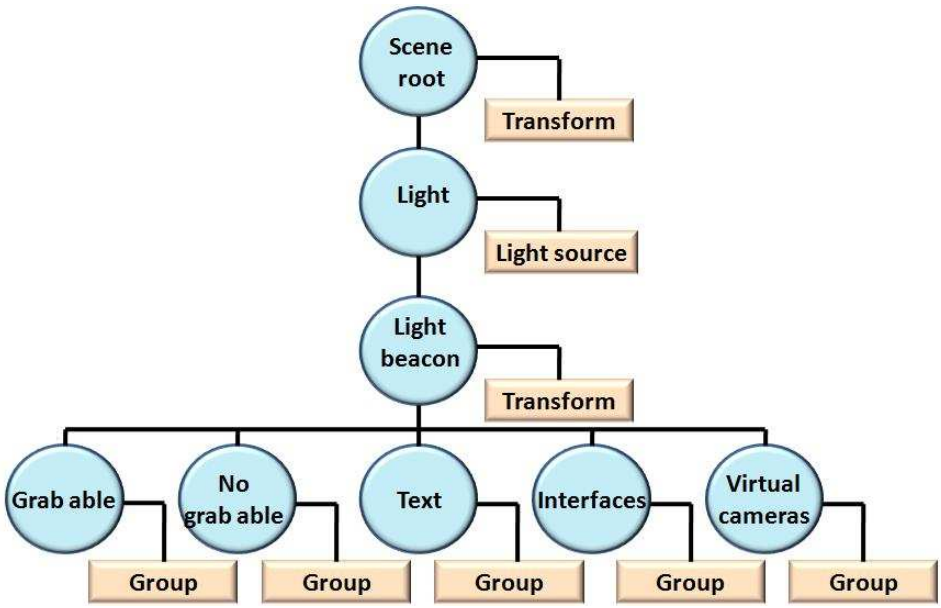

Figure 4.3: Scenegraph of the prototype

by using intermediate file representations. For scenegraph implementation, efficient rendering algorithms necessary for good performance. Therefore, for the prototype, virtual environment geometry was created in a software modeler and was imported into a scenegraph using OpenSG [Reiners 2002]. Inside the VR Juggler kernel loop, the draw method calls OpenSG rendering methods in order to perform the render. Figure 4.3 shows the diagram of the scene graph. The scene graph classifies the objects in the virtual environment into two groups, non-grabbable and grabbable objects. The former cannot be selected, These include, for example walls, floor, ceiling etc. The latter are objects that can be manipulated. OpenSG represents every geometry in the scene as an individual node in a parent-child relationship.

When a hierarchy is used, operations over geometric elements become more efficient. For example, for intersection testing, the prototype ignores the branch corresponding to non grabbable objects in order to traverse and test only the group of objects that are grabbable.

Some 3D interfaces require a geometric representation, for example ray casting requires a long line representing a ray. Go-Go requires a hand. These representations are created in the scenegraph and are included in the non-grabbable branch. Since the interface representations correspond to individual graph nodes, it is easy to modify or change their properties and geometries.

The graphical representations of the interfaces, text, virtual cameras, and other sources are stored as independent branches since they are considered to be support elements with different characteristics not virtual objects.

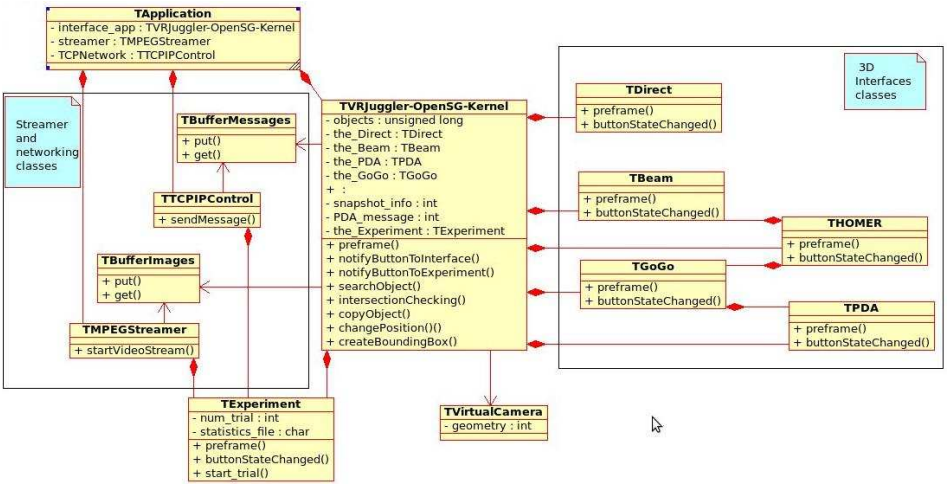

Figure 4.4: UML diagram of interaction architecture.

The rendering module offers a set of methods to directly manipulate the scenegraph. Such basic operations include add, delete nodes, modify color attributes, intersection testing, and so on. The interaction modules calls these methods in order to manipulate geometry.

4.5.3 Interaction Modules

The codes for implementing the interactions are programmed as individual classes grouped into modules. Each interaction must be registered in the VR Juggler preframe method in order to receive events and to be able to call the graphics methods offered by the graphics layer. After registration, a 3D interface can be built by using the interaction modules, and the graphics module automatically propagates the input device data into the interaction module for processing. In this way, each interaction module is responsible for the interpretation of input data.

This class independence among the 3D interfaces provides a great deal of flexibility to the system because complex interfaces can be created as a combination of basic interactions.

4.5.4 Implemented Interfaces

The prototype implements four interfaces with the architecture. They are direct, Ray-casting, Go-Go, and magic lens.

Direct interface requires one unique interaction module that performs a search for an object intersecting the user's hand when a button in the stylus is pushed.

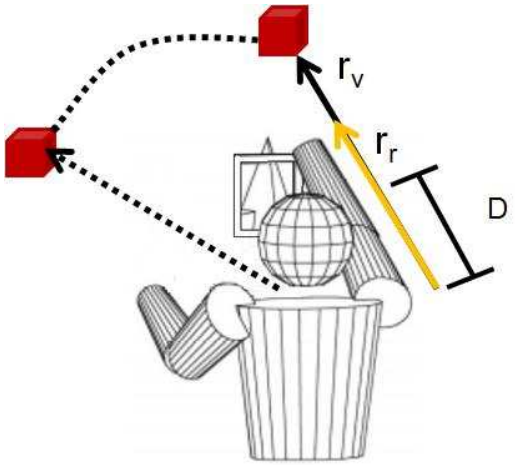

Figure 4.5: The manipulation of objects is performed with a Go-Go interaction technique.

Ray-casting interface is implemented with one interaction module. This module relies on the methods provided by the scenegraph to create a line. Then, when a button is pressed, a possible object intersection is searched for in the scenegraph.

The Go-Go interface consists of an interaction module. The module applies a scale factor when the user's hand is outside the predefined threshold, at which time the user's hand movement is elongated.

The HOMER interface is implemented as a combination of two interaction modules. Object selection employs the ray-casting interaction module at which time later the Go-Go interaction module is applied with a different elongation factor.

Finally, the magic lens interface is implemented as a combination of a specific interaction module that handles object selection. Figure 4.6 (right) shows the selection of an object. Figure 4.6 (left) shows a snapshot taken from the environment. The user can select any object contained in the snapshot. After object selection, a Go-Go interaction module is used to manipulate the object. From the Figure 4.5 the mathematical expression that defines the final position of the virtual object is:

$$r_v = F(r_r) = \begin{cases} r_r & \text{if } r_r \leq D, \\ r_r + \alpha(r_r - D)^2 & \text{otherwise.} \end{cases} \tag{4.1}$$

When used with a carefully selected extension factor, the Go-Go method provides good performance.

Clutching is necessary when the user wants to move the object to a remote

Figure 4.6: On the left, is snapshot taken with our interface. On the right is an object selected directly from the screen by use of a stylus.

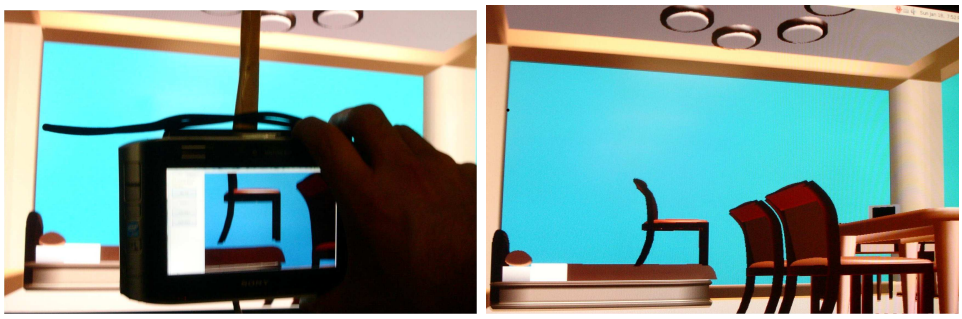

Figure 4.7: Manipulating an object. On the left, a previously selected chair is shown. Tacking advantage of the tracker attached to the handheld device, the chair position changes according to the Go-Go interface interaction technique. When the user decides the final position, the chair's position is fixed after with a push of the button. On the right, the scene after the user has copied the chair using the stored snapshot is shown.

location. However, as the object moves further from the user, its size decreases proportionally. As a result, it can become difficult to see.

Figure 4.7 (left) shows the manipulation of a previously selected object. Figure 4.7 (right) shows how the object position is determined after being manipulated with our interface. The object rotation is controlled using the Virtual Sphere [Chen 1988] approach. Rotation is produced by sliding a finger or styles across the handheld device screen. An extra support, a slide control was added for rotating objects over the Z-axis. This control was added to avoid forcing the user to perform uncomfortable movements, such as attempting a 180 angle rotation by grabbing the handheld device with both hands.

The implementation of the clipboard function is divided into two components, which are distributed to the server and client respectively. On the server, the system stores a clipboard list that includes the information required for ren-

dering the image. This information was previously stored when the handheld device images were generated. On the client, the clipboard is implemented as a graphic interface showing the snapshots taken with the interface. The handheld device stores every snapshot taken, together with its DTS timestamp, which is provided by the streamer layer. When a snapshot is requested, the handheld device sends the DTS timestamp. The server then searches for the corresponding DTS rendering information in order to re-render the snapshot.

A very important consideration for using the magic lens interface in an immersive virtual environment is the projection technology. The interaction with a handheld device is possible in a CAVE system and with a head-mounted display that offers a see-through the lens display. A low latency and correct tracker calibration are necessary for matching the images presented on the virtual environment and the handheld device. Real world occlusive head-mounted displays cannot be used with this interface.

4.6 Summary

The development of 3D interfaces lacks development frameworks and libraries to implement interactions. Generally, virtual reality developers program interactions for every application from scratch. Additionally, the implementation of interfaces requires sufficient flexibility to mix several components. This chapter describes an architecture for building basic, generic interaction techniques, on top of which, developers can implement complex interfaces. A prototype was implemented on top of the VR Juggler framework. Direct manipulation, ray-casting, Go-Go, and HOMER were programmed with this architecture. With the support of the described architecture, the magic lens interface implements the manipulation of objects.

Since handheld devices offer multiple functions such as a touch screen and programmable visual 2D interfaces, the module that oversees a magic lens provides a wide range of functions for supporting interactions. The interface applies the pen-tablet technique in combination with props metaphors. In order to provide physical feedback, stimulus is important for enhancing interaction. Performing the selection of objects from 2D snapshots provides more additional precision and control for small and moving objects. Handheld devices are small computer systems equipped with memory, processors, graphic displays, and libraries that offer applications for implementing visual interfaces. All these elements conform to a flexible development platform, which, in combination with virtual reality toolkits, make feasible the creation of multiple 3D interaction techniques.

Chapter 5

Interface Evaluation

5.1 Introduction

Interface evaluation is the process of measuring its usability. The term usability is used in broad sense, considering all the aspects which take part when a person uses an interface. The interaction with 3D interfaces deals with a bigger and more complex set of variables than traditional 2D interfaces, as a result, there is not a standard method of evaluation. Evaluation methods are designed considering multiple aspects such as, the type of activity, the virtual environment, the conditions where the interface will be applied, the input and output devices used, the user's experience with virtual reality systems and so on. Equally, it is important to reduce and, if it is possible, to eliminate factors that could influence negatively the evaluation. Factors such as fatigue, symptoms of sickness, failures on tracking system, system delays circumstances which breaks the sensation of immersion as crossing by accident in front of the subject and so on.

Important steps on evaluation are the establishment of the metrics in order to define what properties will be measured, the design of a controlled environment for experimentation and the analysis of results. Since the use of an interface involves psychological factors as well, preferences, opinions, suggestions, subjective information are compiled by means of questionnaires.

This chapter describes the method employed for the evaluation of the interface, and its application on the prototypes. The evaluation provided valuable feedback information which helped to identify drawbacks on the design and implementation.

5.2 Methodology

The evaluation of a 3D interface faces more complexities than 2D interfaces, The number of variables considered for the evaluation is bigger (3D space, 6 DOF, virtual environment space), technologies employed to implement virtual reality are more prone to influence the results (input and output devices, delay, wires), human factors are more noticeable (fatigue, symptoms of sickness).

Several evaluation methods have been proposed for 3D interfaces, in [Bowman 2002] a survey explores them. The survey proposes a classification

based on the identification of three characteristics which help to distinguish the type of evaluation.

- The involvement of users: Separates the methods requiring user for its evaluation and methods that do not.
- the context of evaluation: Identifies the methods that can be applied in generic contexts and the methods that only can be applied in specific application contexts.
- the type of results produced: Identifies the evaluation methods that produce qualitative or quantitative results.

As a demonstration of the feasibility of the classification, the same study [Bowman 2002] compares two representative evaluation methods, the testbed evaluation approach described in [Bowman 1999b], against the sequential evaluation approach described in [Gabbard 1999]. Analyzing the evaluation methods through the three above mentioned characteristics, the result is that both methods have similarities and differences, and they can be combined in order to complement each other.

The evaluation of the interface follows the mixed approach. Sequential evaluation was used on the development of the prototype, since it does not require of a user to be tested, it is applied in generic context (no particular environment) and produce quantitative results (networking delay). In the other hand, testbed evaluation is applied to evaluate the manipulation with the interface. It requires of subjects, controlled 3D environments, and qualitative and quantitative information is compiled and analyzed. The sequential evaluation method was used for calculate delays time on the three prototypes describe in the chapter 3.

5.3 Testbed Evaluation Approach

The testbed evaluation approach is described in [Bowman 1999b]. The diagram 5.1 shows the components and its relationship.

Initial evaluation comprises the analysis of the interaction to be implemented and the search of previous developed methods which accomplish the same or similar interaction.

The taxonomy describes the interactions as tasks build as combination of indivisible subtasks.

The outside factors correspond to the considerations to be taken to avoid the equipment and the real environment influence negatively on the performance of the interface.

The performance metrics define the information that the system will register for comparing quantitatively the interfaces.

