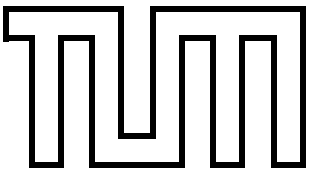


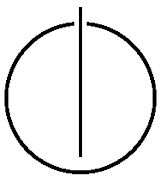
FAKULTÄT FÜR INFORMATIK

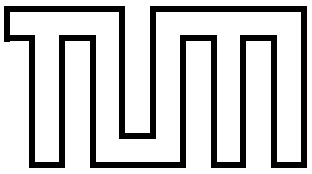
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

# 3D Magic Lens Implementation using a Handheld Device in a 3D Virtual Environment

Alba Huelves





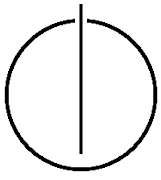
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Masterarbeit in Informatik

# 3D Magic Lens Implementation using a Handheld Device in a 3D Virtual Environment


| | |
|---|---|
| Author: | Alba Huelves |
| Supervisor: | Prof. Gudrun Klinker |
| Advisor: | Amal Benzina, M.sC. and Dr. Marcus Tönnis |
| Date: | August 6, 2012 |

I assure the single handed composition of this bachelor thesis only suported by declared resources.

München, den 13. September 2012 Alba Huelves

## Acknowledgments

I would like thank Prof. Gudrun Klinker for giving me the opportunity to get involved in this fascinating and interesting project.
I would also like to thank my advisors Amal Benzina, M.sC. and Dr. Marcus Tönnis for their support, patience and guidance, their help was essential through all the thesis.
Thanks to my friends who were always encouraging and supportive and gave me the energy that I needed.
And thanks to my parents for their unconditional support and comprehension and for giving me the chance to achieve my ambitions.

# Abstract

When dealing with large data sets in Virtual Environments it is important to be able to manage the viewing of the data to obtain different levels of detail and to explore it from different perspectives. For this the appropriate interaction techniques have to be developed.
This work consists on the investigation of different approaches to implement a flat Magic Lens application which interacts in a 3D Virtual Environment. The Magic Lens represents an interaction metaphor for the 2D selection and manipulation of 3D graphical information. The user by changing the position and the rotation of a hand held device, controls the Magic Lens motion in the Virtual Environment. Therefore two alternative views of the scene are offered, one where the user can appreciate an overview of the data set and the other which shows the focus view. Once some region of the VE is selected by the Magic Lens it is necessary to compute the view frustum, so that we can obtain a snapshot of the desired frustum in the hand held device and manipulate it there. This thesis focuses on the different interaction techniques which can be used in order to explore the virtual world with the Magic Lens.

# Contents

# List of Figures

# Outline of the Thesis

CHAPTER 1: INTRODUCTION

This chapter discusses the importance of interaction techniques in VE and introduces the Magic Lens as one of them. The main purpose is presented and the different approaches are briefly described.

CHAPTER 2: RELATED WORKS

This chapter gives an overview of the previous works and investigations in the Magic Lens research area and discusses the evolution of the Magic Lens concept over the years.

CHAPTER 3: ENVIRONMENT AND SYSTEM ARQUITECTURE

In this chapter the different system elements that take part in the experimental setup are presented. The system architecture required for the interaction between the different elements is also explained in detail.

CHAPTER 4: EXPLORING THE VIRTUAL WORLD USING THE ANDROID TABLET

This chapter presents the different metaphors that were investigated throughout the thesis and explains the different aspects of their implementation.

CHAPTER 5: IMPLEMENTATION

In this chapter some the most relevant aspects of the implementation are presented in more detail.

CHAPTER 6: CONCLUSION

This chapter discusses the different metaphors analysing their advantages and disadvantages and it also discusses the further work that could be done to improve the application and offer additional functionality.

# 1 Introduction

Three dimensional viewing has become recently more and more important since every day more applications are being developed with the help of 3D graphics and so they can be displayed in three dimensions. This adds extra functionality to the current applications and therefore great benefits can be obtained from them. Many applications in different areas and fields like medicine or even oil and gas exploration could benefit from this.

It is not only important to develop mechanisms for three dimensional viewing but it is also necessary to develop appropriate human interaction techniques with the three dimensional environments so that the user can travel through the three dimensional space, select desired regions of it and even manipulate them. Here is where immersive Virtual Environments (VEs) gain importance, because they allow the user to virtually explore the environment without having to take care of physical obstacles or obstructions.

The interaction technique that will be investigated in this work is the so called Magic Lens. This term was first introduced by Bier et al and it was defined as a transparent sheet of glass between an application and a cursor so filters could be applied to modify the presentation of objects or reveal hidden information. For this thesis a flat Magic Lens will be implemented for interaction with a 3D VE, so it can be described as the visual feedback that will be rendered in a 3D terrain which is controlled by means of a hand held device.
Hand held devices such as smartphones and tablets are growing more and more in popularity over the recent years and it is expected that their markets will continue to grow in the next years so the use of a hand held device such as a tablet is a suitable election because they offer many functionalities and advantages.

The Magic Lens represents an interaction metaphor for the 2D selection and manipulation of 3D graphical information so it can offer an alternative view of the virtual world to the user. The user by changing the position and the rotation of the hand held device, controls the Magic Lens visualization in the virtual Environment. For this purpose the hand held device is tracked and therefore its pose can be obtained and updated every time the device is moved. This is achieved by means of an ART which uses several infra-red cameras to obtain very ac-

curate data.

As mentioned before the Magic Lens offers an alternative view of the virtual world to the user, because the user can explore the 3D virtual world with it, to focus on a certain region, and then take a snapshot of it. However since the 3D virtual world consists of a 3D Terrain it is also practical to be able to visualize a certain area of the surface of the Terrain from a perpendicular perspective so that there is the possibility to explore below the surface. The below surface exploration is not covered in this work, since the data available is of the surface of a terrain but it is possible to determine the the intersection with the terrain surface, knowing what area of it is being selected, so that in the future if this data is available, it can be revealed what is underneath the selected area for further exploration.

## 1.1 Purpose

The main purpose is to investigate different interaction techniques for selection and manipulation of 3D graphical information and to offer some visual feedback to the user by means of the Magic Lens so it is intuitive enough to obtain an image of a desired region or to visualize a determined area of the surface of the 3D Terrain, so further on below surface exploration can occur. The interaction will be achieved with a wireless connected Android tablet.

## 1.2 Approach

For this work a Samsung Galaxy Tablet PC was used as a user interface to perform the different tasks of selection and navigation with the Magic Lens. Through the touch screen and the appropriate application the user is able to control a two dimensional Magic Lens in the form of a rectangle which is rendered in the virtual world. Initially the user can travel through the 3D Terrain to obtain the desired contextual view and by entering the Magic Lens Mode, the user can start the movement of the Magic Lens.

The tablet PC is tracked and therefore it can be used as a gesture input to control the motion of the Magic Lens in the virtual world. For this purpose two different approaches are considered: rate control and direct avatar. For the rate control option the rectangle is initially rendered in a fixed position within the chosen view and when the user applies a certain rotation or translation to the hand held device the rectangle will move accordingly increasing it's speed proportionally.

For the direct avatar option the user will get the impression that the Magic Lens behaves in

the virtual world as the hand held device in the real world so that it imitates its gestures and follows the tablet. This is achieved by position control.

In order to visualize what would be seen through the lens it must be considered from which point of view is the user looking through. Therefore two different approaches are investigated concerning the viewpoint position. Firstly the viewpoint of the user can be considered as fixed to the Magic Lens as if the user's eyes were always looking perpendicular to the surface of the Magic Lens. Some visual feedback will be shown to let the user know where the viewpoint is while the rectangle moves in the virtual world. Also there is the option to make the virtual camera follow from a certain distance the Magic Lens so it will always be shown from behind the viewpoint.

The second option involves head tracking, so the viewpoint is set to the user's eye and its position is known relative to that of the tablet. This can then be mapped in the virtual world and it would make a difference on what would be visualized through the lens when the position of the eye changes relative to the tablet.

The user can select the different options through the Android application which will be running in the tablet PC and will be connected through wireless LAN to two different machines. One of machines will be the FRAVE where the contextual view will be displayed and the other would show the focus view, this being, what would be seen through the lens in order to then send it to the tablet PC.

The Virtual Environment that will be used will be the Terrain3D which provides the rendering of a high resolution landscape of Utah for immersive interaction.

# 2 Related Works

## 2.1 Magic Lens Concept

The Magic Lens metaphor is used as an interaction technique for exploring the virtual world and it is useful to apply different effects to a certain region of the virtual environment or even just for selection purposes in large datasets.

The Magic Lens concept was first introduced by Bier et al [1]. It was introduced as a transparent filter that could alter the presentation of application objects by applying different effects in order to be able to reveal hidden qualities of the objects or even enhance them or remove them. So this filter was integrated as one of various tools which could interact in a graphical user interface defined as *the see-through interface*.

However these Magic Lenses were designed for two dimensional interfaces and they were managed by means of a mouse or by rolling a trackball.

This concept has afterwards been developed and extended and many variations have appeared through the years, so an evolution of the Magic Lens concept will be in this section discussed.

### 2.1.1 3D Magic Lenses

The Magic Lens concept was extended to three dimensional environments by Viega et al [3]. Two different modalities were presented: a two dimensional or flat lens in a 3D environment and a 3D or volumetric lens. The flat lens was implemented following the Magic Lens metaphor for 2D interfaces so that only the portion of the application covered by the lens was affected and not the whole object. In the same way, a flat lens in 3D environment will only affect the area of the scene which is visible through the lens and not the entire 3D object, offering an alternative view.

A new concept was introduced for this purpose: the *lens frustum*. It was defined as the six sided volume resulting from casting four rays from the viewpoint across the corners of the

lens and then marked out by the lens itself at the near end and by the chosen far plane, as it can be seen in figure 2.1. It is inside this volume where the different effects are applied.

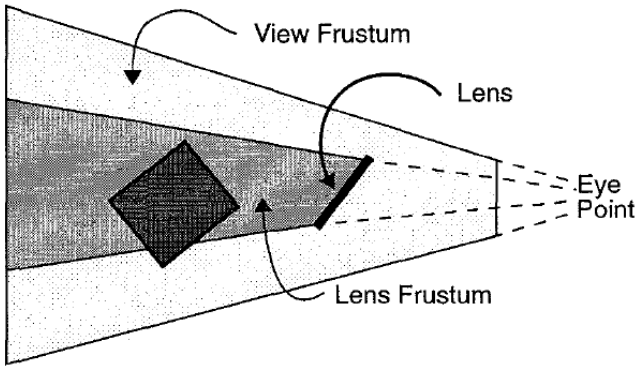



Figure 2.1: Lens frustum [3]

In this work a volumetric Magic Lens was also implemented so in this case the lens frustum is a rigid volume that doesn't change it's shape even if the orientation of the lens varies and it doesn't depend on the user's viewpoint. Some of the uses they gave to these volumetric lenses are culling large data sets, so that only what is inside the volume is rendered, and X ray vision, so that when the user moves the volumetric lens into an object, it reveals the inside of the object.

