DEVELOPING A GENERAL FRAMEWORK FOR THE DELFT VRET APPLICATION

Siemen Roorda
Student number: 9706067
Delft University of Technology
Media and Knowledge Engineering
Supervisor: dr. ir. C.A.P.G. van der Mast
January 2008
Abstract

Developing a general framework for the Delft VRET application
S.J. Roorda (student number 9706067)
Delft, January 2008

Thesis for the master programme Media and Knowledge Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

SUPERVISOR:
Dr. ir. C.A.P.G. van der Mast

GRADUATION COMMITTEE:
Dr. ir. C.A.P.G. van der Mast
Prof. dr. M.A. Neerincx
Drs. P.R. van Nieuwenhuizen
Dr. ir. W.P. Brinkman

SUMMARY:
The Delft Virtual Reality Exposure Therapy application has proved it’s value, but due to different factors, the current implementation is, viewed from a technical aspect, not future-proof. Also, in the course of the time new requirements emerged, both for purposes of research and for usage in the clinics. This research aims at the development of a general framework that acts as a centralized application from which all virtual environments can be managed and accessed, and that allows for easy future extensions. An example implementation of the framework, and a basic virtual world together form a prototype, that serves as an example of the framework and show the new possibilities.
# Table of contents

Abstract .................................................................................. 3
Table of contents ....................................................................... 5
1 Introduction ........................................................................... 7
   1.1 Research goal .................................................................. 8
   1.2 Report outline .................................................................. 8
   1.3 Disclaimer ....................................................................... 8
2 Method ................................................................................... 9
3 The current system ............................................................... 10
   3.1 History ............................................................................ 10
   3.2 General architecture ...................................................... 10
   3.3 The virtual environments .............................................. 11
   3.4 Quick walk-through ....................................................... 14
      3.4.1 Preparing the session ............................................. 14
      3.4.2 Opening a virtual world ....................................... 14
      3.4.3 Registering data .................................................... 15
      3.4.4 Performing the session ......................................... 16
      3.4.5 After the session ends .......................................... 16
   3.5 Setup of hardware and software ...................................... 17
      3.5.1 Personal computer and operating system ............... 17
      3.5.2 Ascension Flock of Birds motion tracker ............... 18
      3.5.3 Cybermind Vissette Pro head mounted display ........ 18
      3.5.4 WorldUp and WorldUp player, and the virtual worlds 18
      3.5.5 Delphi user interface ........................................... 19
      3.5.6 Communication and network connections ............. 19
   3.6 The framework ............................................................... 19
      3.6.1 Phobia.ini ............................................................... 19
      3.6.2 Listener ................................................................. 20
      3.6.3 Therapist’s user interface ..................................... 21
      3.6.4 Patient’s user interface .......................................... 24
   3.7 Evaluation ....................................................................... 24
      3.7.1 Positive points ....................................................... 25
      3.7.2 Negative points ..................................................... 25
   3.8 Further development ...................................................... 25
      3.8.1 Graphics and VR .................................................. 26
      3.8.2 Framework .......................................................... 26
      3.8.3 Hardware ............................................................. 26
4 The new framework ............................................................. 27
   4.1 Requirements ............................................................... 27
      4.1.1 Technical requirements ....................................... 27
      4.1.2 Functional requirements ..................................... 27
   4.2 3D modeller ................................................................. 28
   4.3 Virtual Reality application ............................................. 28
      4.3.1 Quest3D ............................................................... 29
      4.3.2 Vizard ................................................................. 33
      4.3.3 Which application to choose? ............................... 35
   4.4 Framework ................................................................. 37
      4.4.1 Programming language ....................................... 37
1 INTRODUCTION

About seven years ago, in 2000, the Delft University of Technology started in collaboration with the University of Amsterdam a project which goal was to treat patients suffering from several psychological disorders, such as acrophobia (fear of heights), claustrophobia and fear of flying, using a virtual reality therapy. This project, called Virtual Reality Exposure Therapy (VRET), arose from the question whether psychological therapies could not be performed more efficiently.