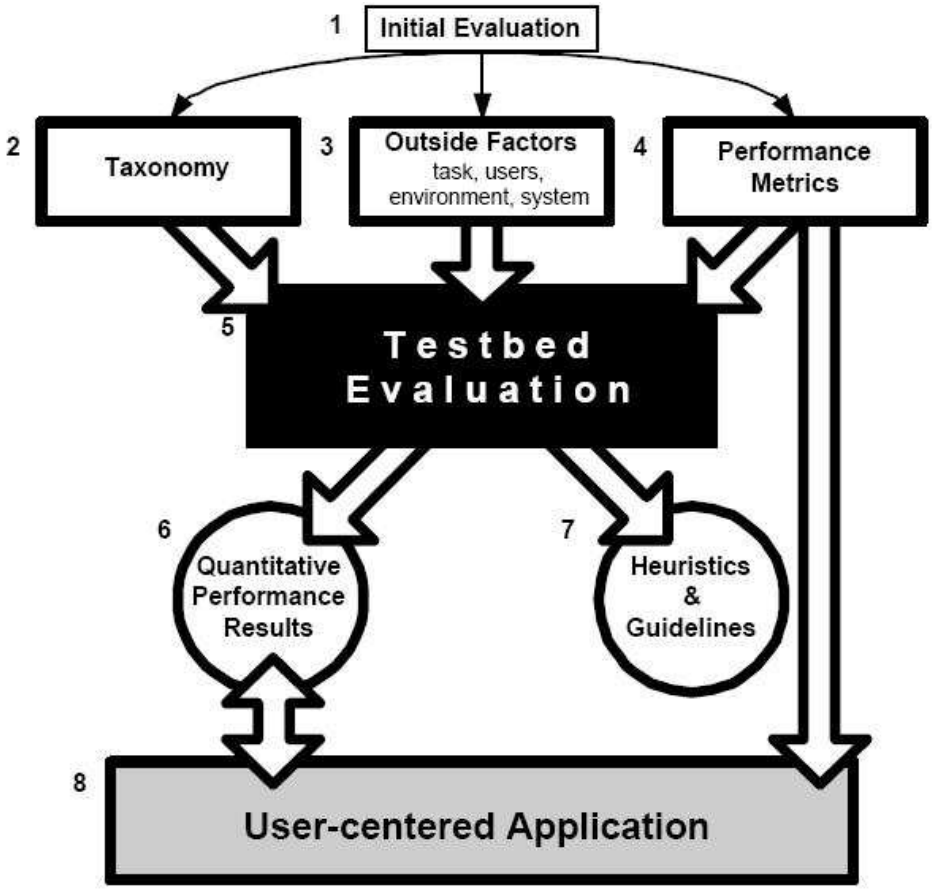

Figure 5.1: Diagram of 3D user interface testbed evaluation approach. Originally published on [Bowman 1999b]

Testbed evaluation is the design of a generic and general experimental environment, the participation of subjects on the user study, the compilation of observations and interaction information.

Results of testbed evaluation are produced after the analysis of the data captured on testbed evaluation. The results represent a quantitative measure of the interface's performance under the conditions described by the outside factors. The result information is structured with the aim of offering heuristics and guidelines for future improvements.

In order to apply interface evaluations in early stages of the development, and for reducing the complexity of the analysis as well, the testbed evaluation was divided in two parts, selection and manipulation. Since the testbed evaluation was applied for every prototype implementation the next sections presents the results on its chronological order.

5.4 Evaluation for Selection on the JPEG-based Streaming Interface

Selection was evaluated with both implementations of the interface, with JPEG and MPEG streaming technology. Although the first implementation of the interface resulted in a delay time no suitable for implementing interaction, its evaluation was performed with the intention of verifying its feasibility and, at the same time, to establish guidelines for the next development stage.

5.4.1 The Outsider Factors

The independent variables is conformed by the interfaces to be compared in the evaluation: Direct, Ray-Casting and Magic Lens. The selection of independent variables is according to the taxonomy developed during the implementation stage. For the selection of virtual objects previous studies [Bowman 1997a] have shown that ray-casting is an interface that is easy to understand and effective. Direct is an orthodox direct manipulation interface (virtual hand) which represents the case that resemble reality selection, it is offers an interesting parameter since is like the selection in our real world.

Since the conditions of the virtual environment influences the performance of the interfaces, the testbed is designed with the aim of controlling the factors that take part on the evaluation process. The factors that can influence the comparison can be classified into four characteristics according to [Bowman 1999b]

- Task: A simple targeting task was designed in which a subject had to select a green virtual cube among a cluster of red cubes aligned in a grid. The virtual environment is defined in a virtual room of 3 x 3 x 2.5 m. Each cube is 10cm on each side. The green target cube is chosen at random for each trial.
- Environment characteristics: The lowest row of the grid is located 40cm above the floor. Two types of layouts, Sparse and Dense are used. Sparse contains 6 x 4 x 3 cubes with a separation of 30cm between cubes, while Dense contains 12 x 7 x 6 cubes with a separation of 10cm. The background of the environment was blue in order to produce a noticeable contrast among the objects. Figure 5.2 shows the environment. The system provides visual and audio feedback information in every event. For example for Direct and Ray-casting, when the subject collocates the stylus in a position where the interface is able to select a cube, the cube's edges change color to bright yellow indicating that the cube is able to be selected. When subject success on selecting the target, the system produces a sound message "that's good" and

an error on selection produces the corresponding audio message "that's bad".

- User characteristics: The evaluation considers subjects with previous experience with virtual environments systems.
- System characteristics: A see-through head mounted display Olympus was used with a HiBall 3000 optical tracking system presenting a delay of 1 ms. The server application was tested on Linux Debian using a Pentium 4 3.2 GHz desktop computer with 1 GB memory, graphic card NVIDIA GeForce 6800. A Toshiba Genio e830 with XScale PXA263 CPU with 128 MB memory was tested as the PDA device. IEEE802.11g was used for the wireless network. The PDA displayed 256x240 24-bit color images at around 12 frames per second. The weight of the PDA was 198 g. and 300 g. for the tracker sensor, for a total of 498 g. For implementing the Direct and Ray-casting interfaces a stylus with a button device was used. The frame rate produced by the graphic card for building the environment was 40 fps.

5.4.2 Performance Measures

Task completion time from the appearance of the target to its selection and number of errors per selections are recorded for each trial.

The subject was asked to fill in a questionnaire in order to collect subjective data. The questions were formulated to be answered given a scale marked from 1 to 7 asking for the impressions that interfaces produces on the subject:

- It was easy to select the object
- I understand easily how to use the interface
- Selecting objects with this interface is natural

And two questions for knowing the impression about the virtual environment:

- Virtual world seems real.
- It is easy to identify the target.

The final question was to ranking all the interfaces in subject preference order.

5.4.3 Testbed Evaluation

For this first evaluation each subject performed three trials using all of the three selection techniques in both Dense and Sparse layouts (in total, six different conditions). The order of presentation to the subject was randomized. In order to start a trial the application asks to the subject to push a button, this pause gives the chance of taking a rest between trials. The application

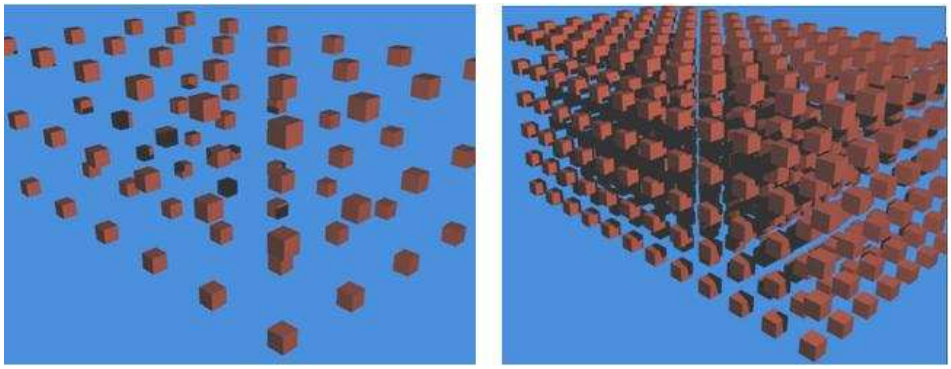

Figure 5.2: On the left, the Sparse environment. On the right, a Dense environment.

defines a limit time for the trial, if within 2 minutes the subject is not able to select the target cube the application considers the trial as an error and finishes it.

5.4.4 Hypotheses

For the selection of objects in Sparse environment, the hypotheses was that Magic-lens would has the same performance as Ray-casting. For Dense environment, the capability of selecting objects by using snapshots would represent an advantage over Ray-casting.

5.4.5 Results

9 subjects aged in their twenties and with previous experience in virtual reality tried the evaluation. Figure 5.3 shows average task completion times in Sparse and Dense layouts. Due to the small number of trials (three per subject) and very large standard deviations, statistical analysis gives no meaningful results. On the other hand, average task completion times for Direct and Ray-casting are relatively stable.

In the Dense layout, a higher performance was expected on the Magic Lens interface compared to Ray-casting, since occlusions would make Ray-casting based selection more difficult, while only a fraction of the target needs to appear in the Magic Lens two-step selection. The results do not support this hypothesis. There are not meaningful difference in number of errors either.

5.4.6 Discussion

Observation suggests that the Magic Lens interface was most difficult to use at first, but soon it became very easy to use. This can be confirmed by Figure

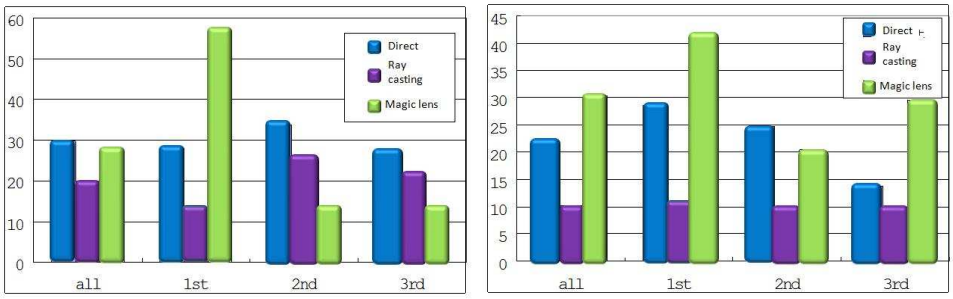

Figure 5.3: Average selection time (in seconds). On the left, the Sparse environment. On the right, a Dense environment.

5.3. In the Sparse layout, average task completion time of the first trial is 56.1 sec. but is drastically reduced to 12.4 and 13.1 sec. in the second and third trials.

5.5 Evaluation for Selection on the MPEG-based Streaming Interface

With a new improvement on networking layer, the new prototype based on MPEG streaming was evaluated applying the same testbed methodology. The user study focus on testing the conditions were other interfaces shows limitations.

5.5.1 The Outsider Factors

The user study considered the same set of interfaces used by previous study as independent variables: Direct, Ray-casting and Magic Lens. The factors that can influence the comparison in the evaluation are:

- Task: the evaluation considers a target selection task. In an empty virtual room of 3m by 3m by 2m (height), each subject is asked to select a target red cube per trial. The target cubes can either be:
  - static: the target cube remains in the same location
  - simple motion: the target cube moves in one of two possible patterns lineal or sinusoidal
  - pseudorandom motion: the target cube moves in a trajectory described in base of perlin noise function [Perlin 1985].

  The dimensions of the cubes are either 4.5cm (large) or 2.25cm (small)
- Environment characteristics: The model of the virtual environment was modeled with the aim of offering a trade-off between visual quality and

fast rendering. Over the room's walls, textures were applied in order to give the impression of black stone bricks. The floor was textured with a grass-like pattern, The final result is that the room resembles a backyard. The red cubes targets contrast noticeably with the walls. With Ray-casting and Magic Lens interfaces the subject was asked to stay standing up at the same location on the floor during the trial. (1.5m in x axis and 2.7m in z axis, Figure 5.5) shows the configuration. In the other hand, subjects could move freely when they were using Direct interface. Two different distances were considered to present the target. Tacking the trial starting subject's position as point of reference, in Far distance the target appears at 2.25 m and in Near the distance was 1 m. The system provides visual and audio feedback information in every event, in similar form that previous study evaluating the JPEG-based interface.

- User characteristics: The evaluation considers subjects with previous experience with virtual environments systems.
- System characteristics: The user study was performed with the server and virtual reality equipment previously mentioned on JPEG-based interface. The only modification was the handheld device employed. An ultra mobile personal computer (UMPC) VAIO VGN-UX72 (Intel Centrino Core 2 Solo, 1 GB memory, 1024x600 display on Windows Vista) was used for implementing the magic lens interface. The handheld device displays 800 x 600 24-bits color images at 25 frames per second with a mean delay time of 127.8 msec. The weight of the PC was 532 g. and for the tracker sensor 300 g. for a total of 832 g. The frame rate produced by the graphic card for building the environment was 40 fps.

5.5.2 Performance Measures

The user study registers two parameters. The task completion time measured as the time elapsed from the appearance of the target until its successful selection, and number of errors represented by the number of failed selection targets.

At the end of the study the subject was asked to fill in a questionnaire in order to collect subjective data. The questions were formulated to be answered given a scale marked from 1 to 7 asking for the impressions that interfaces produces on the subject:

- I understand easily how the interface works
- It was easy to select the object
- I enjoyed using this interface

And two questions for knowing the impression about the virtual environment:

	Direct	Ray-casting	Magic lens
Static	Large Small	Large Small	Large Small
Simple motion	Large Small	Large Small	Large Small
Random motion	Large Small	Large Small	Large Small

Figure 5.4: Trials performed by the user for this user study. Rows represents the object movement. Columns represents the interfaces to be used. Subjects are asked to perform all the possible combinations with both large and small objects.

- Virtual wold seems real.
- It is easy to identify the target.

The final question was to ranking all the interfaces in subject preference order.

5.5.3 Testbed Evaluation

In each of 9 interface-motion combinations, the large target cube was used at a distance of 1 m. from the subject in the first 10 trials, and the small target cube at 2.25 m. from the subject in the last 10 trials. Figure 5.4 summarizes the trials performed by subjects.

If the subject could not select the cube in 90 seconds, the trial is terminated and registered as a failure trial.

Previous to the user study starting the subject is able to learn and adapt to the interface by practicing with three trials without time limit.

Before each trial starts the application displaying a message asking for clicking a button on the input device to proceed to the next trial. Then user are able to take a rest between trials.

5.5.4 Hypotheses

The hypothesis formulated for this user study was that selecting an object using the handheld device based Magic-lens interface will be faster and more accurate than using Direct and Ray-casting interfaces in difficult situations (e.g. the object is small, distant, or moving).

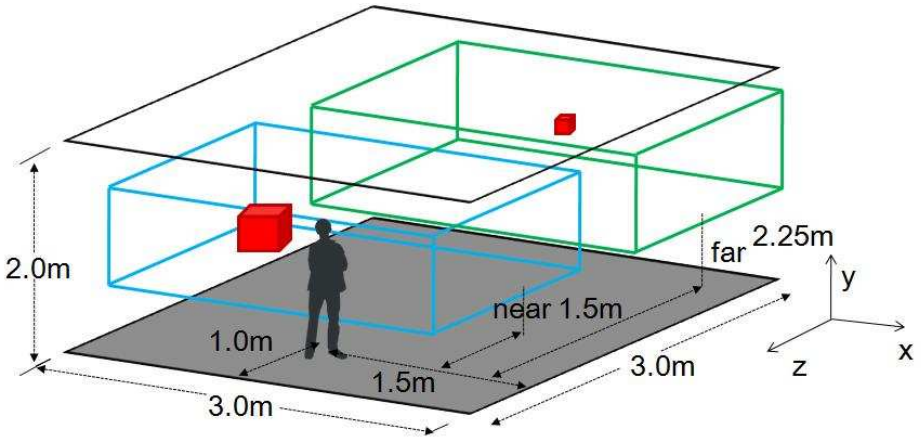

Figure 5.5: Empirical study configuration.