Based on the previous works several papers were issued covering the taxonomy and composition of Magic Lenses and some of them developed applications of the concept. The paper of Stoev et al [4] used the *through-the-lens* metaphor to develop an application for exploring the virtual world, navigate through it and even manipulate objects remotely. For the exploration aspects they made used two different worlds, the *primary world* and the *secondary world*. In each of these worlds a different viewpoint was used, so the primary world represented the world that surrounds the user and it can be explored by a see-through window. On the other hand the secondary world offers an alternative point of view, where only what is seen through the window in the primary world is visualized. This techniques allows therefore to explore locations of interest while maintaining the same location in the primary world.

### 2.1.2 Magic lenses with arbitrary convex shapes

Further work has then been done focusing on the different shapes that volumetric lenses could acquire [5]. This way the user can interactively change the region of interest by changing the shape of the lens. For this purpose a hardware accelerated algorithm was presented

which revealed some advantages to that of Viega et al based on clipping planes. Two different lenses where presented, the so called *camera lens* which is fixed relative to the camera position and the *scene lens* which can be placed anywhere in the virtual scene. The first modality can be used to explore dense data sets, so that the data near to the camera is displayed differently. The second modality can be used as an erase lens where occluded parts of the scene can be revealed or as a texture lens, applying different textures to the objects in the scene.

### 2.1.3 Interactive Worlds in Miniature: WIM

Stoakley et al [2] introduced the *Worlds in Miniature* metaphor which consists on having a miniature copy of the virtual environment to enhance many different interaction techniques such as selection and navigation, and to provide an interface to manipulate objects in the virtual world.

This miniature replica of the virtual world consists of a model that bears a direct relationship with the objects in the original virtual world and it can be manipulated by the user through a hand held device. When the user performs an action on an object of the miniature world the identical action affects the object in the original virtual environment.

One of the advantages described for this metaphor is the fact that through the WIM an alternative point of view can be provided to the user. Whenever the user feels the need to change the point of view, it can be done by rotating the hand held device, while the point of view of the original virtual world remains intact. This may allow the user to visualize part of the scene which might have been hidden from another viewpoint.

Based on this metaphor followed the works of Brown and Hua [6] who by using a previously developed augmented virtual environment implemented an infrastructure for simultaneous viewing of two different perspectives of a virtual world. They used one display to offer a life-sized visualization of the virtual world, while on a different display an alternative view is provided by means of the Worlds in Miniature metaphor. Therefore a different overview is offered in each of the displays at different resolutions and with different amount of details. However they introduced the Magic Lens metaphor because although having two alternative views, they were considered to be insufficient and they considered necessary to introduce a tool which would be able to offer visualization of a point of interest at an intermediate level of detail. The Magic Lens is therefore used as a tool for manipulating the point of interest and a focus view of this point of interest is then obtained. The main functionalities that were explored were presentation, filtering and providing an interface for selection, inspection and manipulation.

For this purpose a hand held Magic Lens was implemented, which could be held over the point of interest in the WIM so that the focus view could be then displayed in the device as if it were enlarged. A second interface was also implemented which enabled the users to place the so called *selection icon* somewhere in the WIM so that in a separate display a focus view could be displayed and manipulated.

### 2.1.4 Magic Lenses and Hand Held Devices

Several other implementations of the Magic Lens metaphor have been achieved with hand held devices. A PDA based interface was used in the works of Miranda et al [7] as a see-through lens to manipulate objects in an immersive environment. The immersive environment used for interaction is a CAVE system and when the user is inside it he carries a tracked PDA which can receive images from the CAVE. In order to perform an action on a desired object the user has just to cover it with the PDA and the resulting image of what is being covered will be displayed in the PDA screen in real time. With the help of the PDA interface the user can then apply changes on the objects that are being displayed on it and these changes will subsequently be applied in the CAVE and hence to the image in the PDA screen. This kind of system can therefore be used for interaction and selection purposes in a 3D virtual world.

A very similar approach was pursued by Sanneblad and Holmquist [8] for interaction with large and high resolution data sets. However their implementation only included two dimensional environments. A large display is used to visualize the large images but its level of detail and its resolution is compromised so by means of a tracked hand held display, certain regions of the image can be examined with higher detail or even additional information can be shown allowing also user interaction by adding or removing features to the displayed image.

## 2.2 Preceding Work

This thesis follows rather closely the previous works of several students that have been working on investigating human computer interaction techniques for navigation and selection in a 3D virtual environment with a hand held device.

Sandro Weber [11] investigated in his bachelor thesis different metaphors to modify the viewpoint from which the virtual world is visualized. To achieve this a hand held tablet PC is used to input hand gestures. Depending on the gesture and the different modes selected

through the user interface a different metaphor applies to change the virtual viewpoint, allowing the user to navigate through the virtual world. The main metaphors that he investigated were the car steering wheel and the air plane wings. Depending on the angle at which the device was held one metaphor would have a bigger influence than the other. His work was an extension of Nicolas Heuser's diploma thesis [10].

For this thesis a tracked hand held tablet PC is also used as the input for hand gestures in order to control the motion of the Magic Lens in the virtual world, however different metaphors are used to move the Magic Lens.

In his bachelor thesis Yanco Sabev [12] implements an interaction technique in a 3D virtual environment for selection and zooming purposes. Once again the user interface is a hand held device, in this particular case an Android smartphone. This time the device is not tracked and instead the different sensors of the phone are used to recognise different hand and touch gestures. Through the phone a 3D cursor shaped as a sphere is controlled in the virtual world as if it were a virtual avatar. This cursor is used for selection purposes so it can be moved freely by means of the phone through the virtual world. Initially it is placed at a fixed position in the virtual world and when and depending on the gesture performed in the hand held device the relative movement of the cursor is calculated.

The concept of the virtual avatar is also used in this thesis, since the Magic Lens will be implemented as an avatar of the hand held device to provide visual feedback to the user of the region of interest that is being selected in the virtual world. The movement of the Magic Lens will also be relative to a fixed starting position but this will be further explained in chapter 4.

In Nicolas's Heuser project *"Window into a virtual world" screen concept* [9], he investigates a paradigm where a display is used to offer a view into the virtual world. This view depends on both the displays and the user's eyes position and orientation in the real world, which is then mapped to the virtual world. The metaphor implies that the user is looking through the display as if it were a window, so if the user's head moves horizontally or vertically relative to the display different objects are revealed in the display. The same happens when we alter the position and orientation of the display since a different region is then revealed.

This paradigm is very useful when considering the integration of head tracking to the Magic Lens metaphor. Not many implementations of the Magic Lens take into account the position of the user's eye when exploring the virtual world with the Magic Lens metaphor. However some research has been done that has proven that head tracking improves the user's

performance when interacting with a 3D virtual environment such as the works of McKenna [13]. In his paper a monitor display and the user's head are tracked in order to generate a *viewpoint dependant image* and an experiment is conducted to evaluate the effectiveness of setting the viewpoint to the user's eye. For that they compare the ability of the user to select objects from a 3D space under different conditions: fixed viewpoint, viewpoint can be moved with a mouse and the viewpoint is controlled by the head movement.

In this work the position of the user's eye relative to that of the hand held device will be taken into account in one of the interaction techniques that is implemented and will then be mapped to the virtual world to obtain the lens frustum mentioned in section 2.1.1.

# 3 Environment and System Architecture

For the realisation of this work several devices were required, each of them offering a different functionality to the application. I will present the specific characteristics of the different elements of the environment such as the tablet PC, used as a user interface, the glasses, the tracking system and the two visualization frameworks that provide the user with two alternative contextual views.

It was necessary to design a system architecture which defined how the different elements of the whole system interact with each other. In this chapter the initial architecture will be discussed and finally the issues with the pilot architecture due to synchronization problems will be also explained.

## 3.1 System Elements

### 3.1.1 Android Tablet

The tablet PC used is a Samsung Galaxy Tab which runs Android 2.2 as an Operating System (OS). The device dimensions are 190.09 x 120.45 x 11.98 mm, offering a 7 inch TFT touch screen with 600 x 1024 pixel resolution. It weighs 380g and provides connectivity through Wi-Fi 802.11 b/g/n.

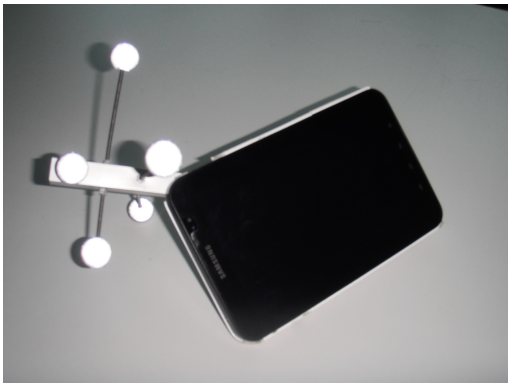
Its features make it light enough to be hand held and at the same time it offers a relatively large screen for displaying images. The fact that it runs an Android OS is a great advantage because the Android SDK provides with the necessary tools and libraries to develop any desired application. By extending the already implemented Graphic User Interface (GUI) in the previous work of Ashry [14] new functionality was added to the Android application in order to perform the tasks required by the Magic Lens system.

The wireless connectivity allows the device to be connected through a WLAN to the other elements of the system and through the screen sensors the user can specify the different operations to perform. This way the application that runs on the Samsung Galaxy Tab acts as a client. The specifics and appearance of the GUI will be further discussed in chapter 4 (section

4.4).

The tablet has various spherical markers attached to its upper left corner (see figure 3.1(b)) which allows the tracking system which will be described in section 3.1.3 to identify it as a target.

(a) Glasses with the A.R.T target for head tracking



(b) Tablet with the A.R.T target

Figure 3.1: Tracked devices used in the environment

### 3.1.2 Glasses

In order to be able to track the user's head a target with markers was attached to 3D glasses as it can be seen in figure 3.1(a). The glasses could then be tracked by the tracking system described below so that the viewpoint could then be set to the user's eye instead of being fixed in front of the Magic Lens when this option is enabled.