In a series of treatment sessions for a phobia, the therapist will want his patient to face the situation he is afraid of. Traditionally, there are two ways of doing this: in vitro - without moving outside the treatment room, the therapist lets the patient imagine he is experiencing his fear. This method is cheap and easy, but not always sufficient due to the lack of 'real' elements that stimulate the senses. The other side of the spectrum is in vivo therapy: the patient is actually taken to the situation that he fears, for example a high rooftop terrace. This method is time-consuming and may be expensive, depending on the type of phobia.

The virtual reality exposure therapy combines the advantages of both methods by using a virtual reality experience while just staying in the office of the therapist. Virtual reality makes it possible to fool the senses - the view, the sounds, even up to some degree the feeling of the virtual world. It will only be in the distant future that virtual reality can replace the real experience for the full hundred percent, but as some research shows (Emmelkamp, 2001), also without this 100% a therapy can be very successful. Depending on the quality of the virtual reality and the feeling of presence, this type of therapy moves somewhere between in vitro and in vivo therapy:

The collaboration of the two universities was impersonated by Martijn Schuemie (Delft University of Technology, supervised by Charles van der Mast, mainly performing the technical part) and Merel Krijn (University of Amsterdam, supervised by Paul Emmelkamp, mainly performing the psychological part), and started with several researches (Schuemie, 2000b; Krijn, 2000) whether virtual reality could be a solution for combining the advantages of both in vitro and in vivo therapy.

Within certain conditions (e.g. regarding the type of phobia) virtual reality proved to be able to substitute in vivo therapy, and the VRET project started. During the years, several virtual environments have been developed for different phobias, and a basic framework has been created that allows therapists to control these environments and use them for sessions, both manipulating parameters and recording session data.

This virtual way of performing sessions proved to be, again under certain conditions, at least as effective as a normal in vivo session (Emmelkamp, 2001). Therefore, it is not wondering that the interest is growing, both from the side of scientists, as from psychological clinics. This makes VRET changing from a mainly research-oriented project to an application that can be put into a clinical environment. This is proved by a Dutch health care organization, that is interested in the VRET-system to extend their market share by offering virtual reality as a new concept of treatment.

Being put into a real environment means the application should be complete, robust, and maintainable. As a result of the research aspect, the main focus never laid primarily on theses characteristics, so in order to complete this, the application should be tested thoroughly and adapted where necessary.

Figure 1.1: VRET can be placed somewhere within the range between in vitro and in vivo therapy. The exact location depends on psychological aspects (type of phobia, the patient), and technical aspects (quality of the virtual world, environment in which the therapy takes place).

\[ \text{in vitro} \quad \text{VRET} \quad \text{in vivo} \]
Developing a general framework for the Delft VRET application

There are two problems that were encountered: the manufacturer of the virtual reality software has discontinued its activities and cannot be reached for information, support or licenses anymore. This implies we are not able to install working development environments, so that neither new applications can be developed, nor existing applications can be finetuned or altered. Also the stagnation of development of the virtual reality software makes it not useful to continue ourselves with a soon outdated version. The second problem is the framework, that is not easy to manage and to extend for new virtual worlds.

Regarding the discontinued development tools - in my research assignment (Roorda, 2005a) I performed research after a lot of other applications, and selected two of them as being interesting. The framework is the main subject of this master thesis.

1.1 Research Goal

This thesis focuses on the general framework of the VRET application rather than the virtual environments and the psychological backgrounds. The current framework will be described, and the new framework will be set up. The end product of this research will be a guidance to the new framework, including a simple prototype. The assignment consists of the following elements:

- Develop a framework that can contain and maintain all current and future virtual environments used with the virtual reality exposure therapy.
- The framework should be set up modular, and should provide a flexible user interface that can easily be adapted for a new world. Also the therapist should be able to change the user interface according his personal requirements, being able to store (and load) this so-called profile on a per-patient basis.
- For easy communication between therapist and patient (in case the treatment is in different locations), a real-time communication channel should be included, preferable in the form of a webcam conversation.
- It should be possible to extend the basic functionality with additional elements in the future; for this, one can think of heartrate sensors or other devices to measure the patient’s anxiety, an artificial intelligence extension to enlighten the therapist’s job or defining ‘levels’ in the treatment procedures.
- The virtual world’s development environment is Quest3D; the programming language used for the user interface can be chosen freely.

1.2 Report Outline

This report starts with a quick outline of the methods used in chapter 2.