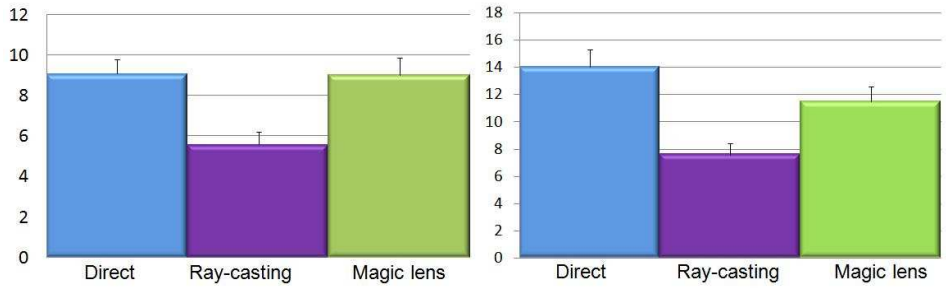

Figure 5.6: Task completion time (seconds). On the left, static, large targets. On the right, static, small targets

5.5.5 Results

6 persons, 2 women and 4 men, participated in this user study. Figure 5.6 (left) shows average task completion time with static, large targets. In this case, one-way ANOVA found a significant difference ($F(2, 177) = 9.68$, $p < 0.001$). With Bonferroni correction found that Ray-casting interface was by far the fastest. Figure 5.6 (right) shows average task completion time with static, small targets. In this case, one-way ANOVA found a significant difference ($F(2, 177) = 5.76$, $p < 0.01$). It was found that Ray-casting was faster than Direct, but no significant difference was found between Ray-casting and Magic-lens interfaces.

Figure 5.7 (left) shows average task completion time with large targets in simple motion. In this case, one-way ANOVA found a significant difference ($F(2, 177) = 24.21$, $p < 10^{-9}$). Using Bonferroni correction again, it was found that Ray-casting interface was the fastest, and no significant difference was found between Ray-casting and Magic-lens interfaces. Figure 5.7 (right)

Figure 5.7: Task completion time (seconds). On the left simple motion, large targets. On the right simple motion, small targets in simple motion.

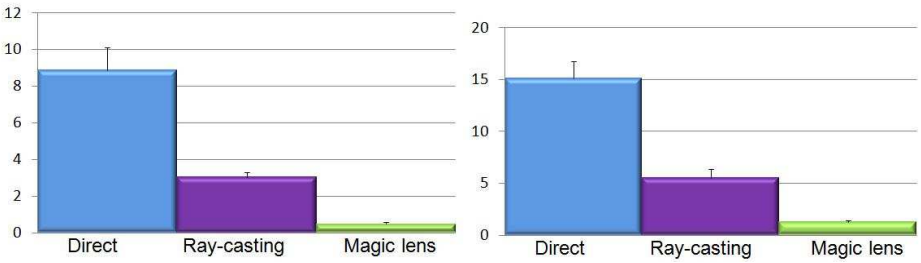

Figure 5.8: Number of errors. On the left, large targets in simple motion. On the right, small targets in simple motion.

shows average task completion time with small targets in simple motion. In this case, one-way ANOVA found a significant difference ($F(2, 177) = 10.71$, $p < 10^{-4}$). It was found that Ray-casting was faster than Direct, but no significant difference was found between Ray-casting and Magic-lens interfaces.

Figures 5.8 and 5.8 show average number of errors per trial with large and small targets in simple motion, respectively. One-way ANOVA found significant differences for these cases ($F(2, 177) = 27.91$, $p < 10^{-10}$ for large targets, $F(2, 177) = 32.38$, $p < 10^{-11}$ for small targets). Further analysis found that number of errors is the fewest in Magic-lens interface, followed by Ray-casting interface.

Figure 5.9 5.9 shows average task completion time with large and small targets in random motion. One-way ANOVA found significant differences ($F(2, 177) = 17.18$, $p < 10^{-8}$ for large targets, $F(2, 177) = 4.92$, $p < 0.01$ for small targets). In these cases, further analysis found that Direct interface was by far the slowest and that there was no significant difference between Ray-casting and Magic-lens interfaces. When the target is in complex motion, benefits of Ray-casting interface disappear and Magic-lens interface becomes one of the best. Figure 5.10 shows the number of errors produced in Magic-lens in-

Figure 5.9: Task completion time (seconds). On the left, large targets in random motion. On the right, small targets in random motion.

Figure 5.10: Number of errors. On the left, large targets in random motion. On the right, small targets in random motion.

terface was about 7 to 12% of that in Ray-casting interface and 1 to 5% of that in Direct interface. Magic-lens interface was again the easiest to perform selection regardless of target size.

Figure 5.11 shows the user's perception about the interfaces. Five out of six users considered that Magic-lens was the best, while only one considered that beam was better. All people coincide in the opinion that direct interface was the worst.

5.5.6 Discussion

When the target is static and large enough, Ray-casting interface seemed to be the best, though its relative benefit to Magic-lens interface is reduced when the target gets smaller. Regarding number of errors, no significant difference was found except that Direct interface was most inaccurate among three for static, large targets.

When the target is in simple motion and large enough, Ray-casting interface seemed to be the best, though its relative benefit to Magic-lens interface is reduced when the target gets smaller. Regarding number of errors, no significant difference was found except that Direct interface was most inaccurate

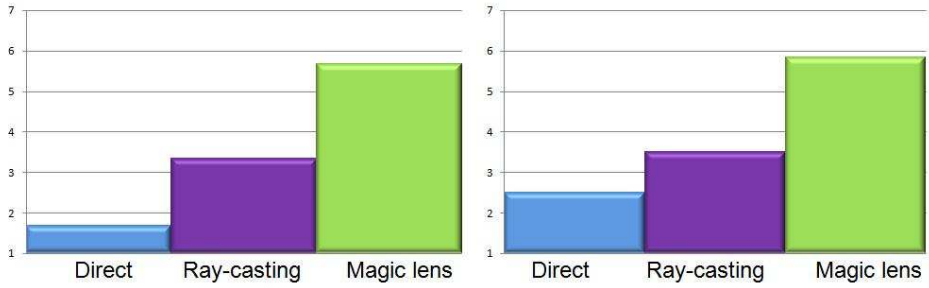

Figure 5.11: User's answer about the interfaces. On the left, results to the question: it was easy to select the interface. On the right, answer corresponding to the question: I enjoyed using this interface.

among three for static, large targets.

When a target is in motion, Magic-lens interface is the easiest to perform selection regardless of target size.

5.6 Evaluation for Manipulation

A more complete evaluation was performed for manipulation. Since manipulation is a more complex task than selection, this evaluation analyzes the manipulation task by subdividing it in two subtasks: first, the object selection and second the object manipulation which comprises the object transportation and its release into new position.

5.6.1 The Outsider Factors

The independent variables are conformed by the set of interfaces to be compared in the evaluation: Direct, Go-Go, HOMER and Magic Lens. Since Go-Go [Poupyrev 1996] and HOMER [Bowman 1997b] represent two of the most extensively studied 3D interfaces they offer a good comparison reference frame. Direct represents the case that resemble the reality of grabbing and manipulating an object.

- Task: The task consisted on collecting virtual insects. The application recreates a virtual student dormitory room with various objects such as bed, table, chairs, computer etc. An insect flying around the room appears when a trial is started. The size of this insect is of 10 cm height, 10 cm long. The insect is able to fly following the three different movement patterns:
    - simple motion: The insect flies in one of two possible trajectories lineal or sinusoidal.

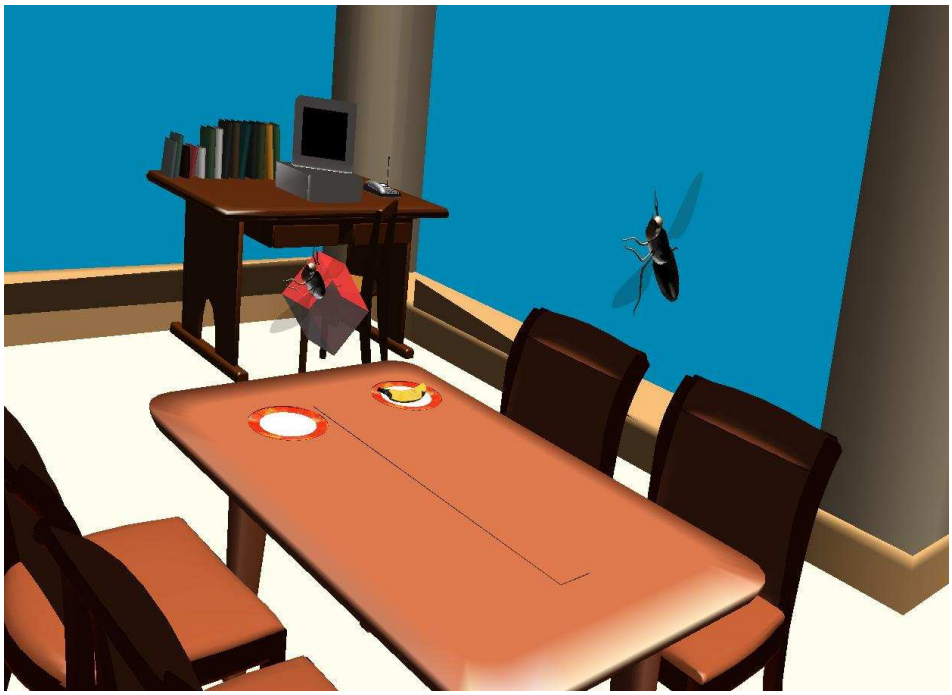

Figure 5.12: Virtual room built to perform the evaluation of manipulation.

- pseudorandom motion: The insect flies in a trajectory specified by a perlin noise function [Perlin 1985]

A semitransparent red box containing a second insect is presented to the subject. The purpose of the second insect is to show the correct position the insect should be collocated in the box. When both insects positions match, the trial is considered successful.

- Environment characteristics: A 3D software modeler was used for creating the virtual room and the geometry is exported as wavefront obj file format. Then the application imports the description file for building the OpenSG scene graph. The virtual object dimensions correspond to real object ones. Image textures are used on some object surfaces, but the model does not pretend to be realistic since the graphics card performance is limited. Object colors were carefully selected for distinguishing them easily. Figure 5.12 shows the environment.

  The system provides visual and audio feedback. On selection, the visual feedback is similar to the described in previous user studies with the difference that when an object is able to be selected the color for the edges was changed to blue. After a successful selection the edges' color changes to yellow and remains until the subject releases the insect. The audio feedback for a successfully select the insect was changed to an "yahoo" and the failure for an "ouch" sound. When the subject release

the insect an audio feedback is produced. If the insect is inserted into the box in correct orientation a sound "that's good" can be heard, if it is not correct the sound produced is "that's bad".

- User characteristics: The evaluation considers subjects with previous experience with virtual environments systems.
- System characteristics: The user study was performed with the same server, client and virtual reality equipment mention on the second evaluation.

5.6.2 Performance Measures

More information was collected for this user study. The registered metrics were:

- the time to select the insect
- the time to manipulate the insect
- the number of errors for selection and manipulation
- the subject's hand and head trajectories
- angular error after manipulation

The subject was asked to fill in a questionnaire in order to collect subjective data. The questions were formulated to be answered given a scale marked from 1 to 7 asking for the impressions that interfaces produces on the subject:

- I understand easily how the interface works
- It was easy to select the object
- It was easy to put the insect inside the box
- I enjoyed using this interface

And two questions for knowing the impression about the virtual environment:

- Virtual wold seems real.
- I understood easily the task to do.

The final question was to ranking all the interfaces in subject preference order.

5.6.3 Testbed Evaluation

Figure 5.13 shows a diagram of the user study setup. At a trial start, the subject stands up at a predefined starting point. This initial point was designed in base of the tracking system covering area. The starting point is located on the floor (0 meters y-axis), 1.5 meters in x-axis and 1 meter z-axis. From this

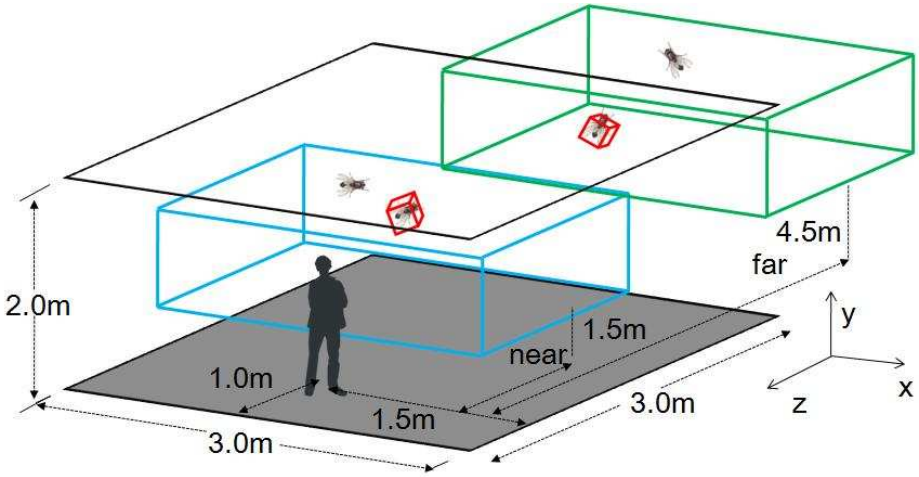

Figure 5.13: The user study setup.

point the subject is able to see the entire virtual environment. The testbed was designed to let analyze the manipulation under different conditions.

The manipulation is directly influenced for the disposition between the selected object (the insect), the red box and the subject. From the subject's starting point the distance for the insect and the box in two groups were grouped. Near distance represents a separation of 50 cm. Far distance represents a separation of 3.5 meters. Near distance is under the tracking area, therefore the four interfaces can be applied with this condition, but for far distance the Direct interface cannot be applied. On the testbed evaluation the user was asked to perform the task with the next conditions,

- The insect is near and the box is near.
- The insect is near and the box is far.
- The insect is far and the box is near.
- The insect is far and the box is far.

Additionally the testbed splits the insect distance cases in two groups, one for simple movement (on straight or sinusoidal line) and the other for complex movement (on a pseudo random trajectory). The subject was asked to perform 3 trials per condition for a total of 78 trials in a counter-balanced order. Figure 5.14 shows the combination of condition the subject performed on the experiment.

5.6.4 Hypotheses

The hypotheses regarding the usability of the interface were divided in selection and manipulation.

	Direct	Go-Go	HOMER	Magic L.
Insect Near Box Near	Simple Complex	Simple Complex	Simple Complex	Simple Complex
Insect Far Box Near		Simple Complex	Simple Complex	Simple Complex
Insect Far Box Far		Simple Complex	Simple Complex	Simple Complex
Insect Near Box Far		Simple Complex	Simple Complex	Simple Complex

Figure 5.14: Trials performed by the user for this user study. Rows represents the arrangement between the insect and the box with respect to the trial starting position. Columns represents the interfaces to be used. Subjects are asked to perform all the possible combinations with both simple and complex object movement.

For selection:

- For near distance HOMER would perform the best. The Magic lens interface would perform as well as HOMER does, because when the user would see the target through the interface the target would occupy a great portion of the screen making possible its direct selection with a pinch on the screen.
- For far distance Magic lens interface would perform the best. Selection of small object in movement could be difficult with ray-casting. The capability of taking snapshot and performing selection on them would give an advantage to Magic lens interface over HOMER.