### 3.1.3 A.R.T Tracking System

In order for the user to be able to interact with the virtual environment an Advanced Realtime Tracking system (A.R.T)[15] was used for positional measurement of the Android tablet and of the user's head in a defined space by using optical trackers. This system consists of six different infra-red (IR) cameras distributed in the environment. Each of them emits an IR radiation which is then reflected back by the markers attached to the target bodies in the same direction of the incoming light.

For this thesis it was required to obtain six degrees of freedom (6DOF) poses of two different bodies that had several markers attached to them, which consist on measuring position

and orientation of the target simultaneously. In order to be able to determine these poses a calibration process was done previously so that the positions and orientations of the cameras were obtained and also the distribution of the markers within the different targets attached to the tablet and the 3D glasses could be collected. The tablet has five reflecting spherical markers whereas the glasses have six markers attached which form the targets (figures 3.1(b) and 3.1(a)).

The cameras are provided with a synchronization signal by the ARTTRACK Controller (ATC) and they produce periodic emission of IR light illuminating the tracking volume. An image is then taken by the cameras and by processing this image the marker reflections are identified and their positions in image coordinates can then be computed with 2DOF. The ATC is then able to obtain a 6DOF pose of the targets attached to the tablet and the glasses and this information is then transmitted through Ethernet to the application.

This data is required by the application in order to establish a relationship between the real world coordinates of the tracked devices and the virtual coordinates. This system offers high accuracy and a maximum frame rate of 60Hz.

### 3.1.4 FRAVE

The Flexible Reconfigurable Cave (FRAVE) is a virtual reality system which offers real time visualization of 3D environments for immersive interaction (figure 3.2). It is presented as an alternative to the previously established CAVE (Cave Automatic Virtual Environment) which consists of in between three or six walls where projections of the virtual world can be visualized giving the user the feeling of immersion in the environment.
The FRAVE offers more flexibility and modularity since it consists of up to ten screens that can be arranged at different angles. For this work six 3D full HD (1920x1080) plasma Panasonic 65″ screens were used configured so that the side displays formed and angle of 30 degrees with the central display. Each of the three pairs of screens are run by three different workstations and parallel image rendering is enabled by the Equalizer Graphics software.

This system was used for visualization of the terrain data of the state of Utah (USA) with the corresponding Terrain3D software developed by the Chair of Computer Graphics and Visualization, offering a 1m resolution. This software uses C++ as a programming language and integrates multiple libraries to enable calculations and rendering operations to implement the system. In the FRAVE setup the software was run in an Ubuntu 10.10 platform that had access to a WLAN through a wireless adapter.

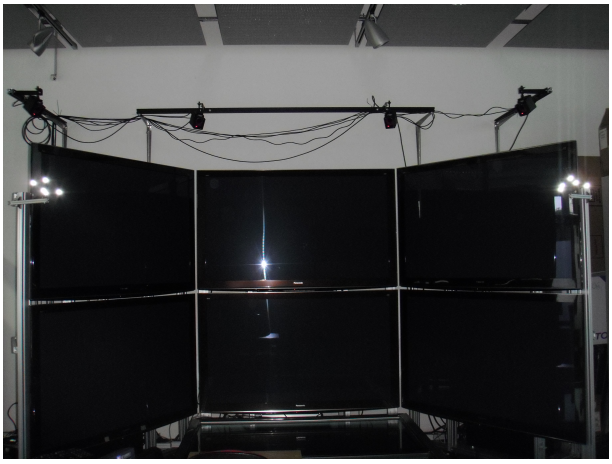


Figure 3.2: Six screens of the Frave setup with four of the six infra-red cameras and the two wall markers

### 3.1.5 Fraveui0

The Fraveui0 is the machine used to offer an alternative view of the terrain data set. The software in this machine runs in a Windows 7 professional platform and it has a Samsung SyncMaster 2442 display of 24 ". The same instance of the Terrain3D software as in the FRAVE setup runs simultaneously in this machine enabling the user to visualize the region of interest selected by the Magic Lens instead of the whole background context.

### 3.1.6 Coordinate Systems

As it has been mentioned we are going to deal with coordinates of the real world and with coordinates of the virtual world, so it is very important to bear in mind that there will be different coordinate systems.

#### Tablet coordinate system

The tablet has its own local coordinate system as shown in figure 3.3 with its origin in the centre of the tablet. To obtain this coordinate system a calibration of the device was done as Heuser explains in his previous work [10] so the offset transformation from the A.R.T target attached to the device to the centre of the tablet can be computed. Weber [11] then did this same calibration for the same tablet used in this work. This coordinate system is defined in the real world.

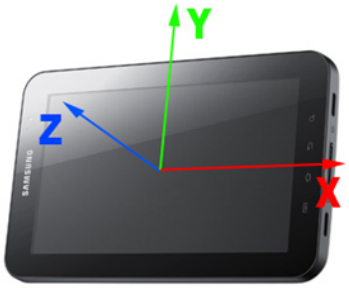



Figure 3.3: Local system coordinate of the tablet [11]

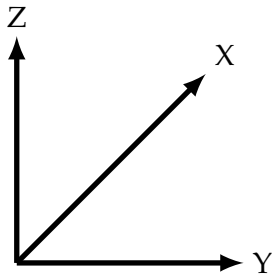



Figure 3.4: Coordinate system of the terrain

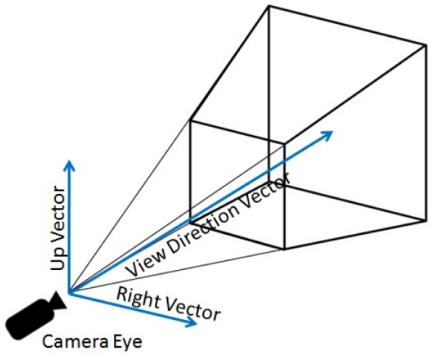



Figure 3.5: Virtual camera coordinate system [12]

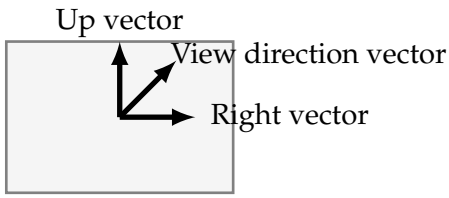



Figure 3.6: Coordinate system of the Magic Lens

**Terrain coordinate system**

As shown in figure 3.4 the Z axis stands for the height of the objects in the terrain, the X axis stands for the depth of the terrain and the Y axis represents the horizontal position in the terrain. This coordinate system has to be taken into account in order to render objects in the terrain at the desired position.

**Virtual camera coordinate system**

The virtual camera represents the viewpoint from which the terrain is visualized. The coordinate system is defined by the up, right and the view direction vectors and the origin is at the camera position. These vectors define the orientation of the camera in the virtual world and the origin defines the position of the camera in the terrain.

The orientation and the position of the virtual camera play an important role on the view frustum, which defines the volume that will be rendered.

#### Magic Lens coordinate system

The Magic Lens has its own local coordinate system in the virtual world (figure 3.6) since it is needed in order to rotate it and change its position. The origin of the coordinate system is at the centre position of the Magic Lens and the vectors are defined in the same way as in the virtual camera coordinate system.

## 3.2 System Architecture

For the implementation of the Magic Lens application the different elements that were described previously need to communicate with each in order to share important data. It was therefore important to design the appropriate communication architecture that defines what data is needed and how the data is transferred. The final architecture that was adopted will be here described and finally the issues that aroused with the initial architecture that made it necessary to include modifications will be explained. A simplified scheme of what the architecture would look like is shown in figure 3.7.

It is important to point out that the same instance of the Terrain3D software runs in the FRAVE and in Fraveui0 but it will behave differently depending on the machine it runs on. To identify the machine a flag was set in the configuration files of the software, the same configuration files also had information about the different IP addresses of the FRAVE, the Fraveui0 machine and the Android tablet device, as well as the port used for the different connections.

The main elements that form the system are the following:

1. **Android Client running on the tablet:** the Android application that runs on the tracked tablet is used as the user interface, and it provides the necessary functionality so that the user can select the different preferences, settings and modes. It is in charge of sending requests to the Android Servers running on the other machines.

2. **FRAVE:** an Android Server runs on this setup in order to receive the requests sent from the Android Client and it activates the corresponding flags so that the application behaves accordingly to what the user specifies. This server also communicates with another Android Server running on the Fraveui0 machine and it sends information concerning the Magic Lens's position and orientation in the virtual world. It is also nec-

essary that the A.R.T system sends to this entity the poses of the tracked tablet and glasses periodically. All this information is needed to render the virtual avatar of the Magic Lens within the terrain.

3. **Fraveui0:** this machine runs the other Android Server, which receives the same requests as the one in the FRAVE, from the Android Client. However this entity also needs to send data to the Android Client whenever a snapshot request is specified by the user. As already mentioned, this Android Server receives data from the server running on the FRAVE, so that it is able to compute the *lens frustum* and therefore present the alternative view.

4. **A.R.T system:** the tracking system provides the necessary tracking data via Ethernet for the FRAVE software to use it, updating it every time there is a change.

### 3.2.1 Network Connections

Several network connections are needed for the correct performance of the architecture and some choices had to be made on what was the appropriate protocol for each of the connections. The different choices and the need for the connection will be explained in more detail next.

#### Connection between Android Client and Android Servers in FRAVE and Fraveui0

In the previous works of Sandro Weber [11] and Yanco Sabev [12] the Android hand held device was connected to the Android Server running on the FRAVE setup through a wireless connection and the transport protocol used for this connection was UDP. This protocol is not connection based and it offers no data flow control, error control or retransmissions and it doesn't guarantee that packets arrive in order. However for these reasons and for using smaller headers than TCP it is suitable for fast transmissions of data and therefore for real time applications.

Since the requests from the Android Client to the servers are sent in the form of *string token* messages and the order of the messages is not critical for the application the same networking conditions are maintained for this connection. The main change that was introduced relies on the fact that now the Android Client needs to open two different sockets, each one to send data to a different IP address, although it can be send to the same port in each machine.

These connections are necessary because the instance running on both FRAVE and Fraveui0 need to receive information about when the Android application is started by the user, when

he wants to select the Magic Lens functionality, when he wants to move the Magic Lens and when he wants to take a snapshot of what would be seen through the lens.