The next chapter, chapter 3, analyses the old VRET system. As at the moment very little technical documentation about the old system is present, this chapter dives into details at some points. After a full historic outline and an overview of the available virtual worlds, a quick walk-through is given, to create some familiarity with the system. Each part of the hardware and software setup is described shortly, and the framework, of course, in more detail. The chapter ends with a summary of advantages and disadvantages and options for the future.

Chapter 4 handles the framework developed for this thesis, starting with the requirements analysis. The three main parts are described: the 3D modelling application, the virtual reality application and the Delphi framework itself.

Chapter 5 describes the prototype I developed: the virtual world and the framework. This chapter ends with an evaluation of the prototype.

Finally the conclusions of my research are stated in chapter 6, which also includes recommendations for further development.

The appendices include information that was not suitable to fit into the report itself, and also contain some side-information about the current system, mainly included from the view of providing historical documentation.

1.3 Disclaimer

The names of patients used in this report and in the prototype are fictive names; also their phobias and sessions are fully fictive.
2 **Method**

This research can be classified as *design science* rather than *natural science*, as it results in creating artefacts to attain goals, instead of explaining how and why things are, which is the domain of the natural sciences. In design science, there are four types of products that can be built (Schuemie, 2003): constructs, models, methods and implementations. The end result of this research will be the implementation of a prototype, derived by following a design method.

The design method I chose is the design method as put by Van der Mast (Van der Mast, 2004) as illustrated in figure 2.1. This design method is based upon behavioural or user centred design, in which the user and the user interaction with the system take a central position.

![Diagram of human computer interaction design process](image)

*Figure 2.1: Human computer interaction design process: requirements analysis, task analysis, global design and implementation, continuously evaluated.*

In this design method, the first step is requirements analysis. Out of this requirements analysis, a task analysis is constructed, which defines the functionalities. After the global design has been drawn, the final product, in this case a prototype, will be implemented and evaluated.

The implementation will be a prototype that shows the ideas as set out in this report. This prototype will be evaluated using the summative evaluation method, including a quick walk-through.
3 THE CURRENT SYSTEM

This chapter explains the original system in detail. A historical overview is given and the available worlds are summarized. After a quick walk-through to get familiarized with the system a breakdown is given of the different parts and layers of the system, both in hardware and software. The therapist’s user interface is treated in detail. The chapter ends with an analysis of advantages and disadvantages and a short view if the current system is good enough to continue working in the future.

3.1 HISTORY

The Delft Virtual Reality Exposure Therapy project was initiated as a collaboration project between the Delft University of Technology (faculty Information Technology and Systems) and the University of Amsterdam (faculty of Psychology) in the year 2000 with a pilot study of Schuemie (Schaemue, 2000b) to the possibilities of treating phobias with virtual reality environments instead of the commonly used graded exposure in vivo. Instead of taking a phobia patient to a frightening situation, that can hardly be reached or is costly to get there, the safe environment of the therapist’s office contains a digital setup for experiencing virtual environments, showing the situations the patient is afraid of.

The results of this pilot study were promising: even with the at that time low quality of the virtual worlds and the state-of-the-art hardware, the experienced fear and the ratio of patients cured approach the level that is achieved with in vivo therapy, while the main disadvantages - time and money - were gone. The road was paved to start the VRET-project.

Up to now different people worked on the VRET-project, starting with Schuemie who performed his PhD project on this topic. The results of this project (Schaemue, 2003), including both theoretical background research as practical research on user interface design, are still used as the main guidelines for follow-up research and implementation. Schuemie developed the main framework and several virtual worlds for treatment of acrophobia, claustrophobia and fear of flying. New virtual worlds with corresponding user interfaces are developed by Rahayu (Rahayu, 2003) (the Rotterdam underground, including stations and trains, for treatment of agoraphobia) and Hooplot (Hooplot, 2005) (the market square of Delft - the first project using another development environment than WorldUp - including weather effects and inter-person interaction, for treatment of agoraphobia; an example screenshot has been placed in figure 4.1); Gunawan (Gunawan, 2003) and Sopacua (Sopacua, 2004) extended and improved respectively the airplane world and the underground world. In 2006, two students of the Rotterdam University performed an internship. Aslan (Aslan, 2006a) migrated the airplane world to new computers and improved some elements as a first project, and continued his graduation project of the Rotterdam University of Applied Sciences at Delft University of Technology, by re-building the fear of flying application in Quest3D (Aslan, 2007). Bui (Bui, 2006) analysed the virtual reality application Quest3D in more depth and created a prototype for the airplane world. Several students are continuing the work on agoraphobia and other social phobias, both in psychological and technical aspects.