For manipulation:

- A near object into a near box: HOMER would perform the best, because the factor that modifies the object position is almost linear for the proximity between the object and the box. Magic lens interface would perform the same as HOMER because the box and the object position would be close enough that would require few manipulations.
- A near object into a far box: Magic lens interface would perform the best. The manipulation in far distance would be easier because the user could reach the target position in one single movement. HOMER

would present a clutching problem because when the user selects a near object the moving factor is setting almost to a linear relationship.

- A far object into a near box: This case would be very similar to the previous one. Magic lens interface would perform the best. HOMER would present the clutching problems.
- A far object into a far box: Magic lens interface would perform the best. The selection distance would be close to the box position, for instance with a small movement would be possible to reach the box target. On HOMER the problem would be lost of the precision produced by the multiplication factor. Far object selection produces a big factor, that extends the object reach distance losing the precision. Go-Go and HOMER could perform at the same level.

5.6.5 Results for Selection

Eight subjects, one female and seven males, participated in the user study. The information was split into two groups: selection and manipulation. In both cases variance analysis testing (ANOVA) was applied for searching significant differences among results, and Bonferroni test for verifying those differences.

- Near objects
  - Simple movement. Figure 5.15 (left) shows the time for selecting an insect. On selection time ANOVA gives (F(3,143) = 9.201669), $p < 10^{-5}$), with average for HOMER of 6.51 sec., followed by Magic lens interface 15.48 sec. Bonferroni test found a significant difference among four interfaces. Considering the number of errors ANOVA shows (F(3,143) = 8.124883), $p < 10^{-5}$), HOMER with an average of 0.16 is the best suited followed by the Magic lens 0.64, although Bonferroni did not find a significant difference between them. Figure 5.16 (left) shows these results. For the trajectories the user walk for selecting an object. Figure 5.17 (left) shows the results. The ANOVA found (F(3,144) = 11.66193), $p < 10^{-7}$), HOMER reporting an average movement of 1.1 meters, followed by Magic lens interface with 1.44 meters. Bonferroni did not find a significant difference between them.
  - Complex movement. Figure 5.15 (right) shows the result for selection time. ANOVA found (F(3, 143) = 6.1217), $p < 0.01$). HOMER average was 7.62 sec., followed by Go-Go with 12.72 sec. but Bonferroni did not verify a significant difference between them. On number of errors ANOVA indicates (F(3, 143) = 9.6885), $p < 10^{-6}$), HOMER with an average of 2.19,

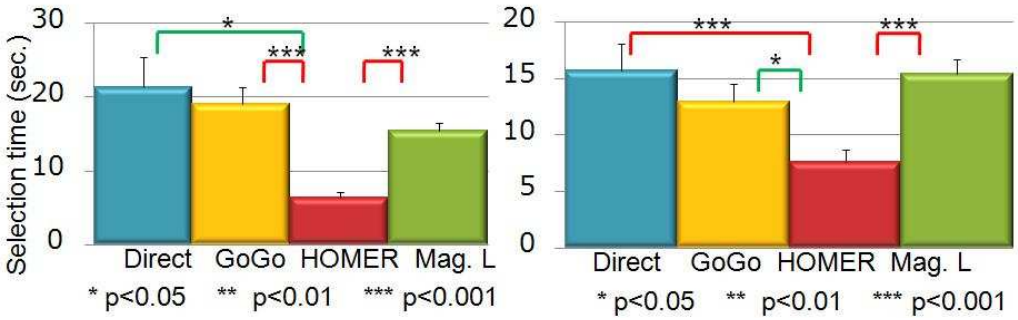

Figure 5.15: Selection times for near objects. On the left, simple movement. On the right, complex movement.

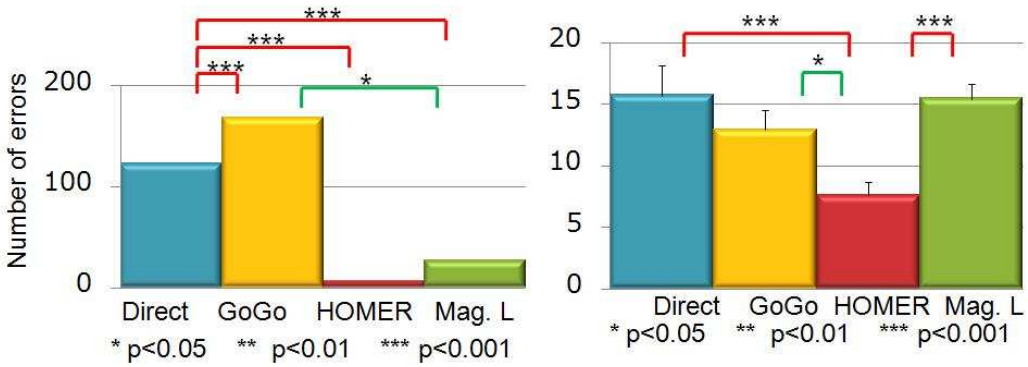

Figure 5.16: Selection errors for near objects. On the left, simple movement. On the right, complex movement.

followed by handheld magic lens with 3.26. Bonferroni did not show significant difference between them. Figure 5.15 (right) shows the result for errors. Figure 5.17 (right) shows the trajectory results. ANOVA found ($F(3, 143) = 11.66193$), $p < 10^{-7}$), HOMER with an average of 1.10 m., followed by Magic Lens with 1.4 m. Bonferroni verified a significant difference between them.

- Far objects
  - Simple movement. Figure 5.18 (left) shows the results for time. ANOVA found ($F(3, 116) = 16.86204$), $p < 10^{-7}$), Magic Lens with an average of 12.399 sec., followed by HOMER interface. However Bonferroni did not verify a significant difference between them. Figure 5.19 (left) shows the results for errors. ANOVA found ($F(3, 116) = 18.6204$), $p < 10^{-8}$), Magic Lens get less errors with an average of 1.5476 followed by HOMER with 4.52381. But Bonferroni did not show significant difference between them. For trajectory ANOVA found ($F(2, 116) = 66.3391$), $p < 10^{-20}$),

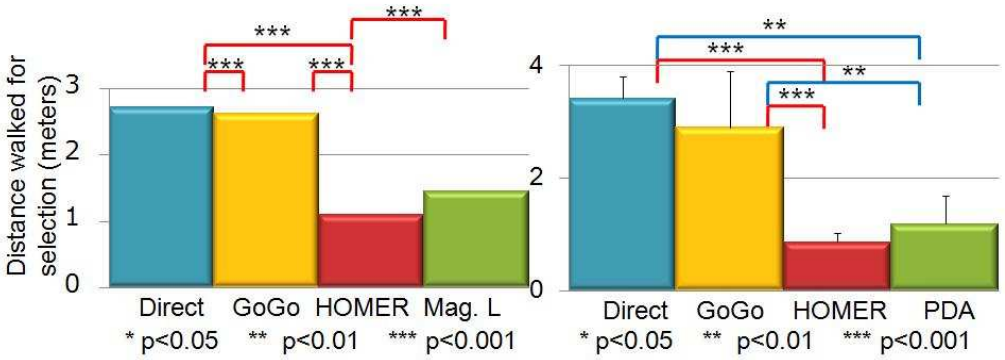

Figure 5.17: Walked distance for selecting near objects. On the left simple movement. On the right complex movement.

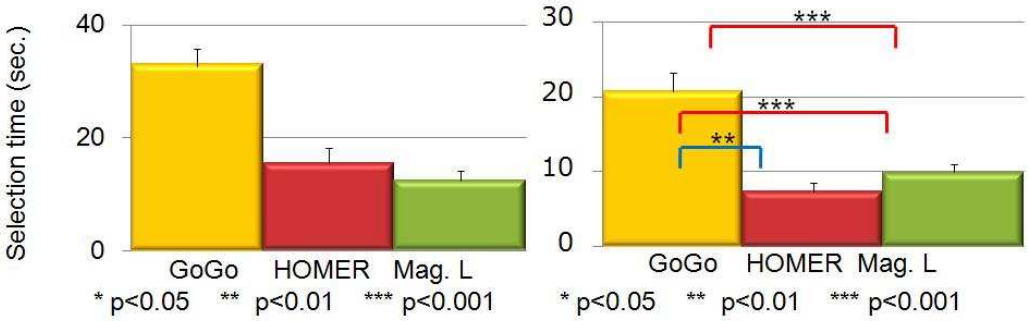

Figure 5.18: Selection times for far objects. On the left, simple movement. On the right, complex movement.

Magic lens interface got the best position with 0.7 meters followed by HOMER with 2.43 meters, although Bonferroni did not verify a significant difference between them. Figure 5.20 (left) shows these results.

- Complex movement. Figure 5.18 (right) shows the selection time. ANOVA found ($F(2,\ 109) = 21.874$), $p < 10^{-8}$), the best time was for HOMER 7.44 sec. followed by Magic lens interface 10.01 sec. although Bonferroni did not verify a significant difference between them. Figure 5.19 (right) shows the errors. On number of errors ANOVA indicates ($F(2,\ 123) = 36.08$), $p < 10^{-13}$), Magic lens interface was the best suited with an average of 0.33 followed by HOMER with 3.54. Figure 5.20 (right) shows the trajectory results. ANOVA found ($F(2,\ 109) = 93.43492$), $p < 10^{-25}$), Magic lens reports an average distance of 0.54 m. followed by HOMER with 1.03 m. However Bonderroni did not verify a significant difference between them.

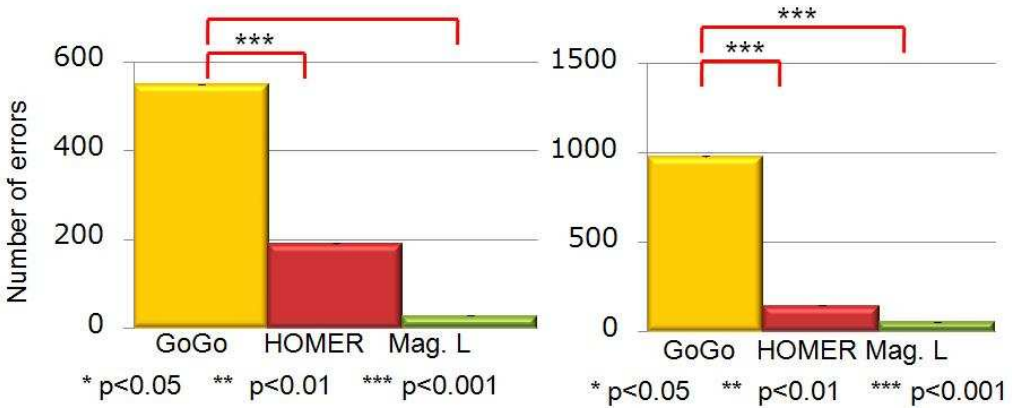

Figure 5.19: Selection errors for far objects. On the left, simple movement. On the right, complex movement.

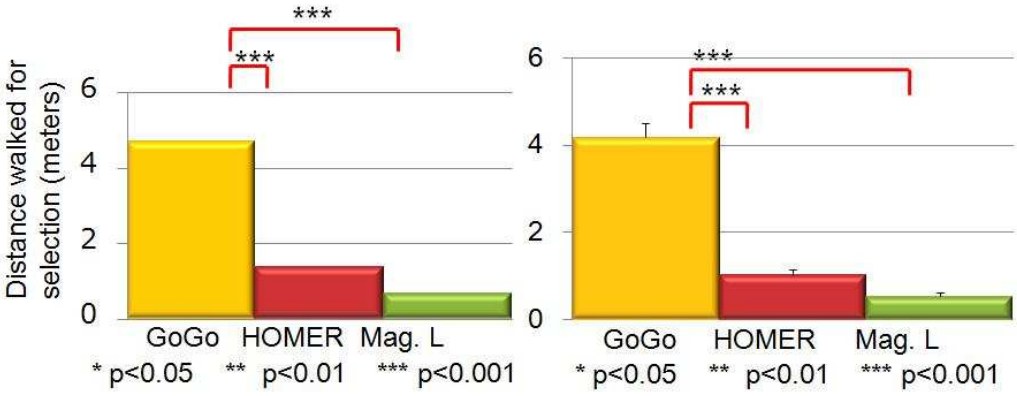

Figure 5.20: Walked distance for far objects. On the left simple movement. On the right complex movement.

5.6.6 Results for Manipulation

The manipulation time represents the time the user moved the object around the space in order to insert the insect into the box. The time, the number of errors, and the trajectory walked by the subjects were registered.

- Near object, near box. Figure 5.21 (left) shows the manipulation time. ANOVA indicates ($F(3,238) = 60.0849$), $p < 10^{-29}$). The best interface was Go-Go with 3.2 sec. followed by Direct 3.48 sec. But Bonferroni did not provide a significant difference among Go-Go, Direct and HOMER. Figure 5.22 (left) shows the number of errors. ANOVA did not indicate a significant difference among four interfaces ($F(3,164) = 2.1320$, $p < 0.09$). On trajectory ANOVA did not find a significant difference. Figure 5.23 (left) shows the distance walked by the user. ANOVA find

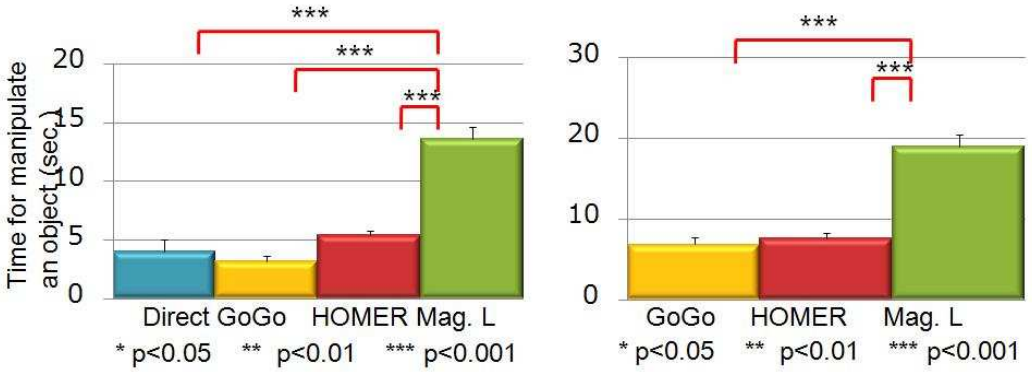

Figure 5.21: Manipulation time. On the left, near objects inserted into near boxes. On the right, near objects inserted into far boxes.

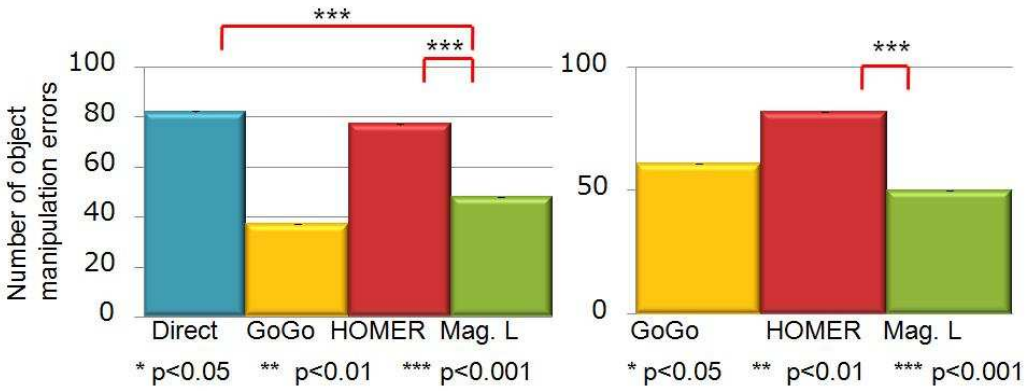

Figure 5.22: Manipulation errors. On the left, near objects inserted into near boxes. On the right, near objects inserted into far boxes.