#### Connection between both Android Servers in FRAVE and Fraveui0

This is a one direction connection from the Android Server in the FRAVE to the Android Server in the Fraveui0 machine. The data that is sent through this connection is:

- the coordinates of the virtual viewpoint through which the *lens frustum* is visualized
- the orientation of the Magic Lens (heading, roll and pitch)
- the coordinates of each of the four corners that constitute the Magic Lens.

Since a considerable amount of data has to be transferred in real time the UDP protocol was also used for this transmission given that it is faster than TCP and the data has to be sent almost continuously. Whenever the Magic Lens is moving this data has to be sent, since its position and orientation has to be updated in the virtual world and therefore what is shown in Fraveui0 also needs to be updated to change the *lens frustum* view.

#### Connection between Android Server in Fraveui0 and Android Client

This connection is needed to send the snapshot captured in the Fraveui0 machine of the *lens frustum* to the Android Client in the tablet so it can be displayed in the device through the Android application. The data to send is an image which has a considerable size and therefore it had to be sent in different fragments so the best transport protocol for this objective was TCP because it guarantees that the data is delivered reliably and in order to the destination.

Whenever a snapshot request is sent from the Android Client running on the tablet to Fraveui0 a TCP receiver in the Android application starts listening for a connection from the Android Server in the Fraveui0 and when the connection is established the image data is sent as a stream of octets and then displayed in the tablet.

#### Connection between A.R.T system and FRAVE

The Terrain3D running on the FRAVE needs to be aware of all the changes in the position and orientation of the Android tablet target and of the glasses target so that whenever tracking is enabled by the user this information can be retrieved. Unlike the other connections which were through a Wireless LAN this data is transferred through the Ethernet by the tracking system. To configure that the poses were sent to the FRAVE machine the TrackD interface

provided by Mechdyne [16] was used were the IP of the FRAVE machine was provided.

In order to process the tracking data and to obtain it, *Ubitrack*[1] libraries using Heuser's integration of the libraries in the Terrain3D [10] were employed, used also in the following works of Weber [11].

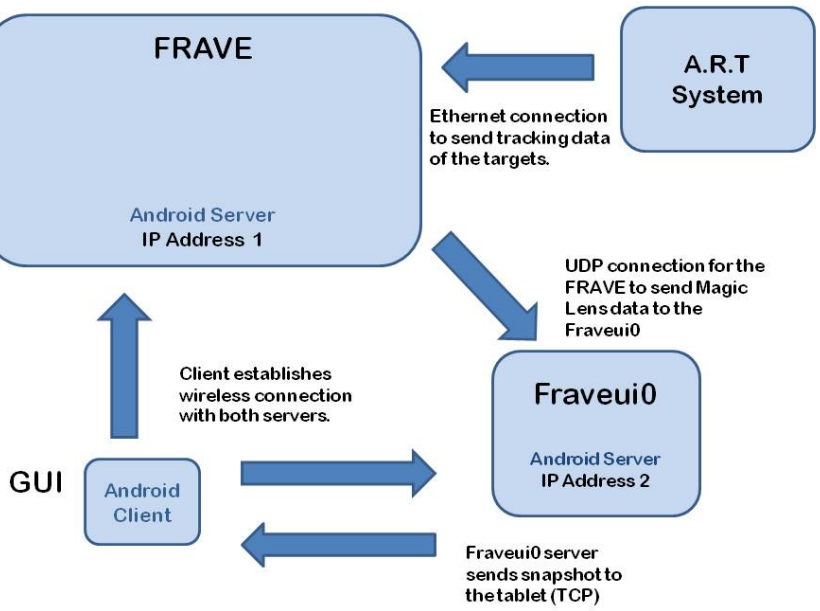



Figure 3.7: System architecture simplified scheme

### 3.2.2 Communication Procedure Overview

For better comprehension of the whole system architecture in this section the whole process will explained step by step from the moment the user starts the application until the user selects a specific region of the terrain data set after having explored it with the Magic Lens.

1. Firstly once the Terrain3D is running on both the FRAVE and Fraveui0 the user starts the Android application and a set of data tokens are sent to both machines from the Android Client so that the selected preferences are set.

2. After having started the application the user can navigate through the terrain by touching the screen of the tablet and moving it using the already implemented navigation

[1] http://campar.in.tum.de/UbiTrack/WebHome

functionality. This action will tell the Android Client to send another message to both visualization machines so that the tracking is enabled and the poses of the targets can be retrieved and updated. In this step the user can navigate and choose the desired view.

3. Once the user has chosen the desired view he can select the Magic Lens mode by pressing a button in the screen. This will also send a message to both machines to indicate that the Magic Lens functionality has been enabled and it will render the Magic Lens in the FRAVE where the background view is offered. In this step the FRAVE will also send to the Fraveui0 the information specified in section 3.2.1 so that the *lens frustum* can be visualized offering an alternative view.

4. The user can now explore the virtual world with the Magic Lens by touching the screen of the tablet and moving it in the defined tracking volume. Again a message will be sent to both FRAVE and Fraveui0 to enable the tracking and the server in the FRAVE will send the server in the Fraveui0 the updated information of the Magic Lens so that the *lens frustum* is updated dynamically.

5. After having selected a specific region of the terrain with the Magic Lens the user can take a snapshot of it by pressing another button in the Android application and the Android Client will send a message to both servers although the server in the FRAVE will ignore this message because the desired view is the one displayed in the Fraveui0.

6. After receiving the snapshot request the server in Fraveui0 will capture a 2D image of what is displayed in its screen and will send it through the TCP connection to the Android Client for the Android application to display.

In figure 3.8 we can see the whole setup, the FRAVE where the Magic Lens is rendered, the Fraveui0 where the *lens frustum* is displayed and the tablet with the snapshot of the selected area.
It is important to bear in mind that the tracking data is constantly being sent to the FRAVE machine every time there is an update, but it will only be retrieved by the Terrain3D when the user enables the tracking by touching the screen.

### 3.2.3 Issues in the initial design

In the initial architecture the connection between both servers was omitted. Instead the Fraveui0 also received the tracking data through the Ethernet connection and it obtained the position and orientation of the Magic Lens by mapping the tablet's position and orientation. However this was a problem because the machines had different frames rates to render the terrain and it is in each frame that the tracking data was retrieved to update the position and

orientation of the Magic Lens in the terrain.

The different frame rates were an issue because both visualization machines were not synchronized and the number of samples of the tracking data they obtained was different and therefore the data they processed was different for each instant of time.

This synchronization problem was particularly critical when the user stopped touching the screen, action that disables the tracking because due to the different frame rate, by the time the tracking was disabled the fastest machine would have retrieved additional tracking samples that the slowest machine had not. This caused that the last update of the position and orientation of the Magic Lens, was different for each of the machines.

To solve this problem the connection between FRAVE and Fraveui0 was added and it was no longer necessary that the A.R.T system sent the tracking data to the Fraveui0 machine. This way they would both share the same information about the Magic Lens so the *lens frustum* shown would correspond exactly to the region selected by the Magic Lens in the background view, which did not happen with the first architecture.

Figure 3.8: Image showing the whole scenario after having received the snapshot

# 4 Exploring the Virtual World using the Android Tablet

The main goal of this thesis is to investigate several interaction techniques through the implementation of the Magic Lens metaphor. The Magic Lens provides us with a tool to explore the virtual environment, in this particular case, a 3D terrain which represents the landscape of the state of Utah, while procuring some visual feedback to the user. It is also a tool which offers an alternative view of the region of interest which can be selected for further manipulation.

The area of interest is selected by means of a hand held device which will be changed in position and orientation in the real world and this will result in a change of position and orientation of the Magic Lens in the VE.

Once some region of the VE is selected it is necessary to compute the view frustum bearing in mind the eye view point and the position and orientation of the magic lens, so that we can take a snapshot of the desired frustum in the hand held device and manipulate it there.

In order to achieve this, there are certain aspects to consider, since it is important to understand how the viewpoint of the user, the movement of the lens and the virtual camera play an important role. Several different interaction modalities were developed each one offering variations concerning the viewpoint of the lens, the motion of the Magic Lens and the visualization of the Magic Lens in the terrain, as will be explained next.

The Android tablet acts as an input device and as the Graphical User Interface (GUI) and it will be in this chapter also discussed.

## 4.1 The Magic Lens virtual avatar

The Magic Lens is represented by a virtual avatar in the Terrain3D in the form of a rectangle imitating the shape of the hand held device. This rectangle represents the frame through which the alternative view is obtained as if it were a window to the virtual world [9]. This avatar is the visual feedback provided to the user and it is also the tool that will enable the

user to explore freely the virtual world. To determine the dimensions of the rectangle, first the dimensions of the tablet screen were obtained, which is easily done knowing the positions of the upper left, lower left and lower right corners of the device's screen, which are stored during the calibration process. So once the width and height of the tablet are known they are scaled by a 400 factor since the terrain has a 1m resolution and the tablet has dimensions of a few cm. This scaling factor was necessary so that the Magic Lens had a considerable size for the user, otherwise it would be appear rather small in the VE.

Initially when the Magic Lens was selected the virtual avatar was rendered in a fixed starting position in the Terrain3D in the primary view (the one visualized in the FRAVE). Then the user could move freely the Magic Lens in the virtual world and the fact that the virtual avatar was just a rectangle was confusing for the user because he could loose notion of where the lens viewpoint was, after rotating the lens.

For this reason two alternatives to solve this problem were implemented.

### 4.1.1 Turning camera

To avoid confusing the user as to were the lens viewpoint was, one of the options is to make the virtual camera turn with the Magic Lens so whenever the avatar is rotated the virtual camera also rotates with it and therefore the viewpoint is always in front of the camera. This approach only affects the orientation of the virtual camera but its position remains fixed even if the lens draws away or comes near to the camera.

### 4.1.2 Changing the appearance of the Magic Lens

The alternative that was implemented was a change in the appearance of the Magic Lens so that the viewpoint of the lens was also shown and the borders of the pyramidal volume defined by the viewpoint and the lens was also rendered. The surface of the pyramidal volume was also shaded with the sufficient transparency so that the user could see through it. This representation gives a better hint to the user as to were the viewpoint of the lens is while maintaining the virtual camera fixed. Finally the intersection point, between the four rays that start at the viewpoint and go through the lens corners, with the terrain surface was calculated and if there was an intersection these rays were also rendered and the areas defined by them shaded so that the user could then appreciate the surface of the terrain that would be visible through the lens, which is defined by the four intersection points. This way the lens frustum is rendered in the virtual world to avoid the user being confused. So the final result would be as shown in figure 4.1.