While at the start of the project the therapists that used the VRET application most were Emmelkamp and his students, performing research at the University of Amsterdam, gradually more people became aware of the existence of VRET and the promises it made. Schaemue and Krijn had close contacts with the VALK foundation, a foundation that aims at treatment of fear of flying. As from 2005, the PsyQ institute treats acrophobia patients using VRET, which proved very effective after some sixty patients treated with almost no drop outs. One year later, August 2006, the VALK foundation installed a running (but slightly adapted) version of the original airplane environment to use for their fear of flying therapy (Aslan, 2006a and 2006b).

While the interest of third parties grew, one of the suppliers spoiled the progress: in 2004, it became clear Sense8 discontinued development and support for WorldUp and WorldToolKit, the two software packages that are used for development of the virtual worlds. As the only running instances of this software are two computers of the year 2000 and new licenses for re-installs could not be achieved, it was obvious that we should look for a successor of this software. In Roorda’s research assignment (Roorda, 2005a) a comparison of products that are available is performed; this master thesis report is the first step in the implementation of the new development environment.

3.2 GENERAL ARCHITECTURE

The general functional architecture of the Delft VRET system can be summarized by figure 3.1 below (Van der Mast, 2006). Part of the therapy is the direct communication between therapist and patient. Currently most sessions take place in one room, so communication is direct (voice and visual); otherwise, an AV channel should be set up, using webcam, speakers and microphones.
All other communication, in the widest sense, flows through the VRET system. The therapist controls the system by performing his session, and evaluates the session by means of a.o. SUD’s. The VRET system immerses the patient by visual, audible and other stimuli, and fetches information about navigation, orientation, actions and other measurements, all of them together giving a good interpretation of the well-being of the patient.

### 3.3 THE VIRTUAL ENVIRONMENTS

The VRET system can handle different virtual environments for the treatment of the respective phobias. Table 3.1 shows the currently available worlds, their purpose and a short description. It is unlikely the number of worlds for the old system will be expanded due to the problems with Sense8 mentioned above.

<table>
<thead>
<tr>
<th>World</th>
<th>Phobia</th>
<th>Description</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutral</td>
<td>-</td>
<td>The neutral world is not meant to treat a phobia, but is used to familiarize the patient with the concept of virtual reality and get used to navigating and orientating in the virtual world. The world consists of an idyllic courtyard, partially outside and partially inside</td>
<td><img src="image" alt="Screenshot" /></td>
</tr>
</tbody>
</table>

Figure 3.1: The functional architecture of the Delft VRET system. The two-way communication of therapist and patient is direct if both are in the same room. An AV intercom connection is needed if both are not in the same room. The Delft VRET system is using two computers to be connected over the internet which supports tele-treatment.
<table>
<thead>
<tr>
<th>World</th>
<th>Phobia</th>
<th>Description</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firestairs</td>
<td>Acrophobia</td>
<td>A firestairs setting of a building in Amsterdam is digitally rebuilt. The therapist has the possibility to let the patient walk to stages at different heights, where the patient can look down at a view of the city below.</td>
<td><img src="image1.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>Construction</td>
<td>Acrophobia</td>
<td>Starting at a construction site, a firestairs can be climbed up to four levels high.</td>
<td><img src="image2.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>Rooftop</td>
<td>Acrophobia</td>
<td>Coming out of the elevator, the patient reaches the rooftop terrace, from where he can look down to the street.</td>
<td><img src="image3.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>Elevator</td>
<td>Claustrophobia</td>
<td>The patient goes into the elevator and chooses a level. Coming out of the elevator, he arrives in a hallway.</td>
<td><img src="image4.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>World</td>
<td>Phobia</td>
<td>Description</td>
<td>Screenshot</td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Closet</td>
<td>Claustrophobia</td>
<td>This world consists of a room in which a closet is placed. The patient can move into the closet. The therapist has the option to enlarge or reduce the size of both the room and the closet (independently of each other) in order to increase the level of fear.</td>
<td>![Closet Screenshot]</td>
</tr>
<tr>
<td>Corridor</td>
<td>Claustrophobia</td>
<td>This world consists of four connected corridors that become smaller each time the next one is entered through a door. The sound of closing doors and noisy, invisible people increase the level of fear.</td>
<td>![Corridor Screenshot]</td>
</tr>
<tr>
<td>Airport</td>
<td>Fear of flying</td>
<td>A complete airport is modelled in which the patient can navigate, from entering the airport via the check-in counters through the corridors up to the boarding gates. Sounds (announcements, airplanes) can be played to increase the level of presence.</td>
<td>![Airport Screenshot]</td>
</tr>
<tr>
<td>Airplane</td>
<td>Fear of flying</td>
<td>A complete flight can be simulated in this airplane, from take off to the landing. Announcements from pilot and cabin crew are present, as other sounds are. Even turbulence can be made very realistic by subwoofer speakers in the (real) airplane chairs.</td>
<td>![Airplane Screenshot]</td>
</tr>
</tbody>
</table>
Developing a general framework for the Delft VRET application