($F(3,238) = 3.1118$), $p < 0.028$), Magic lens with 0.9634 m. followed by HOMER with 1.17 m. However Bonferronin did not provide a significant difference among four interfaces.

- Near object, far box. Figure 5.21 (right) shows the manipulation time. ANOVA found ($F(2,192) = 47.669$), $p < 10^{-17}$). The best interface was Go-Go with 6.85 sec. followed by HOMER 7.64 sec., however Bonferroni did not validate this difference. The number of errors is shown in Figure 5.22 (right. ANOVA indicates a significant difference ($F(2,123) = 3.5389$, $p < 0.03$) the Magic lens interface got the best average value 1.11 followed by Go-Go, but Bonferroni did not support the difference. Figure 5.23 (right) shows the trajectory results. ANOVA found ($F(2,192) = 14.92137$), $p < 10^{-06}$), Magic lens whit a distance of 1.5 m followed by Go-Go with 2.72 m. However Bonferroni did not very the difference between both.

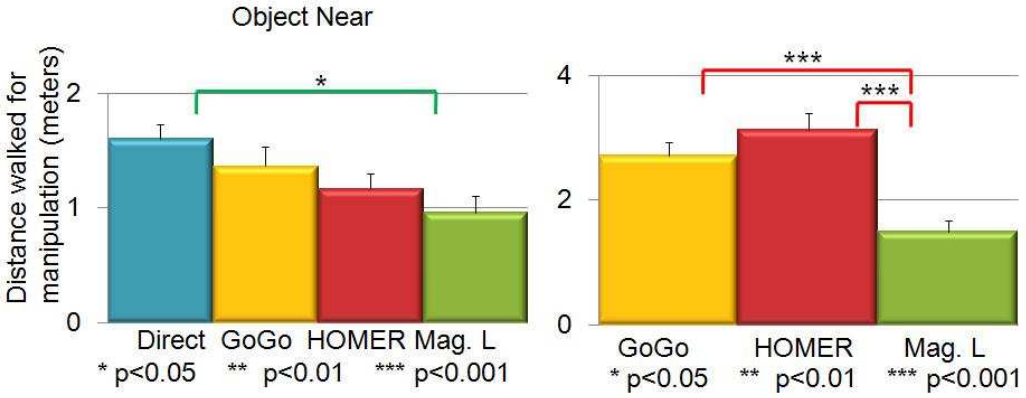

Figure 5.23: Walked distances for manipulating objects. On the left near objects inserted into near boxes. On the right near objects inserted into far boxes.

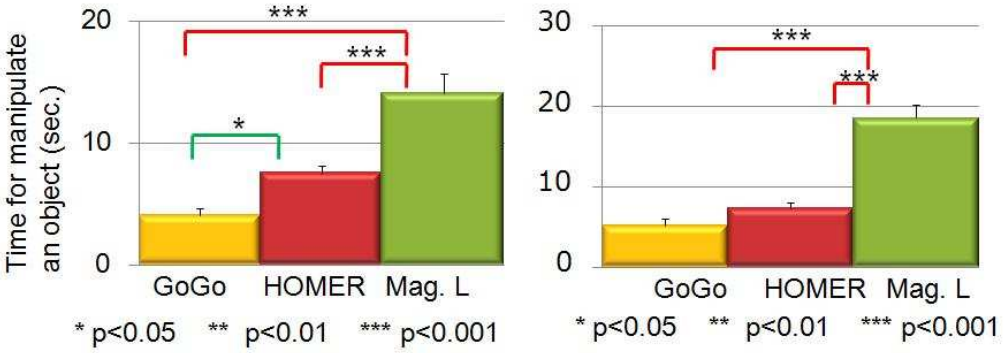

Figure 5.24: Manipulation time. On the left, far objects inserted into near boxes. On the right, far objects inserted into far boxes.

- Far object, near box. Figure 5.24 (left) shows the graphs for manipulation time. ANOVA indicates a significant difference ($F(2,163) = 24.9699$, $p < 10^{-10}$), Go-Go was the best interface with 4.04 sec. followed by HOMER with 7.5 sec. However Bonferroni did not verify the difference between both. Figure 5.25 (left) errors. ANOVA indicates a significant difference ($F(2,163) = 4.7914$, $p < 0.0099$), Magic lens with an average of 0.9285 errors, followed by Go-Go with 1.071. However Bonferroni did not verify this difference. For trajectory ANOVA did not find significant differences. Figure 5.26 (left) shows the results for the walked distance. ANOVA did not find differences ($F(2,163) = 3.3478$, $p < 0.0385$).
- Far object far box. Figure 5.24 (right) shows the time manipulation. ANOVA indicates a significant difference ($F(2,163) = 44.5027$, $p < 10^{-15}$). Go-Go was the best interface with 5.19 sec. followed by

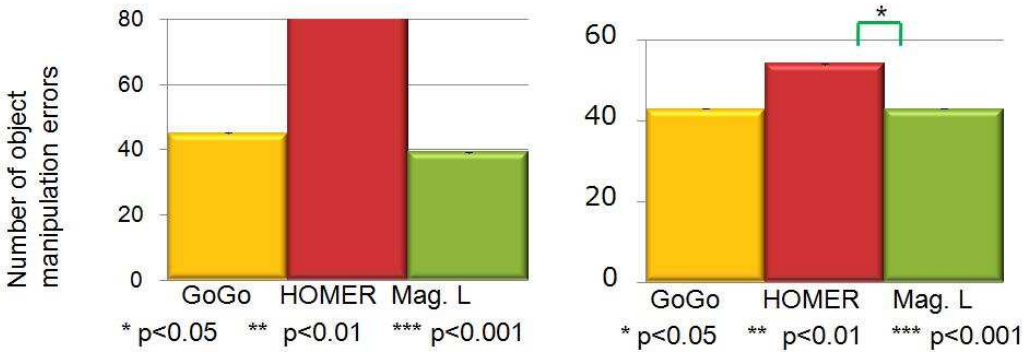

Figure 5.25: Manipulation errors. On the left, far objects inserted into near boxes. On the right, far objects inserted into far boxes.

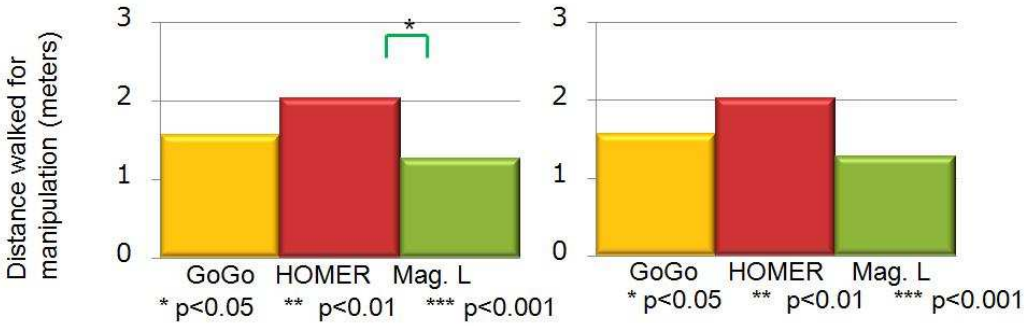

Figure 5.26: Walked distance for manipulating objects. On the left far objects inserted into near boxes. On the right far objects inserted into far boxes.

HOMER with 7.37 sec, but Bonferroni did not support this difference. In number of errors ANOVA did not find a significant difference among the three interfaces ($F(2,123) = 0.4952$, $p < 0.4952$). Figure 5.25 (right) shows these results. Figure 5.26 (right) shows the trajectory results. ANOVA did not find significant difference ($F(2,123) = 0.195$, $p < 0.823$).

5.6.7 Results for Task Completion

Considering the complete task which comprises selection and manipulation. Measuring the time the results are as follows:

- Near object near box. ANOVA indicates ($F(3,135) = 11.41307$, $p < 10^{-6}$). The best interface was HOMER with 19.48 sec. followed by Go-Go with 21.42 sec. Bonferroni did not find a significant difference between Go-Go and HOMER but there are differences among the others

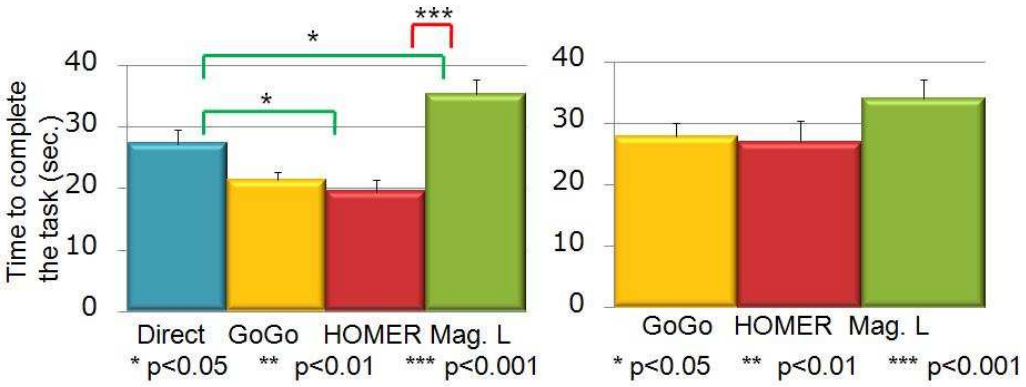

Figure 5.27: Task completion time. On the left, near objects inserted into near boxes. On the right, near objects inserted into far boxes

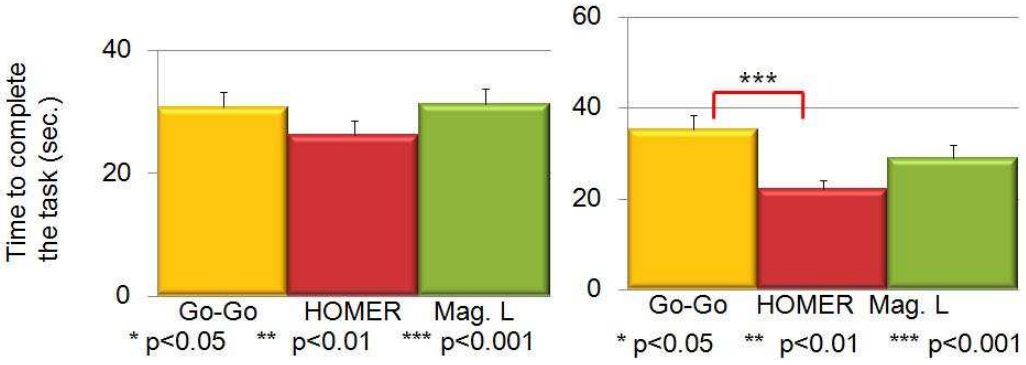

Figure 5.28: Task completion time. On the left far objects inserted into near boxes. On the right far objects inserted into far boxes.

interfaces. Figure 5.27 (left) shows these results.

- Near object far box. ANOVA did not find a significant difference among the interfaces ($F(3,135) = 1.7962$, $p < 0.1736$). Figure 5.27 (right) shows the results.
- Far object near box. Figure 5.28 (left) shows the completion task time. ANOVA did not find significant difference among interfaces ($F(3,135) = 1.328$, $p < 0.2709$).
- Far object far box. ANOVA indicates a significant difference among the interfaces ($F(3,135) = 7.2212$, $p < 0.001$). HOMER performs the best with 22.12 sec. followed by Magic lens with 28.9821 sec. Figure 5.28 (right) shows the results.

All subjects answered a questionnaire at the end of the user study. Figure 5.29 registers the perception generated by each interface on the subjects.

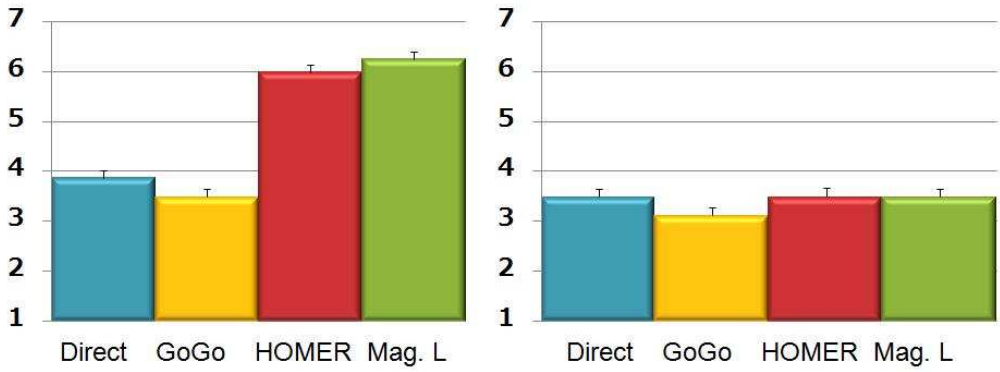

Figure 5.29: Graphs showing results from questionnaire. On the left, the perception of how easy was to select the object. On the right, the perception of how easy was to manipulate the object. A one on the scale represents great difficulty, and a seven means great facility.

5.7 Discussion

For selection, the results support the hypothesis that HOMER is a very effective interface for selecting objects that are close to the user. It was found that HOMER performs very well for object with complex movement contrary to the original hypothesis. Subjects using the Magic lens interface takes time to select the object with the handheld device for two reasons:

- they tried to follow the object position with the handheld device or
- they wait in a static position along the path waiting for the moving object to appear inside the screen.

It is very important to consider the relationship between the distance of the object with its size. An object that is the size of the handheld device in a near distance covers entirely the mobile display, the user is forced to separate from the object in order to capture it in the screen.

For far objects the performances of HOMER and the Magic lens interface were similar. The size of the target are big enough to be selected with HOMER even when it is moving in a chaotic pattern. However there is a big difference considering the number of errors. With HOMER the subjects found it difficult to select the target with only one shot. Subjects tried a lot of times until the correct object could be selected. Most of the time consumed by the subject with the Magic lens interface was on taking the snapshot. For objects moving in a simple pattern taking the snapshot was very easy but in complex movement sometimes the object suddenly went out of the screen area.

Contrary the original hypothesis, although is not statistically well supported, it seems that Go-Go shows a little advantage for manipulation on both pre-

vious cases. The user found it very difficult to select objects with Go-Go but easy to manipulate them. With HOMER the problem was related with clutching. When the user selects a close object for later inserting it into a far box the factor applied to the manipulation was so small that the subject must release the object, select it again and repeat until the object could reach the box. On the far object the case was vice versa, the factor was so big that the user could not manipulate it correctly in close distances. The weight and size was a factor that the subjects complain about.

On these cases again, although is not statistically supported Go-Go has shown a small advantage. Go-Go again represents difficulties for selecting objects but, after the selection, manipulation was very comfortable. Reviewing the results of the completion task times, Magic lens interface could not perform as well as was expected. Subjects found the device too big and heavy, and they get tired very quickly on manipulation. Another less noticeable problem was the delay. The interface has a delay of around 150 milliseconds producing a small difference between the position sensing and rendering on the device screen, and it varies according to the complexity of the scene. When the subject was manipulating the object in far distances this delay produced confusion for determining the current object position.