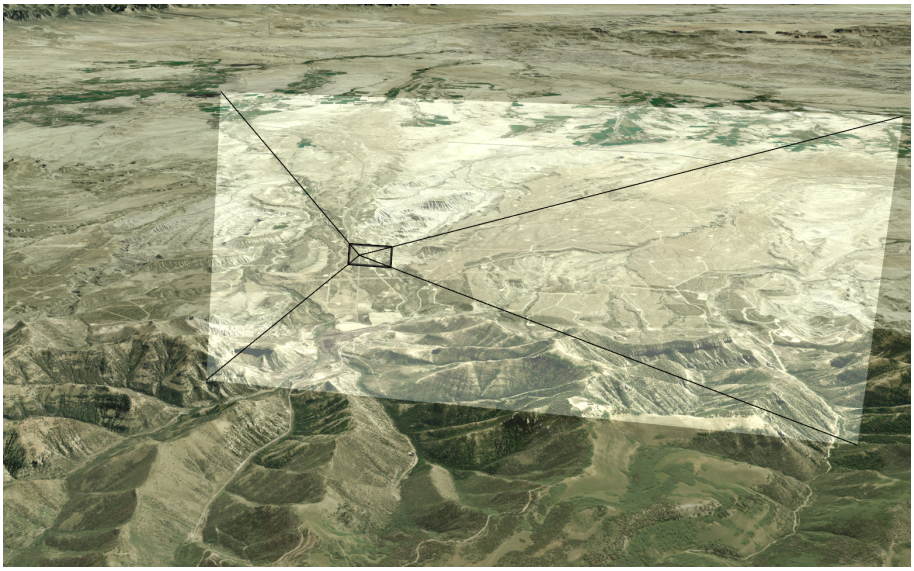


Figure 4.1: Screen capture of the appearance of the Magic Lens avatar

## 4.2 Magic Lens Motion Control

The Magic Lens virtual avatar is controlled through the hand held device so there were different techniques considered for its movement in the virtual world. Two motion control options are available to the user: rate control and direct avatar. For both of them the Magic Lens is rendered in the terrain at a starting position and it moves in accordance to the change in position and orientation of the tracked hand held device. However the movement of the hand held device in the real world is mapped differently to virtual world in each of these techniques.

### 4.2.1 Rate control

For the rate control option whenever the Magic Lens mode is selected in the Android application the virtual avatar will be rendered 300m in front of the virtual camera view direction and it will initially be orientated so that it is looking towards the surface of the terrain, this is with a 90 degrees pitch, which the rotation around the right vector of the Magic Lens (figure 3.6).

In order to move the Magic Lens in the VE the user has to touch the screen, enabling the tracking. Whenever this action is performed the pose of the tablet at this instant is stored as the initial pose and as the user moves the tablet, the delta translation and rotation is calculated relative to the initial pose.

To calculate the delta translation only the position change of the tablet in the real world is considered, independent of the rotation of the device. This way when the user holds the tablet in front of him, and moves it towards the FRAVE display, the Magic Lens will move forward in the view direction of the virtual camera and will draw away from the camera. Analogously when the user moves the tablet side-ways the Magic Lens will move along the right vector of the virtual camera(figure 3.5) and when he moves it vertically the avatar will move along the up vector of the virtual camera.

The delta translation will be updated for each frame to update the position of the Magic Lens but it will be multiplied by a sensitivity factor so the movement can be appreciated in the VE:

*translation=deltaTranslation*sensitivity*

The above formula applies for all of the three directions and the sensitivity factor considered appropriate used for this option was of 20 units.

If the movement of the tablet does not exceed a threshold of 1.5cm from the initial position the delta translation would be considered zero and the Magic Lens will not move. This is done to avoid unintended movements like hand shaking. Once that the translation in each of the directions is known a rate control algorithm was implemented to avoid hand fatigue of the user, because the application is meant for a large display like the FRAVE setup and the avatar can move through a large space. So whenever the translation exceeded a threshold of 0.8m, already considering the sensitivity factor, in any of the directions the rate control is activated and the further the hand held device is drawn away from the initial position the faster the movement of the avatar will be in the VE. So each time the threshold is exceeded the rate factor will be increased by a value proportional to the translation and this factor will be added to the current translation:

```
if (translation > 0.8) {
   rate += (translation - 0.8);
}
if (translation < -0.8) {
   rate += (translation + 0.8);
}
```

When considering the calculation of the delta rotation of the tablet its local coordinate system was considered (figure 3.3). The rotation around the X axis of the tablet, also defined as

pitch and the rotation around the Y axis of the tablet, also defined as roll, were considered, always relative to the initial position. The rotation around the Z axis was on the contrary ignored because this rotation could confuse the user because the view offered by the Magic Lens could be tilted upside down or rotated and this doesn't seem intuitive. Also some limitations were imposed to the pitch so that the it doesn't exceed a 90 degrees angle to avoid again getting an upside down view.

So the delta angle for the pitch and the roll of the tablet are obtained for each frame and mapped accordingly to the pitch and the heading of the Magic Lens so that each delta is added up to the previous one to compute the total pitch and heading. A rate control mechanism was also implemented for the rotation, using the same algorithm as for the translation, so that the Magic Lens rotates faster as the angle becomes larger. So for the pitch whenever the rotation exceeds the threshold angle of 15 degrees the rate factor is increased by a value proportional to the delta rotation and this factor is added to the total pitch. For the heading the threshold angle was set to 30 degrees since the range of values it can have is larger.

### 4.2.2 Direct avatar

The alternative to the rate control option is that the Magic Lens behaves as a direct avatar of the hand held device, this meaning that the tablet's movement, position and rotation, is mapped to the virtual avatar's movement giving the impression to the user that it follows him. This is achieved through position control, so whenever the tablet is moved the virtual avatar will move proportionally in the VE.

To implement this technique thoroughly it would have been necessary to know the exact position and orientation of the screen corners of the FRAVE setup. This way the distance from the tablet to each of the corners could have been obtained and the Magic Lens would have been rendered in the virtual world accordingly. However an approximation was developed using position control. For this, two additional A.R.T targets were tracked, one of them fixed in the left side of the upper left screen and the other one in the right side of the upper right screen of the FRAVE. Knowing the position of the left and right wall of the FRAVE and the position of the tablet, the distance to each of the walls can be calculated. This information allows us to know where the user is holding the tablet in relation to both of these walls and therefore it can be known if the tablet is near to the left wall, to the right one or if the tablet is in the middle of both. Depending on its proximity to either one the walls, when the Magic Lens mode is selected, the avatar will be rendered in the nearest pair of screens. If the tablet is held out of the space delimited by both walls, then the Magic Lens is not rendered.

In first place the distance from the tablet to both of the wall targets is calculated $d_l$ (distance to left wall) and $d_r$ (distance to right wall). Secondly the separation distance between both walls is obtained $d$. When the Magic Lens mode is selected there are five different scenarios:

1. $d_l$-$d_r \geqslant threshold$ and $d_l > d$: the Magic Lens is not rendered since it is to the right side of the right wall.

2. $d_r$-$d_l \geqslant threshold$ and $d_r > d$: the Magic Lens is not rendered since it is to the left side of the left wall.

3. $d_r$-$d_l \geqslant threshold$ and $d_r < d$: the Magic Lens is rendered in the middle of the left side pair of screens since it nearer to them.

4. $d_l$-$d_r \geqslant threshold$ and $d_l < d$: the Magic Lens is rendered in the middle of the right side pair of screens since it nearer to them.

5. $d_l$-$d_r < threshold$ : then the Magic Lens is rendered in the middle of the central pair of screens since the difference between both distances does not exceed an established threshold and therefore they are quite close in value.

So once the Magic Lens is set to its corresponding initial position when the user touches the screen of the hand held device and moves, the virtual avatar will follow. The translation of the Magic Lens is calculated in the same way as in section 4.2.1 but this time the sensitivity factor for each direction varies. When the user moves the tablet towards the FRAVE display the Magic Lens will move in depth in the terrain, towards the positive values of the view direction vector of the camera, and so the the further away the Magic Lens is, the larger the distance it has to travel in the right direction and the up direction of the camera, to disappear from the display. This way depending on the distance from the camera to the plane defined by the Magic Lens, the sensitivity for the the right direction and the up direction will vary, being larger as the Magic Lens draws away into the terrain.

These sensitivity factors were obtained mainly by testing. The sensitivity factor for the view direction was set at a constant value of 300 units. A suitable value for the sensitivity factor of the right direction translation was the distance between the virtual camera and the Magic Lens center position. And finally the sensitivity factor for the up direction was set to half of the sensitivity factor of the right direction. This way the further away the Magic Lens is from the camera the larger the sensitivity factor will be for the right and up directions.

Another aspect to consider is the calculation of the rotation of the Magic Lens. Since the virtual avatar has to behave as the tablet the orientation of the Magic Lens has to be mapped

to that of the tablet whenever the user is touching the screen. The same approach as in the previous section is not valid because the change in orientation was calculated taking into account the initial orientation which is obtained every time the user touches the screen to enable the tracking. Here we need to calculate the total rotation, so the initial orientation was defined as to have a zero value for the pitch, heading and roll. This way the delta rotation is obtained relative to this orientation. However the rotation of the virtual camera has to be taken into account because if the camera has non-zero value for the pitch or the heading this value will have to be added to that of the Magic Lens. If for instance the virtual camera has been rotated 90 degrees around its up vector (heading), then the heading of the Magic Lens should be the delta heading from the initial zero orientation plus the heading of the virtual camera because if the virtual camera rotates it is as if in the real world the user had also rotated to get another view. The same applies for the pitch.

## 4.3 The Magic Lens viewpoint

The Magic Lens viewpoint represents the point from where the user would be looking through the lens in order to visualize the *lens frustum*. Depending on its position relative to the Magic Lens the *lens frustum* changes and therefore what is displayed in the secondary view (the Fraveui0 disaply) varies. In the Terrain3D running in the Fraveui0 machine the virtual camera needs to be set to this viewpoint in order to show the lens frustum. Two alternative metaphors were implemented in this thesis.

### 4.3.1 Fixed viewpoint

For this approach the Magic Lens viewpoint is fixed at a certain distance in front of the centre position of the Magic Lens avatar so that the view direction is perpendicular to the area defined by the Magic Lens. This way the *lens frustum* is symmetrical and the near plane is defined by the Magic Lens itself and its dimensions would define the distance to the top, bottom, left and right planes as shown in figure 4.2. The far plane has always the same value, 400000m independently of the viewpoint's position.