<table>
<thead>
<tr>
<th>World</th>
<th>Phobia</th>
<th>Description</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underground</td>
<td>Agoraphobia</td>
<td>The underground world simulates the Rotterdam underground including train and station. This project has been started before the discontinuation of WorldUp and at the moment of writing, the expectation is that this project will not be finished.</td>
<td><img src="image" alt="Screenshot" /></td>
</tr>
</tbody>
</table>

Table 3.1: Overview of the virtual environments and their phobia, description and example screenshot, that are currently available in the VRET system.

3.4 Quick walk-through

Before continuing about the details of the different layers of the system, a quick walk-through might provide a better understanding of the system. For this walk-through, I will use an example with the acrophobia world *Fire stairs*.

3.4.1 Preparing the session

Before the patient arrives, the therapist will ensure the VRET system has been set up completely, so the patient does not have to wait for this during the session. This includes connecting and turning on all hardware devices, turning on both computers, ensuring the Listener application is running on the patient’s computer and the network connection is working. For more information about the setup of the system, refer to (Roorda, 2005b).

3.4.2 Opening a virtual world

The patient enters the room, and the session will commence. Probably, the therapist will start with some things that are not directly VRET related: a little chitchat to relax the patient, outlining the session of today, and asking or answering questions.

When it is time for the virtual part of the session, the therapist looks at his computer desktop (an example of this is shown in figure 3.2), and finds between the categorized icons of virtual worlds, the *Fire stairs* world. A simple double-click will start the application. The patient still has not put on the head mounted display, for there is nothing to see yet!
3.4.3 Registering Data

The first step in the virtual session is the therapist entering patient number, therapist number and session number. These reference numbers will be used to collect and store data. Next, the therapist chooses Condition 2: user interface. Not because he wants this time a user interface, but merely because the other option is not working. A click on the Ok button will let the program continue.

![Figure 3.2: Example of a Windows desktop containing icons for the different virtual worlds.](image)

![Figure 3.3: A typical VRET session starts for the therapist with entering session specific data.](image)
In case a number that is already present in the system has been entered, an error message is shown, together with the request to enter a correct number. After all data has been validated, the main user interface shows up, and a trigger is sent to the patient’s computer to start the same world, with the same parameters. At the moment the network status at the bottom left of the screen tells you the computers are connected, the program has been started completely on the patient’s computer and the patient will be asked to put on the head mounted device.

3.4.4 PERFORMING THE SESSION

Maybe the therapist wants to get the patient used to the virtual environment and the way of navigating in the virtual world. He lets the patient look around, and maybe gives a simple assignment. Using the second monitor, he sees exactly what the patient sees. When the patient is immersed into the virtual world, the session can start ‘for real’: leading the patient up to the first level of the fire stairs, and look down; the second and third level exactly the same. At the top level, the patient can acclimatize a little, and is asked to look down once more.

At all stages, the therapist will examine the patient, sometimes asking what he feels, or which level of fear he feels. All answers are written down in the digital notebook, and put into the SUD (‘subjective unit of discomfort’) table. This data will be written to the database.

The patient is allowed to walk down again slowly. Arrived at the lowest level, he is asked to put off the head mounted device. The therapist closes the program (the patient’s computer will automatically follow this action), and after a quick analysis of the session, this session ends.