5.8 Summary

There is not a standard evaluation method for 3D interface development because many factors influence the effectiveness of an interface. It is difficult to design experiments that controls all the aspects related the performance of an interface. However, previous research have provided guidelines to create experimental testbeds which can be used to evaluate the performance of an interface by comparing it against similar interfaces. In this research this testbed approach was applied. The three prototypes developed along this research were compared with well-known previous interfaces.

For the first prototype, a magic lens implemented in PDA using JPEG video stream mechanism, the interface to compare with were Direct and Ray-casting. Because the evaluation was performed on an early stage of the interface development, only selection of objects was evaluated. A combination of sparse and dense static object environments were built and used in the evaluation. The results of the first evaluation shown that the interface offers a good performance for selecting objects. The capability of selecting object from snapshots provide more accuracy.

The second prototype improved the JPEG video stream mechanism. New environment conditions were tested. Static and moving objects located at two different distances and with different sizes provided a different set of conditions. Previous interfaces have shown limitations for selecting tiny and

far away objects. The magic lens interface shown a similar performance to Ray-casting for large objects, and advantage for small objects located at far distance from user.

On the third prototype, the PDA was replaced by an ultra mobile PC. Moreover the video stream mechanism was replace for MPEG technology. A more complete user study with a complex environment were developed. Selection and manipulation of virtual objects were considered with this prototype. Direct, Go-Go and HOMER interfaces were compared against the magic lens approach. The task consisted on select flying insects and then, put them into a box. Different separation distances were tested among the insect, box and user. The results confirmed that magic lens approach has similar performance that Ray-casting for selecting objects near the user, and it shows a better performance for tiny and objects located at far positions from user. However for manipulation the result shown that Go-Go was the best interface. The delay combined with the technology employed to implement the prototype could impact negatively the performance of the magic lens. More experimentation is necessary in order to analyze the performance in manipulation and implement new prototypes with smaller and lighter technologies.

Chapter 6

Discussion

6.1 Introduction

This Chapter comprises a discussion about the concept of the interface, its originality, the problems that still faces, the possible improvements and future work.

6.2 Conceptual Model

With magic lenses it is possible to see different views of the environment. They works as magnifying lenses do in reality, the characteristics of the lenses modifies the visual perception of the objects behind of them. By modeling virtual lenses it is possible to freely define their properties, form and characteristics. In [Bier 1993] the lens is modeled as a rectangular transparent window and it changes its properties with the support of a tool pallete. The user selects a functionality from the pallete and immediately the lens gain the capability of modify the view according to the selected functionality. This mechanism offers a broad the gamma of different views, that additionally can be combined. The lenses are not restricted to rectangular 2D windows. In [Viega 1996] lenses are implemented as a 3D virtual cube able to explore volumetric data.

Virtual lenses offers the flexibility of changing and combining multiples views, however as any virtual element they suffer of the same limitations: incorrect depth cues, occlusion problems, no physical feedback.

Other approach to implement the magic lens metaphor is proposed in [Schmalstieg 1999]. The inclusion of a physical prop that represents the lens provides physical stimulus that enhance the interaction. In this implementation a Plexiglas tablet with a tracker sensor attached to it represents the "real" part of the lens. The system calculates the area covered by the lens and it draws an interface on that area representing the "virtual" part of the lens. Then, the user can interact with the interface by touching the tablet using a pen that has a tracker sensor attached to it. The advantage of this approach are:

- the user manipulates the lens using a "real" object

- the interaction is supported by virtual controls. The controls are manipulated by touching the tablet with the pen. This mechanism provides important physical touch stimulus.
- The properties of the lens are modified by pushing the controls represented in the virtual interface.
- Integration of image-plane interaction technique [Pierce 1997] with the support of the tablet and the pen.

This approach carries several advantages over the original magic lens approach, however the tablet provides exclusively a "real" representation of the glass and touch feedback. The interface and controls are still virtual objects integrated inside the virtual environment, limiting the interaction.

The originality of the work presented in this dissertation is to separate completely the lens and controls from the virtual environment, for implementing them totally in a handheld device. We consider that this approach gives more flexibility for creating new interactions and provides the next advantages:

- The interface forms an unit. The lens, controls and visual interface are implemented inside the handheld device.
- The interface is easily manipulated because is implemented in a real object, the handheld device.
- The handheld device offers "real" physical controls (buttons) and a touch screen that provides touch feedback stimulus.
- With the handheld device, the lens is able to take snapshots, manipulate and stored them for future reference.
- With the handheld device it is possible to implement the image-plane interaction techniques directly as the props does, but with the extension of using snapshots. This mechanism improves the accuracy for some task such as object selection.
- The handheld device offers a flexible development platform. It is possible to create interfaces with controls represented on the handheld device's screen and furthermore with touch screen support.
- Some handheld devices offer other kind of stimulus that can be inserted in interactions such as accelerometer, sound, vibration.
- Easy integration of 2D interaction tasks into 3D spaces such as text writing and reading, hand drawing, Internet surfing and so on.

The implementation of the magic lens in the handheld device provides a "real" touchable magic lens for interacting within virtual environments. A full set of 2D interactions provided by the handheld device are easily integrated into 3D virtual environments. And as HOMER [Bowman 1997b], it is possible to combine two or more interactions for creating new ones. One example of this

characteristic was described at Chapter 4 for manipulating virtual objects. First, an image-plane interaction technique using snapshot is used for selecting the virtual object and then a Go-Go interaction movement provides the object manipulation.

6.3 Prototype implementation

Three prototypes of the magic lens interface were built along this research. They are described in Chapters 3 and 4. The main drawback that lead to the development of the three prototypes was network latency. By changing hardware and software components the initially 2 s. latency was reduced to 127 ms. Latency was measured by taking the time between the event of sending an image from the server machine and the event of receiving an acknowledge packet produced by the client when it receives that image. Last implementation, showing the best performance, employs MPEG streaming technology using a ultra micro PC as handheld device. 25 fps with resolution of 800*600 24-bit color images provides a pleasant magic lens effect for exploring virtual environments.

In order to extend the magic lens for using it as a selection and manipulation interface a software architecture was designed and implemented. Then, user studies were performed, changing the environmental conditions in each one. For investigating the selection of virtual objects, first, an environment with static objects in a sparse and dense layout was implemented. The intention was to determine if the interface is effective under occlusion and imperfect depth cues problems. The Figure 5.3 shows the results of comparing the magic lens interface againts virtual hand and ray casting interfaces. For sparse layout the magic lens interface got a poor performance at the beginning, but after the second trial it becomes the better one. By observing the subjects performing the first trial, we noticed that they faced difficulties to select the first object. The button for taking snapshots was confused easily. However on the second trial subjects learned how to use correctly the interface, and the results reflects the improvement on time. For dense environment, the latency of 2 s. together with the poor 6 fps rate were big drawbacks that lead the subjects took many unsuitable snapshots for selecting the object. This problem was reduced in the second and third implementations.

The second user study was oriented to test the performance of the interface for selecting objects with different size, distance relative to the subject and object movement. The Figures 5.6 through 5.11 showed the result of the user study, confirming that the magic lens interface performs better for selecting object of small size, located distant from the subjects and in movement. The selection from the snapshot produce noticeable less errors, and provides more acurracy. When the object was near the subject the best performance was for

the ray-casting interface, the magic lens interface could not perform at the same level because when the object is clearable identifiable pointing it with ray-casting is quicker than searching it with handheld device's screen.

The third user study was designed to confirm the previous selection results and at the same time to test the object manipulation task. A new environment and task were created with this purpose as was described in Chapter 5. Results were separated in two groups for easier analysis. The Figures from 5.15 to 5.20 showed the results for selection meanwhile the figures from 5.21 to 5.26 showed the results for manipulation. On selection the tendency showed by the second user study repeated. Magic lens is a good approach for selecting objects located far away from subjects and specially following a random trajectory. But when the object is near the subject the ray-casting (HOMER) is the best interface. For selecting a near object with the magic lens the subject took long time for capturing a suitable snapshot. Then, the relationship between the size of the target and the size of the handheld device has a big influence on the number of failed snapshots. The field of view of the handheld device is too narrow to capture near objects that lead to the need of taking great number of snapshots. The next section discuss a possible method for alleviate this problem. On manipulation although the magic lens is based on Go-Go interaction technique the result showed that it performs below Go-Go itself. Because the interaction mechanism was implemented using the same software components and with exactly the same physical movement approach, by analyzing the differences the result could be influenced that:

- The network latency. 127 ms. can be good enough for exploring objects and even for selecting them. However, as the third user study was designed for setting the object in a fixed position and orientation the manipulation required higher level of accuracy. Under this environmental characteristics the limitations of magic lens interface became evident. Outside the threshold line a small movement on the handheld device is amplified, situation that increase as the object is located farther away from subject. Then, the delay of 127 ms. produces unaccurrate location for the object because the tracker sensor registered a position and the mobile display shows the 127 ms. image in the past.
- The field of view. When the object is near the subject it was difficult to capture the target with the handheld device. Two possible solutions for this problems are: first, the possibility of adding a zoom in and zoom out capability to the handheld device view. The second, to combine the handheld device magic lens virtual magic lens with a wider field of view.
- The size and weight of the handheld device. Unfortunately the design of the user study does not consider to measure muscular tension or other biometric information to establish the relation of the weight of

the device with the physical effort. But many subject complaint at the end of the user study about the weight of the interface, 832 g. represents a heavy device to be carried for long period of time. The solution to this problem is to implement the interface using more light components.

- Handling the interface with both hands. This problem is closely related with the previous one. In the original design the magic lens can be carried with one hand, giving freedom to the other hand to select elements or controls inserted in the interface. In the third implementation the device was so heavy that the original interface was modified to carry the device with both hands. The result was that the subjects found difficulty to select the virtual object because there was no free hand.
- Head mounted display image mismatch. For the three user studies a see-through head mounted display was employed. Images are generated in front of the subjects eyes, and through the lens the subjects was able to see the images on the handheld device screen. The mentioned network latency produced the undesirable effect of image mismatching, most noticeable when the mobile devices was suddenly moved, making difficult the selection and manipulation. By reducing the latency this problem would tend to disappear, however differences on resolution and color can continue. The definitive solution to this problem is to use a different technology for create the virtual environment. By using CAVE or other projection based technology system, this problem is not present.

6.4 Future work

The analysis of result and the different discussions lead to the proposal of several improvements. They are compiled as future work.

6.4.1 Migration to a smaller device

New tendencies in cellular phones and music players present continuously smaller and lighter devices, including more powerful graphic capabilities. For example, iPhone cellular phone weight is 135 g. capable of play MPEG-4 video at 30 fps and a wireless network 802.11b/g and additionally offering accelerometer, GPS and proximity and ambient light sensors. The size of the display area is smaller (115*62 mm) but this limitations can be handle by using virtual magic lenses that complement the narrow field of view. On tracker devices the optical technology is becoming more robust. The OptiTrak FLEX 100 tracker device provides a latency on tracking of 10 ms. in a range of 7 meters using 6 cameras. Tracker cameras search for plastic metallic color trackers that are very light.

6.4.2 Integration of virtual magic lenses

Tacking advantage of the tracker sensor attached to the handheld device, a virtual magic lens attached the handheld device can be created. The virtual magic lens will represent an extension of the device' screen. The Figure 6.1 shows a diagram of the design. The size and properties will be adjustable. The form of the virtual lens will be not limited to rectangular areas.

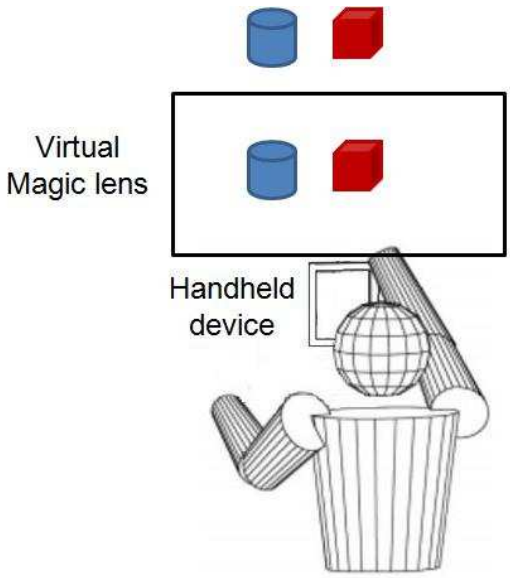

Figure 6.1: A rectangular magic lens

The Figure 6.2 shows a circular virtual magic lens.

By creating a virtual magic lens the limitation of the original magic lens comes up again. The user does not have any tactile feedback. In order to solve this problems the interface can implement two solutions.

- To establish a scale correspondence between the handheld device screen and the virtual magic lens. In this approach the screen on the handheld display will represent a smaller scale version of the entire virtual magic lens. The disadvantage is the accuracy will be lost as the size of the virtual magic lens becomes bigger.
- To decouple the virtual magic lens when the user wants to perform a selection or manipulation for using the handheld device as a magic lens over the virtual magic lens. This can be a good solution, however it has a drawback, the level of redundancy can lead to confusion on the users. With an extra support of a 2D interface to grab the virtual magic lens and manipulate it, this proposal can become the best one. The Figure 6.4 shows a virtual magic lens decoupled from the handheld device.

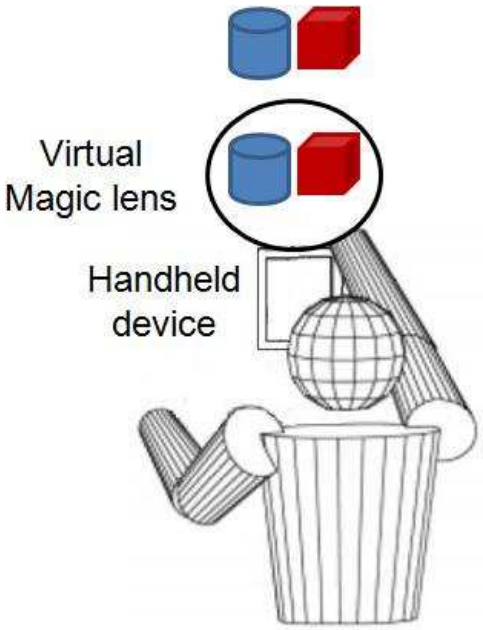

Figure 6.2: A circular magic lens

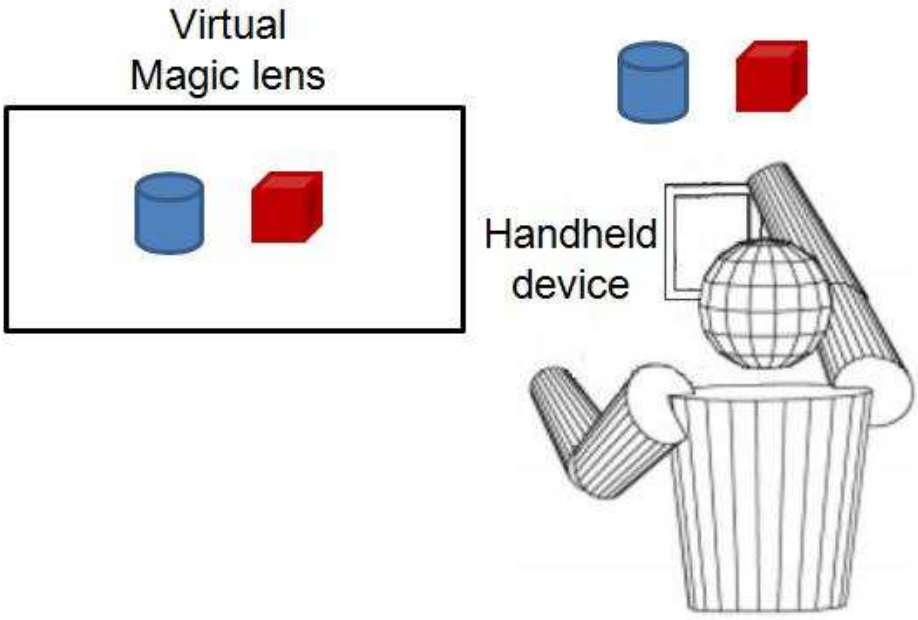

Figure 6.3: A fixed magic lens decoupled from the handheld device

6.4.3 Automatic switching of metaphors

Other extension for the interface would be to modify the behavior of the interface according to the orientation of the handheld device. When the mobile device is facing to the user is clear that she/he wants to use handheld as a magic lens. However, if the handheld is rotated 180 degrees in Z-axis becoming the handheld device screen upside-down, then, because the screen can not be seen the handheld can be used as a stylus pointer, and the interaction can be switched to a ray-casting or Go-Go. By pushing the touch screen the user can represent pussing button action in a normal stylus.