The fact that the viewpoint is fixed makes it unnecessary to track the user's head. Knowing the orientation and position of the Magic Lens in the virtual world, which is defined by its local coordinate system, the viewpoint is placed at a distance of 40m in the virtual world in the negative values of the view direction vector (figure 3.6). To be able to show the *lens frustum* in the Fraveui0 the virtual camera of the terrain is assigned with the same position as the Magic Lens viewpoint and the same orientation as the Magic Lens and the distance to the top, bottom, left, right, near and far plane are computed so that the view frustum matches the

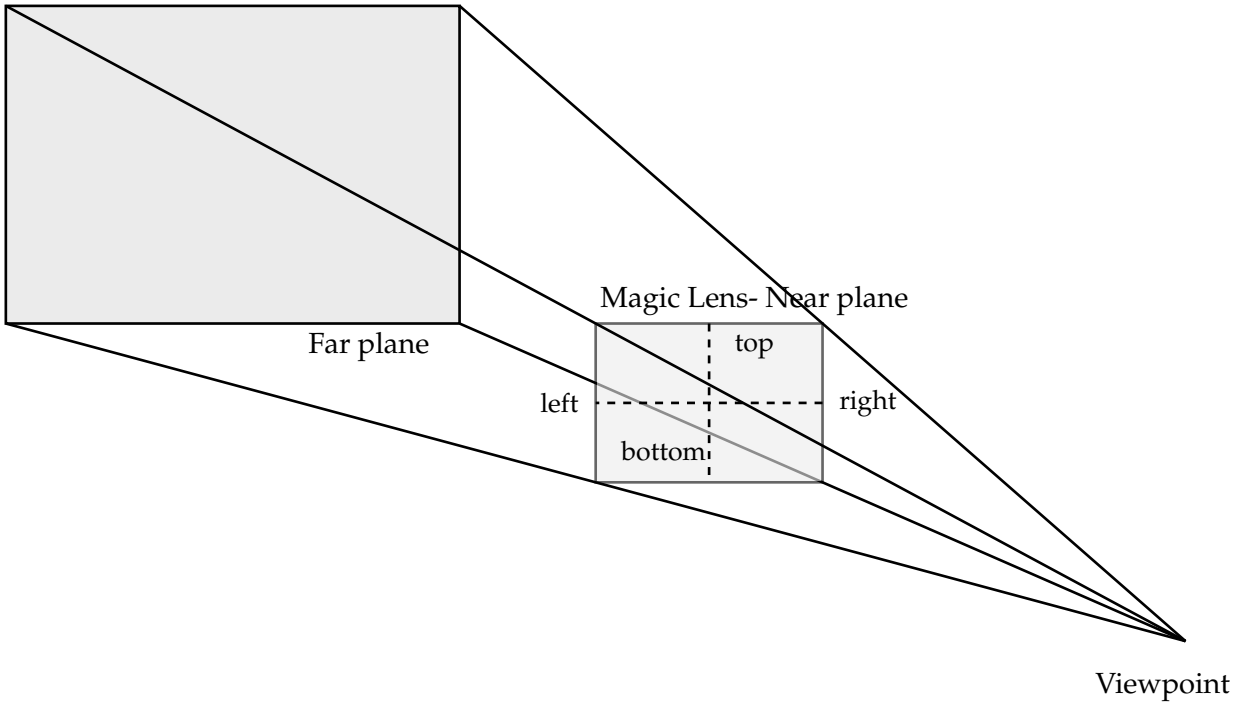



Figure 4.2: Lens frustum showing the distance to the four planes

*lens frustum*. The calculation of this symmetric frustum is just a particular case of the general off axis frustum case that will be explained in the following section where the viewpoint is set to the user's eye.

### 4.3.2 Viewpoint set to the user's eye

For this approach it was necessary to track the user's head in order to determine the relative position of the user's eye to that of the center of the tablet. For this purpose the glasses with the A.R.T target were used so in the real world the tracking system would provide us with the positions of the tablet target and of the glasses target. The offset transformation is applied to obtain the the position of the centre of the tablet ($C_t$). The relative position of the eye viewpoint ($E$) , set to the glasses target, to that of the centre of the tablet is obtained by the transformation vector: ($E_r$)= ($E$)-($C_t$). Once we have this transformation vector for the real world it is scaled by the same factor as the Magic Lens dimensions, and it is added to the the centre position of the the Magic Lens ($C_{mg}$) in the virtual world in order to obtain the virtual viewpoint ($E_{virtual}$).

Even though we are obtaining a 6DOF pose of the user's eye, the position relative to the

tablet is only taken into account since it will be assumed that the user is always looking through the center of the lens so the orientation of the user's head is not considered.

Once we have obtained the virtual viewpoint position ($E_{virtual}$) the next step is to compute the *lens frustum* which is not necessarily symmetric as in the previous section. However the same calculations apply to both cases. This calculations follow the previous works of Heuser in his 'Window into a virtual world' screen concept [9]. The only difference is that the near plane is set to the Magic Lens plane.

First we need to know the rotation of the Magic Lens relative to the viewpoint, so for that we calculate the two perpendicular vectors that are defined by the Magic Lens dimensions and a third vector perpendicular to the plane defined by the other two. This was done using the coordinates of three of the corners of the Magic Lens, the upper left corner ($U_{left}$), the lower left corner ($L_{left}$) and the lower right corner ($L_{right}$):

$$X_{axis} = L_{right} - L_{left}$$
$$Y_{axis} = U_{left} - L_{left}$$
$$Z_{axis} = X_{axis} \times Y_{axis}$$

This vectors where then normalized. The next step is to calculate the relative position ($R_{vp}$) of the virtual viewpoint to that of the lower left corner of the Magic Lens:

$$R_{vp} = E_{virtual} - L_{left}$$

Next we need to determine the distance to the near plane which will be the norm of a vector which goes from the virtual viewpoint position to the plane defined by the Magic Lens and is perpendicular to this plane. This distance is determined by the following scalar product:

$$d_{near} = R_{vp} \bullet Z_{axis}$$

The next step is to compute the distance from this same vector to the left, right, top and bottom borders of the Magic Lens because they will define the distance to the left ($d_{left}$), right ($d_{right}$), top ($d_{top}$) and bottom ($d_{bottom}$) planes of the frustum.

$$d_{left} = -R_{vp} \bullet X_{axis}$$
$$d_{right} = width_{ml} - d_{left}$$
$$d_{bottom} = R_{vp} \bullet Y_{axis}$$
$$d_{top} = height_{ml} - d_{bottom}$$

Once these distances are computed we have the necessary information to set the projection matrix using the OpenGL function *glFrustum*. The virtual camera for the Fraveui0 will

be assigned with the virtual viewpoint position and the orientation of the Magic Lens. This calculation is also applied to compute the *lens frustum* for the fixed viewpoint case since it is just a particular case.

Since the viewpoint for this approach changes in position it could happen that the user moves his head behind the tablet. So for each update of the glasses and the tablet's position if the distance to the near plane is lower than 1m in the virtual world the viewpoint will be set 1m in front of the Magic Lens. Otherwise the distance to the near plane could end up being negative and the view would not be correct.

The tracked viewpoint approach only makes sense when using the direct avatar approach. The reason for this is that for the rate control option the user is able to rotate the Magic Lens without having to rotate the tablet by the same angle. So while the user might be holding the tablet at a small angle from the vertical direction, the Magic Lens might be totally horizontal in the virtual world, so the relative position of the viewpoint to the centre of the tablet would not correspond to the same relative position in the virtual world so there is not an exact mapping from the real world to the virtual world. Therefore the rate control option is only combined with the fixed viewpoint approach, while the direct avatar option can be combined with both fixed viewpoint and tracked viewpoint.

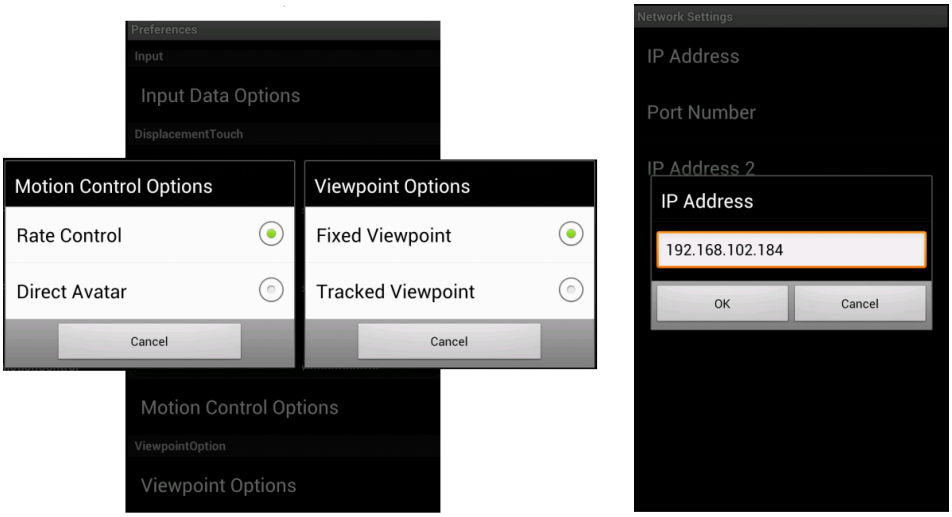
## 4.4 Graphical User Interface

The Graphical User Interface for this work extends the GUI used for the Navigation application developed by Ashry [14] and therefore it can be used for navigation purposes and to control the Magic Lens. Two different views are therefore available and the user can switch between both of them from Navigation mode to Magic Lens mode through a button click. Initially the Navigation view is visible which provides the necessary buttons to start and stop the application, as well the as button labeled 'Magic Lens' to switch modes. The user can also specify certain preferences, which will change the behaviour of the Magic Lens and the network settings. The screen orientation is set to landscape so the user is intended to hold the tablet in the same way.

### 4.4.1 Preferences view

This view is enabled when the user touches the menu button of the device ans selects the 'Preferences' option, so a list of preferences is provided. The additional preferences integrated in the GUI for the Magic Lens application were the 'Motion Control Options' and the 'Viewpoint Options' as seen in figure 4.3(a). Through the 'Motion Control Options' the user can

select the rate control mode or the direct avatar mode and through the 'Viewpoint Options' either the fixed viewpoint can be enabled or the tracked viewpoint. Whenever the tracked viewpoint is selected the motion control option will be set automatically to the direct avatar. These preferences will be then sent to the Android Servers upon clicking the 'Start' button so if any changes are made in the preferences the user needs to restart the application to inform the Android Server of the new options selected.




(a) Motion Control Options and Viewpoint Options menus (b) Network Setting menu to change IP Addresses and port

Figure 4.3: The Preferences (a) and the Networks Setting (b) views of the GUI

### 4.4.2 Network Settings view

This view is provided by also touching the menu button of the device and by selection the 'Network Settings' option (figure 4.3(b)) . Through this menu the the user can edit the IP addresses to which the Android Client will send the data, and the corresponding port. These IP addresses have to be set to that of the FRAVE and the Fraveui0 machines, where the Android Servers are running.

### 4.4.3 Navigation view

Initially the Navigation view is visualized with a green 'Start' button, a red 'Stop' button and the 'Magic Lens' button as seen in figure 4.4. When pressing the 'Start' the application is started and therefore it sends all the data set in the 'Preferences' to both machines, where the Android Servers are running and it enables the user to navigate through the VE by touching

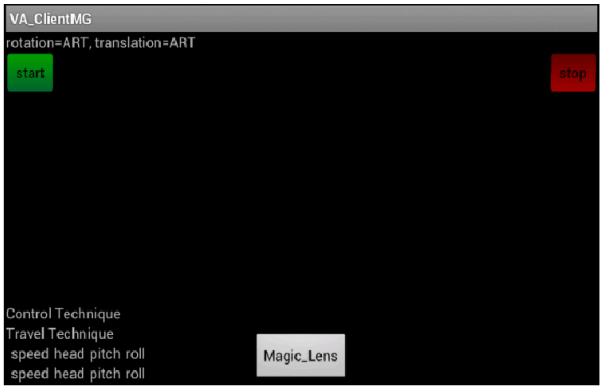



Figure 4.4: Initial view of the GUI

the screen and moving the tablet. The screen touch enables the A.R.T tracking and therefore a String message is sent to both IP addresses specified in the 'Network Setting' through the UDP connection mentioned in chapter 3. This way the user can travel to obtain the desired view before using the Magic Lens.

To start the Magic Lens application the user has to click the 'Magic Lens' button so the view will be flipped and also a String message will be sent to the Android Servers indicating that the Magic Lens mode has been selected, which will render the Magic Lens at its initial position. The red 'Stop' button disables the sensors of the device and sends also a message to the Android Servers.

### 4.4.4 Magic Lens view

This view consists of two buttons, one to exit the Magic Lens mode, which switches back to the Navigation view, and the other to take a screen capture of what is visualized through the lens. They are labeled 'Exit Magic Lens' and 'Snaphot' (figure 4.5).

When the exit button is clicked on, a message is sent to the Android Servers so that the Magic Lens is no longer rendered in the VE. While this view is enabled the user by touching the screen and moving the tablet can move the virtual avatar in the virtual world, according to the specified motion control options. The screen touch as in the Navigation view enables the A.R.T tracking for both the tablet and the glasses, if the tracked viewpoint option is selected. Once the user has selected a region of interest in the VE by clicking the 'Snapshot' button, the Android Client requests the Android Server running in the Fraveui0 to send the image corresponding to the lens frustum, and once the image is received it is displayed in the tablet.

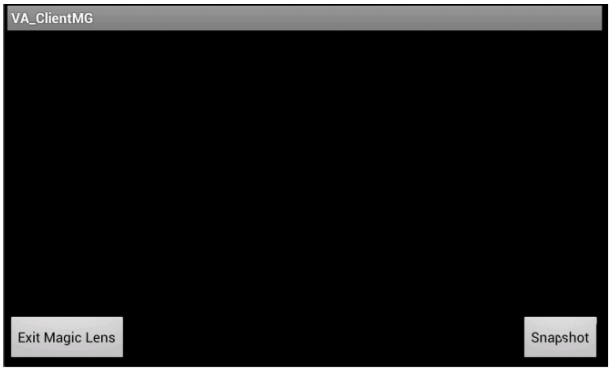



Figure 4.5: Magic Lens view of the android application

The user can repeat this process by returning to the Magic Lens view with the back arrow of the device and selecting any other region of interest.

# 5 Implementation

A more detailed explanation of the most relevant aspects of the Magic Lens interaction techniques will be presented in this chapter. It will be explained how the Magic Lens coordinates are obtained and updated according to the tablet's movement for the different motion control options and how the Magic Lens is rendered with the OpenGL [17] libraries. Another relevant aspect is to understand how the snapshot of the *lens frustum* was obtained in Fraveui0 and how the image was compressed in order to send it within a suitable time to the Android Client running in the tablet. The TCP receiver of the Android Client, implemented to receive the image from Fraveui0, will also be discussed.

When obtaining the pose of the A.R.T targets the data was so accurate that some thresholds had to be imposed and averaging was implemented, and this aspect will be also explained.

## 5.1 Rendering the Magic Lens

In order to render the virtual avatar at its start position in the FRAVE, each time the Magic Lens mode is enabled, the coordinates of the centre of the rectangle need to be determined as well as its local coordinate system 3.6. The start position varies depending on the motion control option. For the **rate control** option the Magic Lens is rendered 300m in front of the camera so the start position is obtained as follows:

```
centreStartPos = camPosition+ 300 * direction_cam;
```

Where `direction_cam` is the view direction vector of the virtual camera.

For the **direct avatar** option, the initial centre position of the rectangle varies depending on where the user is standing (section 4.2.2). So if the user is standing in front of the central pair of screens of the FRAVE then the start position will be the same as above but if the user is standing near to the right pair of screens or to the left pair of screens the initialisation would be:

```
//Magic Lens drawn in the right side pair of screens
centreStartPos = camPosition + 300 * direction_cam+ 300 * right_cam;
```

```
//Magic Lens drawn in the left side pair of screens
centreStartPos = camPosition + 300 * direction_cam- 300 * right_cam;
```

Since initially the Magic Lens is rendered looking towards the negative Z axis of the terrain, so that there is intersection with the terrain surface, the pitch of the Magic Lens is set to 90 degrees, while the heading and roll of the Magic Lens are set to the same values as those of the virtual camera. Knowing the heading, pitch and roll of the Magic Lens we can determine the vectors that define the local coordinate system of the rectangle that represents the Magic Lens, `up_rec, right_rec` and `direction_rec`.

Once the centre position of the rectangle and the local coordinate system vector are known the corners of the Magic Lens are obtained as follows:

```
//Upper Left corner
up_left = height/2 * up_rec - width/2 * right_rec;
u_left = centrePos + up_left;
//Upper Right corner
up_right = height/2 * up_rec + width/2 * right_rec;
u_right = centrePos + up_right;
//Lower Right corner
low_right = -height/2 * up_rec + width/2 * right_rec;
l_right = centrePos + low_right;
//Lower Left corner
low_left = -height/2 * up_rec - width/2 * right_rec;
l_left = centrePos + low_left;
```

Where `up_left, up_right, low_right` and `low_left` are the vectors that define the transformation from the centre position to the corners of the Magic Lens and `width` and `height` are the dimensions of the Magic Lens.

For each frame if the A.R.T tracking is enabled by the user touching the screen of the tablet, the centre position of the rectangle will be updated, adding up the translation in the three different directions. We need to multiply the translation values for the different directions of the real world by the camera vectors and add it to the starting position:

```
centrePos = centreStartPos + translationx * right_cam
```

```
                                    + translationy * up_cam
                                    + translationz * direction_cam;
```

Also the heading and pitch values are updated according to the movement of the tablet, and the roll, which is the rotation around the view direction vector of the Magic Lens, will remain zero so that we don't get a tilted view as it was already mentioned. So for each frame the updated centre position of the Magic Lens and the heading and pitch angles are obtained, and are used to compute the the coordinates of the corners of the Magic Lens.

The virtual viewpoint through which the user looks through the lens is also updated for each frame. For the **fixed viewpoint** metaphor it is fixed 40 metres in front of the lens:

```
viewPos = centrePos - 40 * direction_rec;
```

Whereas for the **tracked viewpoint** the transformation vector that defines the relative position of the user's eye to that of the centre of the tablet was mapped to the virtual world and scaled so that the same transformation was applied to the centre position of the Magic Lens. The transformation vector had to be multiplied by the camera vectors which define its coordinate system, to obtain the coordinates of the viewpoint position in the terrain.

```
viewPos.x()=centrePos.x()
            -scaleFactor*distRW*trans_vec.z()*direction_cam.x()
            +scaleFactor*distRW*trans_vec.x()*right_cam.x()
            +scaleFactor*distRW*trans_vec.y()*up_cam.x();
viewPos.y()=centrePos.y()
            -scaleFactor*distRW*trans_vec.z()*direction_cam.y()
            +scaleFactor*distRW*trans_vec.x()*right_cam.y()
            +scaleFactor*distRW*trans_vec.y()*up_cam.y();
viewPos.z()=centrePos.z()
            -scaleFactor*distRW*trans_vec.z()*direction_cam.z()
            +scaleFactor*distRW*trans_vec.x()*right_cam.z()
            +scaleFactor*distRW*trans_vec.y()*up_cam.z();
```

As it can be seen the negative value of the Z coordinate in the real world is scaled and multiplied by the view direction vector of the camera. The values of the X coordinate and of the Y coordinates of the real world vector were scaled and multiplied respectively to the right

vector and the up vector of the virtual camera.

Once the position of the virtual viewpoint and of the Magic Lens corners are obtained, to render the virtual avatar as a pyramid, we need to calculate the four intersection points of the four rays formed by the viewpoint and the lens's corners, with the terrain surface. With this information and after setting the corresponding projection matrix and view matrix the Magic Lens is rendered by using OpenGL functions.