The orientation of the handheld device can be used to establish different manipulation metaphors as well. For example, in manipulation, the predefined interaction technique to be used would be Go-Go. But if the mobile device is rotated 90 degrees the interaction can be changed to virtual hand for a more controlled and accurate movement. Far away located object can benefit of the automatic creation of a virtual magic lens displaying a zoom in view of the manipulation space.

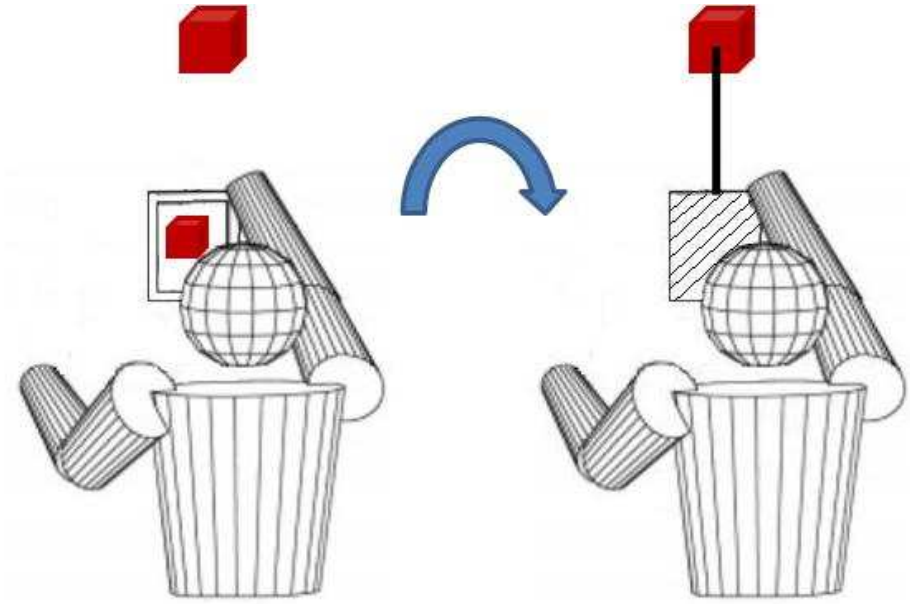

Figure 6.4: Changing interaction technique. On the left, the magic lens metaphor. On the right, when the user rotate the handheld device 180 degrees in Z-axis the interaction technique changes to ray casting.

6.4.4 Integrating navigation

In order to offer the most basic interaction functionalities in an immersive virtual environment, the interface should provide navigation. By implement-

ing the World in Miniature, the user will view a smaller version of the entire environment in the handheld device screen. By scrolling and slicing, the user would be able to zoom in, zoom out and navigate the environment as using a map in Internet. When the user finds the location that she/he wants to go, by touching the place on the handheld device's screen the user will be teleported to that location in the virtual environment.

The user would be able to select and manipulate virtual object using the objects shown in the miniature environment presented on the handheld device.

6.5 Summary

This chapter complements the evaluation of the interface presented in Chapter 5. It remarks the originality of the research which consists on giving more realism to the implementation of a magic lens using a handheld device. Previous studies proposed its implementation by using pure virtual components and a mixure of real-virtual elements. Moreover, the chapter compiles observations, comments and suggestions related to the concept, design, implementation and evaluation of the interface. The network latency has been the main problem that had led the development of three different implementations. However, more improvements should be made in order to offer an effective interface for manipulating virtual objects. The weight and size of the interface are two other drawbacks. Using a more modern lighter handheld device and tracker sensor could resolve this problem. The field of view of a smaller device can be compensated employing virtual lenses.

Magic lenses implemented in mobile device offer the flexibility of integrate 2D and 3D interaction techniques within virtual environments and it can represents a good approach for standarizing 3D interfaces.

Chapter 7

Conclusions

The objective of the present study was to develop a 3D magic lens interface with the aim of providing new methods of interaction within virtual reality environments. The novelty of the study consists of proposing a new interface design implemented on a handheld device. The touch screen, graphic display, programmable 2D interfaces, networking, sound, vibration and all the other device functions are integrated into the development of new 3D interactions. Using this approach, the handheld device becomes a flexible and powerful interaction tool by combining, extending and giving new definitions to previously isolated 2D and 3D metaphors.

The interface takes advantage of the familiarity levels resulting form the use of mobile technologies. PDA's and cellular phones are, in modern life, devices employed in daily activities. People are, therefore, "accustomed" to the interactions and capabilities they provide. The interface applies the 2D interactions user have previously mastered, to the creation of 3D interactions. Taking snapshots and storing them on a clipboard resembles the use of a digital camera. The selection of the object from a snapshot is similar to selecting a telephone number from the screen of a handheld device, and so on. The use of 2D interactions for performing 3D activities minimizes the interaction space and operations requiring high precision. For example, selecting an object from a snapshot is easier than trying to select it directly from 3D space. Furthermore, the mixture of 2D and 3D interaction provides a more comprehensive set of functions that complement each other. So for example, in situations where manipulation in 3D environments requires more freedom of movement, the 2D interface expands this capability. In these situation, an elongation metaphor provides a greater range of movement.

Several 3D interactions have been developed previously. However their application is limited to specific conditions resulting in the proliferation of interfaces and input devices. This study pursues the creation of an interface that offers a generic set of interaction techniques that can be used in a wide range of environments and conditions. The study moves in the direction of searching for a generic method of interaction in 3D environments. The interface facilitates the application of basic interaction tasks as fundamental operations, which are then used as the foundations for more complex interactions.

This study proposed an architecture for implementing the magic lens metaphor using a handheld device. The design involves the sharing of a

computing task between the handheld device and virtual environment generator computer system, the integration of a tracking sensor for calculating the handheld device position, and the communication infrastructure required to exchange images and interaction information. After considering the differences in processing and storage specifications between several handheld devices, and with the aim of offering a imminently portable architecture to a wide number of mobile technologies, the distribution of the task proposed utilizes the pixel distribution approach. In this approach, the virtual environment render machine generates all the images used by the system, including the images displayed on the handheld device. Ater which, a client server architecture is implemented where the render machine has the role of server and the handheld device assumes the role of client. The server senses the position of the handheld device with the tracking sensor, then it calculates the portion of the image the handheld device is covering in the environment. A scheduler captures images at predefined intervals of time in order to generate a video stream that is sent to the client. The client captures interactions provided by the touch screen, buttons, and 2D interfaces, and transmits them to the server in order to evaluate the captured information for the modification of the environment.

Three prototypes were fabricated to validate the feasibility of the proposal:

- a JPEG based streaming prototype with the support of OpenGL/ES for 2D interface design.
- a JPEG based streaming prototype with the support of GAPI for 2D interface design.
- and a MPEG based streaming prototype with the support of Qt for 2D interface design.

A performance test, applied to the prototypes, showed that the last prototype, employing MPEG technology, was the most suitable for implementing real-time interactions. The results for measuring the delay was an average of 127.8 m. transmitting images of 800*600 24-bits color at 25 fps.

After the design of the basic magic lens interaction, an architecture for implementing 3D interactions that possessed the flexibility required for mixing and integrating them around the magic lens metaphor, was implemented. The architecture neccessitated the creation of a software layer to distribute the input information to specialized interaction modules based on the context of the interaction. The same input information can perform different actions depending on the task the user is performing at a certain determined time. Pushing a button can mean, for example, select an object or to change the color of an object. The architecture implements the different definitions separately and redirects the input to the corresponding processing module for

correct interpretation. The result is a flexible software layer that can be easily extended in order to support different interface types.

A prototype implementing the proposed architecture was fabricated. The prototype was implemented on top of the existing magic lens prototype. Interfaces such as virtual hand, ray-casting, Go-Go and HOMER were implemented with the architecture. The magic lens interaction was mixed with a Go-Go interaction technique in order to improve the manipulation of virtual objects in 3D space. This architecture was found to help integrate different interaction methods, offering an environment suitable for experiments with interface combination and extensions.

During the design of the interface, interaction evaluation were performed.

For the first prototype, a user study suggested that the interface is easy to use for selection and that selection accuracy is markedly improved. For the second prototype, the user study revealed that the interface is considerably easier to use when selecting an object, compared to the ray casting technique, when the target object is in motion. This is because there is no need to accurately track the moving target with our interface. Instead, the user only has to capture a snapshot of a large portion of the environment that includes the moving target. Due to the pen-tablet technique based interaction, the size of the object does not matter for precise selection. Finally, in the third prototype, the user study confirmed that the interface has good object selection performance, specifically when those objects are moving and are distant from the user. During manipulation, the primary problems with the interface were its weight and size. A new prototype implemented with tracker device that is smaller in size and a mobile component lighter in weight will be fabricated and evaluated in future work. Based on the user study results, applying the Go-Go in combination with the magic lens appears to be a good approach for manipulating objects. However more user studios must be conducted in order to confirm or refute this contention.

In the future, the interfaces will integrate the World in Miniature interaction. It is expected that the use of the handheld device for displaying a miniature version of the environment would offer a more flexible form of interactions. The use of maps for localize an address is an actual metaphor the interface is trying to offer in 3D virtual environments. Navigation by using previous marked locations stored in the clipboard can be used for creating a path to interesting view that the user could immediately return to by making a selection on the mobile display. Accelerometers, which are included in some handheld devices, can be integrated to automatically switch the metaphor employed. When the mobile device is facing directly to the user's face, it could inidcate the user wants to use the magic-lens effect. When the screen points away from the user, it could mean the user wants to use another interaction technique, such as ray casting or HOMER. The overall goal of the interface is to offer an interaction mechanism easy to understand and apply

by integrating handheld device as tool for generating meaningful interaction techniques. The vision governing the application of this interface is that the user should be able to interact immediately within an immersive virtual environment while supported only a handheld device and a tracker sensor. In this vision the user intuitively would discover the functions offered by the interface without requiring long training periods. Thus, the interface will function as an important tool for the environment the user perceives virtually.

Bibliography

[Ayatsuka 2000] Y. Ayatsuka, N. Matsushita and J. Rekimoto. *HyperPalette: a hybrid computing environment for small computing devices.* In CHI '00 extended abstracts on Human factors in computing systems, pages 133–134, 2000. 7, 8, 11

[Bier 1993] E. Bier, M. Stone, K. Pier, B. Buxton and T. DeRose. *Toolglass and Magic Lenses: The See-Through Interface.* In Proc. of SIGGRAPH'93, pages 73–80, 1993. xi, 3, 6, 7, 11, 21, 22, 26, 91

[Bierbaum 2001] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker and C. Cruz-Neira. *VR Juggler: A Virtual Platform for Virtual Reality Application Development.* In Proceedings of the Virtual Reality 2001 Conference (VR'01), pages 89–96, 2001. 39, 49

[Boling 2003] D. Boling. Windows CE .NET. Microsoft Press, 2003. 33

[Bowman 1997a] D. Bowman and L. Hodges. *An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments.* In Proc. of the 1997 ACM Symposium on Interactive 3D Graphics (I3D'97), pages 35–38, 1997. 53, 66

[Bowman 1997b] D. Bowman, D. Koller and L. Hodges. *Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques.* In Proc. of the 1997 IEEE Virtual Reality Annual International Symposium (VRAIS'97), pages 45–52, 1997. 6, 18, 20, 75, 92

[Bowman 1999a] D. Bowman, D. Johnson and L. Hodges. *Testbed Evaluation of VE Interaction Techniques.* In Proc. of the 1999 ACM Symposium on Virtual Reality Software and Technology (VRST'99), pages 26–33, 1999. xi, 3, 6, 18, 19

[Bowman 1999b] D.A. Bowman and L. Houges. *Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments.* Journal of Visual Languages and Computing, vol. 10, pages 37–53, 1999. xii, 64, 65, 66

[Bowman 2001] D. Bowman, E. Kruijff, J. J. LaViola Jr. and I. Poupyrev. *An Introduction to 3D User Interface Design.* Presence, vol. 10, no. 1, pages 96–108, 2001. 2

[Bowman 2002] D. Bowman, J.L. Gabbard and D. Hix. *A Survey of Usability Evaluation in Virtual Environments: Classification and Comparison of Methods.* Presence: Teleoperation. Virtual Environments, vol. 11, no. 4, pages 404–424, 2002. 63, 64

[Bowman 2005] D.A. Bowman, E. Kruijff, J.J. LaViola Jr. and I. Poupyrev. 3D User Interfaces. Theory and Practice. Addison-Wesley, 2005. 2

[Bowman 2006] D.A. Bowman, J. Chen, C.A. Wingrave, J.F. Lucas, A. Ray, N.F. Polys, Q. Li, Y. Haciahmetoglu, J. Kim, S. Kim, R. Boehringer and T. Ni. *New Directions in 3D User Interfaces.* International Journal of Virtual Reality, vol. 5, no. 2, pages 3–14, 2006. 2

[Bowman 2008] D.A. Bowman, S. Coquillart, B. Froehlich, M. Hirose, Y. Kitamura, K. Kiyokawa and W. Stuerzlinger. *3D User Interfaces: New Directions and Perspectives.* IEEE Computer Graphics Applications, vol. 28, no. 6, pages 20–36, 2008. 2

[Brooks 1999] F.P. Brooks. *What's Real About Virtual Reality?* IEEE Computer Graphics Applications, vol. 19, no. 6, pages 16–27, 1999. 10, 25, 34

[Brown 2006] L.D. Brown and H. Hua. *Magic Lenses for Augmented Virtual Environments.* IEEE Computer Graphics Applications, vol. 26, no. 4, pages 64–73, 2006. 6

[Brutdzman 2007] D. Brutdzman and L. Daly. X3D: Extensible 3d graphics for web authors. Morgan Kaufmann, 2007. 10

[Burdea 2003] G.C. Burdea and P. Coiffet. Virtual reality technology. Wiley-Interscience, 2nd édition, 2003. 1

[Carey 1997] R. Carey and G. Bell. The annotated VRML 2.0 reference manual. Addison-Wesley Professional, 1st édition, 1997. 10

[Chen 1988] M. Chen, S.J. Mountford and A. Sellen. *A Study in Interactive 3-D Rotation using 2-D Control Devices.* In Proc. of the 15th annual conference on Computer graphics and interactive techniques (SIGGRAPH '88), pages 121–129, 1988. 60