First the rectangle is drawn:

```
//Rectangle
glBegin(GL_LINE_LOOP);
//Upper Left corner
glVertex3f(u_left.x(), u_left.y(), u_left.z());
//Upper Right corner
glVertex3f(u_right.x(), u_right.y(), u_right.z());
//Lower Right corner
glVertex3f(l_right.x(), l_right.y(), l_right.z());
//Lower Left corner
glVertex3f(l_left.x(), l_left.y(), l_left.z());
glEnd();
```

Then the four triangles which define the four sides of the pyramid are determined by the viewpoint position and two of the intersection points with the terrain. The blend option was enabled to make the triangles semi-transparent:

```
  glEnable (GL_BLEND);
glBlendFunc (GL_SRC_ALPHA,GL_ONE);
glBegin(GL_TRIANGLES);
//Left side triangle
glVertex3f(viewPos.x(),viewPos.y(),viewPos.z());
glVertex3f(selectedPointUPL.x(),selectedPointUPL.y(),selectedPointUPL.z());
glVertex3f(selectedPointLOL.x(),selectedPointLOL.y(),selectedPointLOL.z());
//Upper side triangle
glVertex3f(viewPos.x(),viewPos.y(),viewPos.z());
glVertex3f(selectedPointUPR.x(),selectedPointUPR.y(),selectedPointUPR.z());
glVertex3f(selectedPointUPL.x(),selectedPointUPL.y(),selectedPointUPL.z());
//Bottom side triangle
```

```
glVertex3f(viewPos.x(),viewPos.y(),viewPos.z());
glVertex3f(selectedPointLOR.x(),selectedPointLOR.y(),selectedPointLOR.z());
glVertex3f(selectedPointLOL.x(),selectedPointLOL.y(),selectedPointLOL.z());
//Right side triangle
glVertex3f(viewPos.x(),viewPos.y(),viewPos.z());
glVertex3f(selectedPointUPR.x(),selectedPointUPR.y(),selectedPointUPR.z());
glVertex3f(selectedPointLOR.x(),selectedPointLOR.y(),selectedPointLOR.z());
glEnd();
```

To enhance the visualization of the Magic Lens and so the user could get a better notion of where the viewpoint is and what is the central point of the region of the terrain selected two spheres were drawn. A red sphere was drawn at the viewpoint position with a small radius and a bigger sphere was rendered in the intersection point of the ray, from the viewpoint and through the centre of the lens, with the terrain. Also four lines that represented the four rays that go through the lens's corners were rendered for the same reason.

## 5.2 Obtaining and sending the snapshot

Whenever the user has explored the VE with the Magic Lens and has selected a particular region of the terrain he can obtain a 2D image of the selected area. This image is captured in the Fraveui0 machine where the scene can be seen from the Magic Lens viewpoint, so once the snapshot request is received the pixels that are being displayed in the screen are stored in a buffer. So knowing the viewport dimensions the raw data can be obtained using the OpenGL function:

```
  glReadPixels((GLint)0,(GLint)0,(GLint)ViewPortWidth-1,
(GLint)ViewPortHeight-1,GL_RGB, GL_UNSIGNED_BYTE, bmpBuffer);
```

Initially the raw data was encoded to the bitmap format and then sent to the Android Client, but since the image had a considerable size the transfer was too slow. To solve this problem the image was compressed to JPEG format by using the open source library courtesy of Geldreich [18]. This way the image size was smaller and the transmission time was lower ($\sim 174ms$). The compressed image was stored in a buffer and the Android Server sent this buffer through a TCP connection.

In the Android Client the appropriate TCP receiver was implemented to receive the data stream, so that it kept reading from the socket until all the data was received. So first it listens for the connection request and once the connection is established, it starts reading chunks of

bytes from the socket until no more data is received.

```
server = new ServerSocket(port);
sckt = server.accept();
in = new BufferedInputStream(sckt.getInputStream());
out = new ByteArrayOutputStream();
do {
   rd = in.read(recvData, 0, recvData.length);
   if (rd != -1) {
      out.write(recvData, 0, rd);
   }
}while(rd != -1);
```

These bytes are then arranged into a buffer and then decoded using the `BitmapFactory` Android class so that the image is displayed in a separate android activity of the application using the `ImageView` Android widget which loads the image.

## 5.3 Tracking data: thresholds and averaging

Since the user holds the tablet with the attached target in his hands there might be some unintended gestures or even hand shaking while he is moving the Magic Lens avatar. The same applies to the glasses attached to the user's head. The tracking system is very accurate and these unintended movements are registered. To avoid this affecting the movement of the avatar and getting an unstable view in the Fraveui0 some thresholds were imposed.

When calculating the translation of the tablet the difference between the immediately previous value and the current one is computed so if this difference does not exceed a determined threshold the value will be added up to a variable (initialised to zero) and the translation will be set to the previous value. If the difference does exceed the threshold, then the average of the previous added values will be computed and assigned to the current translation value, and the add up variable is set to zero again. The same procedure takes place with the translation of the glasses.

For the roll and the pitch of the tablet the same was done so that when the rotation around each of the axes exceeded a certain threshold angle the average value was assigned to the current rotation angle.

# 6 Conclusion

Nowadays 3D environments are becoming more and more popular and interaction in these environments is not as intuitive using a common mouse since its movement is bound to the 2D display. This is the reason why it is important to develop new interaction techniques. This work offers a powerful interaction technique through the Magic Lens metaphor which not only enables the user to explore a 3D virtual environment but it also serves as a tool to select certain regions of the VE and offers an alternative view of the scene. These tasks can be performed in different ways and this work offers three main metaphors:

- Rate control motion with a fixed viewpoint.
- Position control motion (direct avatar) with a fixed viewpoint.
- Position control motion (direct avatar) with tracked viewpoint.

Although no user evaluation was developed after implementing all of the different metaphors and testing them, several conclusions can be drawn.

The first of the metaphors provides the user with an intuitive motion control technique while allowing him to remain static in his place and by reducing hand fatigue. An almost full control, since the heading of the tablet is ignored, of the motion of the Magic Lens is available. The fact that the viewpoint is fixed exempts the user from using his head, but this also limits the possibility of an alternative perspective view while the lens is fixed.

The second metaphor also provides the user with a full control of the motion of the Magic Lens, but in order to explore the scene that is displayed, the user needs to move the tablet longer distances which implies that the user will have to move to cover the area of the display. This has both advantages and disadvantages, the user can enjoy a more immersive experience, as the Magic Lens behaves the same way in the virtual world as in the real world, but at the same time it can be tiring for him to have to move in order to move the lens, because by only moving the tablet from a static position he is not able to explore the whole scene with the Magic Lens.

The third metaphor enables the user to change the viewpoint by moving his head. This offers extra functionality since the user can fix the Magic Lens and he can look through it from different perspectives supplying a more interactive experience. However the viewpoint position is obtained relative to the tablet but the focus view selected by the Magic Lens is offered in the Fraveui0 display so this can be less intuitive for the user than using the fixed viewpoint metaphor.

## 6.1 Future Work

This application was specifically implemented using the Terrain3D software which offers the dataset to represent the relief of a 3D terrain but under the surface of the terrain there is no data available. It would be interesting to be able to do below surface exploration by using the Magic Lens metaphor. So for example when a certain region of the surface of the terrain is selected it would be useful to have the option to see what is below the superficial layer. For instance, if a building of the terrain is selected it would be interesting to be able to see the underlying structure of the building and the inside of it. This would be interesting because it would allow to explore the internal structure of 3D models.

Furthermore to make the application more intuitive, specially when using the tracked viewpoint metaphor it would be useful to be able to visualize the the lens frustum not only in the Fraveui0 machine but in the tablet itself. This would offer a real time image to the user of what he is selecting with the Magic Lens, instead of having him look at the secondary display and the tablet would behave as a window into the virtual world just as in Heuser's work [9].

This work has essentially focused on the implementation and therefore no user study evaluation has been done, just some expert users have provided some feedback. So as further work it would be interesting to carry out a user evaluation where all of the three metaphors are tested by the users by performing a specific selection task. This will be helpful to improve the performance of the application by maybe modifying some of the parameters such as the thresholds or the sensitivity factors.

Another aspect to improve would be to obtain the pose transformation between the target attached to the glasses and the middle point between both glass lenses. This would require a calibration process as the one that is performed to obtain the middle pose of the tablet, and it would provide better user experience when using the tracked viewpoint metaphor.

# Bibliography


[1] E.A. BIER, M.C. STONE, K. PIER, W. BUXTON and T.D. DEROSE , *Toolglass and Magic Lenses: The See-Through Interface*, SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques, Pages 73-80, ACM New York, 1993 .

[2] R. STOAKLEY, M.J. CONWAY, and R. PAUSCH, *Virtual Reality on a WIM: Interactive Worlds in Miniature*, Proc. ACM Conf. Human Factors in Computing (CHI), ACM Press, Pages 265-272,1995.

[3] J. VIEGA, M.J. CONWAY, G. WILLIAMS and R. PAUSCH , *3D Magic Lenses*, UIST '96 Proceedings of the 9th annual ACM symposium on User interface software and technology, Pages 51 - 58, ACM New York,1996.

[4] S. STOEV, D. SCHMALSTIEG and W. STRAÃER , *The Through-The-Lens Metaphor: Taxonomy and Application*, Proceedings of the IEEE VRâ02, Pages 285-286, 2002.

[5] T. ROPINSKI and K. HINRICHS , *Real-Time Rendering of 3D Magic Lenses having arbitrary convex Shapes*,In Proceedings of WSCG ,Pages 379-386, 2004.

[6] L.D. BROWN and H. HUA , *Magic Lenses for Augmented Virtual Environments*,IEEE Computer Graphics and Applications, vol. 26, no. 4, Pages 64-73, 2006.

[7] M. MIRANDA MIGUEL, K. KIYOKAWA and H. TAKEMURA , *A PDA-based See-through Interface within an Immersive Environment*,ICAT '07 Proceedings of the 17th International Conference on Artificial Reality and Telexistence, Pages 113-118, IEEE Computer Society Washington, DC, USA 2007.

[8] J. SANNEBLAD and L.E. HOLMQUIST , *Ubiquitous Graphics: Combining Hand-held and Wall-size Displays to Interact with Large Images*,SIGGRAPH '05 ACM SIGGRAPH 2005 Emerging technologies Article No. 26, ACM, New York, USA 2005.

[9] N. HEUSER, *Window into a virtual world screen concept*, Systementwicklungsprojekt , Technische Universität München, 2008.

[10] N. HEUSER, *Methoden und Metaphern zur Navigation in virtuellen Landkarten* , diploma thesis, TU Munich, 2010.

[11] S. WEBER, *Investigating Viewpoint Control Metaphors for hand-held devices in Virtual Environments*, bachelors thesis, Technische Universität München, 2011.

[12] Y. SABEV, *Selection and Zooming using Android Mobile Phone in 3D Virtual Reality* , bachelors thesis, Technische Universität München, 2011.

[13] M. MCKENNA,*Interactive Viewpoint Control and Three-dimensional Operations*, Proc. 1992 Symposium on Interactive 3D Graphics, Pages 53-56.

[14] ASHRY MOHAMED, *Development of Android touch interface for navigation in an immersive environment such as CAVE*, bachelor thesis, TU Munich, 2010

[15] *Advance Realtime Tracking GmbH*, http://www.ar-tracking.com

[16] *Mechdyne*, http://www.mechdyne.com/trackd.aspx

[17] *OpenGL Programming Guide*, http://glprogramming.com/red/index.html

[18] *R. Geldreich*, JPEG-Compressor, http://code.google.com/p/jpeg-compressor/