[Chen 2001] H. Chen, Y. Chen, A. Finkelstein, T. Funkhouser, K. Li, Z. Liu, R. Samanta and G. Wallace. *Data distribution strategies for high-resolution displays.* Computers Graphics, vol. 25, no. 5, pages 811–818, October 2001. 6, 29, 31

[Cruz-Neira 1993] C. Cruz-Neira, D.J. Sandin and T. A. DeFanti. *Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE.* In Proc. of the SIGGRAPH '93 Computer Graphics Conference, pages 135–142, 1993. 11, 13

[FFmpeg 2009] FFmpeg. *FFmpeg.* http://ffmpeg.org, November 2009. 35

[Fisher 1987] S.S. Fisher, M. McGreevy, J. Humphries and W. Robinett. *Virtual environment display system.* In Proc. of the 1986 workshop on Interactive 3D graphics (SI3D '86), pages 77–87, 1987. 1

[Fnorb 2009] Fnorb. *Fnorb The pure Python CORBA ORB.* http://fnorb.sourceforge.net, November 2009. 33

[Furness 1986] T.A. Furness. *The Super Cockpit and its Human Factors Challenges.* In Proc. of the Human Factors Society, pages 48–52, 1986. 1

[Gabbard 1999] J.L. Gabbard, D. Hix and J.E. Swan. *User-Centered Design and Evaluation of Virtual Environments.* IEEE Computer Graphics and Applications, vol. 19, no. 6, pages 51–59, 1999. 64

[Gamma 1995] E. Gamma, R. Helm, R. Johnson and J.M. Vlissides. Design patterns. elements of reusable object-oriented software. Addison-Wesley Professional, 1995. 55

[Gray 2003] K. Gray. Microsoft DirectX 9 programmable graphics pipeline. Microsoft Press, 2003. 12

[Hartling 2005] P. Hartling and C. Cruz-Neira. *Tweek: A Framework for Cross-Display Graphical User Interfaces.* In ICCSA (3), pages 1070–1079, 2005. 7, 8, 11

[Henning 1999] M. Henning and S. Vinoski. Advanced CORBA programming with c++. Addison-Wesley Professional, 1999. 33

[Henrysson 2005] A. Henrysson, M. Billinghurst and M. Ollila. *Face to Face Collaborative AR on Mobile Phones.* In Proc. of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR '05), pages 80–89, 2005. 28

[Hoffman 1998] H. Hoffman, A. Hollander, K. Schroder, S. Rousseau and T. Furness. *Physically touching and tasting virtual objects enhances the realism of virtual experiences.* Virtual Reality, vol. 3, no. 4, pages 226–234, December 1998. 23

[Humphreys 2002] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P.D. Kirchner, J. Klosowski and T. James. *Chromium: a stream-processing framework for interactive rendering on clusters.* ACM, Transaction on Graphics, pages 693–702, 2002. 30

[IJEP 2009] IJEP. *Independent JPEG Group.* http://www.ijg.org, November 2009. 35

[JPEG 2009] JPEG. *JPEG.* http://www.jpeg.org, November 2009. 34

[Kelso 2002] J. Kelso and L.E. Arsenault. *DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments.* In IEEE Virtual Reality (VR'2002), pages 183–190, 2002. 39, 49

[Kessler 2000] G.D. Kessler, D.A. Bowman and L.F. Hodges. *The Simple Virtual Environment Library: An Extensible Framework for Building VE Applications.* In Presence: Teleoperators and Virtual Environments, pages 187–208, 2000. 39, 49

[Kruger 1994] W. Kruger and B. Frohlich. *The Responsive Workbench.* IEEE Computer Graphics and Applications, vol. 14, no. 3, pages 12–15, 1994. 14

[Kruijff 2003] E. Kruijff, S. Conrad and A. Mueller. *Flow of Action in Mixed Interaction Modalities.* In Proc. of HCI International 2003, pages 706–715, 2003. 7, 8, 11

[Kukimoto 2005] N. Kukimoto, J. Nonaka, Y. Ebara and K. Koyamada. *Scientific Visualization in Collaborative Virtual Environment with PDA-Based Control and 3D Annotation.* JSME International Journal Series B Fluids and Thermal Engineering, vol. 48, no. 2, pages 252–258, 2005. 7, 11

[Lamberti 2003] F. Lamberti, C. Zunino, A. Sanna, A. Fiume and M. Maniezzo. *An Accelerated Remote Graphics Architecture for PDAs.* In Proc. of the eighth international conference on 3D Web technology (Web3D'03), pages 55–61, 2003. 6, 35, 45

[Lamberti 2007] F. Lamberti and A. Sanna. *A Streaming-Based Solution for Remote Visualization of 3D Graphics on Mobile Devices.* IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 2, pages 247–260, 2007. 35, 45

[Lawton 2006] G. Lawton. *Making Virtual Reality more Accessible.* Computer, pages 12–15, June 2006. 1

[Looser 2004] J. Looser, M. Billinghurst and A. Cockburn. *Through the looking glass: the use of lenses as an interface tool for Augmented Reality interfaces.* In Proc. of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia (GRAPHITE '04), pages 204–211, 2004. 6

[Martz 2007] P. Martz. OpenSceneGraph quick start guide. pmartz, 2007. 39, 55

[McCarty 1994] W.D. McCarty, S. Sheasby, P. Amburn, M.R. Stytz and C. Switzer. *A Virtual Cockpit for a Distributed Interactive Simulation.* IEEE Computer Graphics and Applications, vol. 14, no. 1, pages 49–54, 1994. 1

[Mechdyne 2009] Mechdyne. *CAVELib.* http://www.mechdyne.com/integratedSolutions/software/products/CAVELib/CAVELib.htm1, November 2009. 39, 49

[Nintendo 2009] Nintendo. *Wii.* http://www.nintendo.com/wii, November 2009. 1

[Nobuyuki 2005] K. Nobuyuki, N. Jorji, E. Yasuo and K. Koji. *Scientific Visualization in Collaborative Virtual Environment with PDA-Based Control and 3D Annotation(<Special Issue>Advanced Fluid Information).* JSME international journal. Ser. B, Fluids and thermal engineering, vol. 48, no. 2, pages 252–258, 2005. 28

[OpenGLES 2009] OpenGLES. *OpenGL ES.* http://www.khronos.org/opengles, November 2009. 30

[Perlin 1985] K. Perlin. *An Image Synthesizer.* SIGGRAPH Computer Graphics, vol. 19, no. 3, pages 287–296, 1985. 69, 76

[Pierce 1997] J. Pierce, M. Forsberg, M. Conway, S. Hong, R. Zeleznik and M. Mine. *Image Plane Interaction Techniques in 3D Immersive Environments.* In Proc. of the 1997 ACM Symposium on Interactive 3D Graphics (I3D'97), pages 39–44, 1997. xi, 3, 7, 18, 19, 92

[Poupyrev 1996] I. Poupyrev, M. Billinghurst, S. Weghorst and T. Ichikawa. *The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR.* In Proc. of the 1996 ACM Symposium on User Interface Software and Technology (UIST'96), pages 79–80, 1996. xi, 6, 18, 75

[Poupyrev 1998a] I. Poupyrev, N. Tomokazu and S. Weghorst. *Virtual Notepad: Handwriting in Immersive VR.* In Proc. of the Virtual Reality Annual International Symposium, pages 126–132, 1998. 27, 54

[Poupyrev 1998b] I. Poupyrev, S. Weghorst, M. Billinghurst and T. Ichikawa. *Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques.* vol. 17, no. 3, pages 41–52, 1998. 2, 3, 6, 17, 18

[Qt 2009] Qt. *Qt.* http://qt.nokia.com, November 2009. 46

[Ray 2007] A. Ray and D.A. Bowman. *Towards a system for reusable 3D interaction techniques.* In VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology, pages 187–190, 2007. 50

[Reiners 2002] D. Reiners. *OpenSG: A Scene Graph System for Flexible and Efficient Realtime Rendering for Virtual and Augmented Reality Applications.* PhD thesis, Vom Fachbereich Informatik der Technischen Universitat Darmstadt, 2002. 39, 55, 57

[Reitmayr 2005] G. Reitmayr and D. Schmalstieg. *OpenTracker: A flexible software design for three-dimensional interaction.* Virtual Reality, vol. 9, no. 1, 2005. 39

[Schmalstieg 1999] D. Schmalstieg, M. Encarnaçäo and Z. Szalavari. *Using Transparent Props for Interaction with the Virtual Table.* In Proc. of the 1999 ACM Symposium on Interactive 3D Graphics (I3D'99), pages 147–154, 1999. xi, 3, 6, 7, 23, 91

[Schmalstieg 2002] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L.M. Encarnaçäo, M. Gervautz and W. Purgathofer. *The Studierstube Augmented Reality Project.* Presence: Teleoperation and Virtual Environments, vol. 11, no. 1, pages 33–54, 2002. 1, 7, 11

[Shibano 2001] N. Shibano, P.V. Hareesh, M. Kashiwagi, K. Sawada and H. Takemura. *Development of VR Experiencing System with Hemi-Spherical Immersive Projection Display for Urban Environment Design.* In Proc. of the Seventh International Conference on Virtual Systems and Multimedia (VSMM '01), pages 499–505, 2001. 14

[Shreiner 2004] D. Shreiner, M. Woo, J. Neider and T. Davis. OpenGL programming guide. Addison Wesley, 4th édition, 2004. 12, 30, 39, 55

[Stevens 1998] W.R. Stevens. UNIX network programming, volume 2: Interprocess communications. Prentice Hall, 2nd édition, 1998. 33

[Stoakley 1995] R. Stoakley, M. Conway and R. Paush. *Virtual Reality on a WIM: Interactive Worlds in Miniature.* In Proc. of the 1995 ACM Conference on Human Factors in Computing Systems (CHI'95), pages 265–272, 1995. xi, 17

[Stoev 2002] S.L. Stoev and D. Schmalstieg. *Application and Taxonomy of Through-the-lens Techniques.* In Proc. of the ACM symposium on Virtual reality software and technology (VRST'02), pages 57–64, 2002. 6

[Sutherland 1965] I. Sutherland. *The Ultimate Display.* In Proc. of IFIP Congress 65, pages 506–508, 1965. 10

[Sutherland 1968] I. Sutherland. *A Head-Mounted Three Dimensional Display.* In Proc. of the December 9-11, 1968, fall joint computer conference, part I. AFIPS '68 (Fall, part I), pages 757–764, 1968. 11

[Szalavári 1997] Z. Szalavári and M. Gervautz. *The Personal Interaction Panel - a Two-Handed Interface for Augmented Reality.* Computer Graphics Forum, vol. 16, no. 3, 1997. 6, 7, 23

[Taylor 2001] R.M. Taylor, T.C Hudson, A. Seeger, H. Weber, J. Juliano and A.T. Helser. *VRPN: a device-independent, network-transparent VR peripheral system.* In Proc. of the ACM symposium on Virtual reality software and technology (VRST '01), pages 55–61, 2001. 38

[Viega 1996] J.C. Viega, M.J. Conway, G. Williams and R. Pausch. *3D Magic Lenses.* In Proc. of the 9th annual ACM symposium on User interface software and technology (UIST '96), pages 51–58, 1996. xi, 11, 21, 22, 26, 91

[Watsen 1999] K. Watsen, R.P. Darken and M.V. Capps. *A Handheld Computer as an Interaction Device to a Virtual Environment.* In Proc. of the Third Immersive Projection Technology Workshop, 1999. 7, 11

[Welch 2001] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller and D. Colucci. *High-Performance Wide-Area Optical Tracking: The HiBall Tracking System.* Presence: Teleoperators Virtual Environments, vol. 10, no. 1, pages 1–21, 2001. 37

[Wingrave 2008] C.A. Wingrave and D.A. Bowman. *Tiered Developer-Centric Representations for 3D Interfaces: Concept-Oriented Design in Chasm.* In IEEE Virtual Reality Conference 2008 (VR 2008), pages 193–200, 2008. 50

List of Publications

Journals Papers

1. Miranda Miranda Miguel, Kiyokawa Kiyoshi and Takemura Haruo. *Implementation and Evaluation of a Magic lens Interface Using a Handheld Device in an Immersive Virtual Environment.* The Journal of the Institute of Image Information and Television Engineers. pp. 816-821. Vol. 63, No. 6, 2009.

International Conference Papers

3. Miranda Miranda Miguel, Kiyokawa Kiyoshi and Takemura Haruo. *A PDA-based See-through Interface within an Immersive Environment.* In Proc. Int. Conf. on Artificial Reality and Telexistence (ICAT). pp. 113-118,.Esbjerg Denmark, November 2007.
4. Miranda Miranda Miguel, Kiyokaya Kiyoshi, Takemura Haruo. *Interaction within immersive virtual environments through a See-Through interface implemented on a PDA.* In Proc. of the Int. Workshop on Ubiquitous Virtual Reality 2008. Session 8, pp 1-14, Jan.2008 Osaka Japan. January 2008.

Domestic Conference Papers

5. Miranda Miranda Miguel, Ogawa Takefumi, Kiyokawa Kiyoshi and Takemura Haruo. *A PDA-based See-through Interface for Extending Interaction Functionality in a CAVE Display.* In Proc. of Institute of Electronics, Information and Communication Engineers, pp. A-16-24, Nagoya Japan. March 2007.
6. Miranda Miranda Miguel, Kiyokawa Kiyoshi and Takemura Haruo. *A PDA-based See-through Interface within an Immersive Environment.* In Proc. of Institute of Electronics, Information and Communication Engineers, Vol.107, No.80 pp. 37-42. Tokyo Japan, June 2007.
7. Miranda Miranda Miguel, Kiyokawa Kiyoshi and Takemura Haruo. *A PDA-based See-through User Interface within an Immersive Environment.* In Proc. of the Virtual Reality Society of Japan, pp. 2A2-6. Fukuoka Japan, September 2007.

8. Kiyokawa Kiyoshi, Miranda Miguel, Nozaki Kazunori, Yasufuku Kensaku, Itoh Kazuo, Iwata Yasunori. *HOPE - Development of a High-definition Immersive Projection Display.* In Proc. of the Virtual Reality Society of Japan, Fukuoka Japan, September 2007.
9. Miranda Miranda Miguel, Kiyokawa Kiyoshi and Takemura Haruo. *A PDA-based See-through Interface within an Immersive Environment.* In Proc. Int. Conf. on Artificial Reality and Telexistence (ICAT), pp. 113-118,.Esbjerg Denmark, November 2007.
10. Miranda Miranda Miguel, Kiyokaya Kiyoshi, Takemura Haruo. *Interaction within immersive virtual environments through a See-Through interface implemented on a PDA.* In Proc. of the Int. Workshop on Ubiquitous Virtual Reality 2008. Session 8, pp 1-14, Jan.2008 Osaka Japan. January 2008.
11. Miranda Miranda Miguel, Kiyokaya Kiyoshi, Takemura Haruo. *Implementing a See-through the lens interface for enhancing interaction in a CAVE.* In Proc of Human Interface Symposium 2008. Osaka Japan. September 2008.
12. Miranda Miranda Miguel, Kiyokawa Kiyoshi, Takemura Haruo. *Implementation and Evaluation of a See-through the lens interface in an Immersive Virtual Environment.* In Proc. of Institute of Electronics, Information and Communication Engineers, Vol. 108, No. 226 pp.27-32.-A Kushiro Japan. October 2008.

Demonstrations

13. Supercomputing 2008. Austin Texas. November 15-20, 2008.