3.4.5 AFTER THE SESSION ENDS

At a later moment, the therapist can review the data stored during the session, by opening the database file. Don’t expect a user friendly report: the data is shown just as raw numbers. Analysing the database is mainly used for research purposes. (Aslan, 2006b) describes the database structure.
3.5 Setup of Hardware and Software

The current system

The main strength of the VRET system is the fact it runs on more or less common hard- and software. No expensive mainframe is needed, as the state-of-the-art (even in 2000) personal computers are powerful enough to render the virtual worlds used and to synchronize the worlds and the actions realtime. The operating system is Microsoft Windows and except for the VRET-specific programs no other specialized software is necessary. Even the devices that enable the virtual reality experience - the motion tracker and the head mounted device - are relatively simple and thus not too expensive, and are commonly available.

In figure 3.5 the setup of the different hardware and software layers is shown. These layers are overviewed in this paragraph, while the two main software parts (WorldUp and the Delphi user interface) are only quickly summarized, but will be described in more detail in the next paragraphs.

For the interconnection between the different hardware components, please take a look in appendix A, which shows a connection diagram.

![Diagram of hardware and software setup]

Basically, the setup is equal for both systems: the underlying operating system is Microsoft Windows, where a graphics driver is installed, outputting the data to the monitor or head mounted display. This data is provided by the WorldUp players, with additional functionality added in the Delphi scripts. The movements of the patient are recorded by the Flock of Birds motion tracker and captured by the WorldUp player. The communication flows over a TCP/IP connection.

A lot of information about the system setup and the working of the system can also be found in (Schuemie, 2002). However, that document aims at the end user rather than the developer.

3.5.1 Personal Computer and Operating System

Both the therapist’s and the patient’s computer are normal desktop personal computers. Where these had to be top-market during the start of the project, currently even budget systems are powerful enough to run the VRET application. The most important requirement, although even this has become the standard, is a graphics card that has dual monitor output (only necessary for the patient’s computer) and is capable of rendering 3D information in realtime.

The operating system running on the computers is Windows NT for the therapist’s computer, and...
Developing a general framework for the Delft VRET application

Windows 98 for the patient’s computer. The reason for the difference is the fact that WorldUp only has license keys for Windows NT, but Windows NT is not capable of supporting dual head graphic cards - Windows 98 indeed does support them. For two different operating systems are used, two different virtual world-players are required too; the therapist’s computer runs the Direct3D WorldUp player, the patient’s computer runs OpenGL.

### 3.5.2 Ascension Flock of Birds motion tracker

To trace the movements of the patient, both position and orientation, a motion tracker is necessary. With information from this device, the virtual worlds can be rendered as if the patient really is navigating through it and looking around.

![Figure 3.6: Ascension Flock of Birds motion tracker.](image)

The Flock of Birds motion tracker has six degrees of freedom which make it possible not only to turn your head to look around, but even to move a little distance. Three values for position (x, y, z) and three values for rotation (pitch, roll, yaw) are measured and sent as a matrix to the running program by a device driver. The tracker consists of a control box, a transmitter and a sensor.

The transmitter spreads a magnetic field in a sphere around itself, with a radius of approximately 1.30 meter. Using this magnetic field, the sensor determines its position and orientation with respect to a base plane (be careful not to cross this hemisphere divider, as unexpected results will occur; normally it will be no problem however, if the location of the transmitter is chosen well). All this information is gathered by the control box, and sent by a serial cable to the host computer. The range of the magnetic field is limited but sufficient for the setting used by VRET where the patient, if moving at all, only moves within a very limited area.

### 3.5.3 Cybermind Visette Pro head mounted display

![Figure 3.7: Cybermind Visette Pro head mounted display.](image)

The head mounted display is used to immerse the patient in the virtual world. Although there are other ways to create a immersive world such as a cave, a head mounted device is chosen for both its low price and the portability of it.

Specifically, the Visette Pro has been chosen for its wide field of view (about 70° diagonally) and quality of graphics (640 by 480 pixels). These two aspects contribute to the sense of reality the patient experiences. Disadvantages however are the relatively high costs of the head mounted device and the weight of it: after a short time of therapy, patients start complaining about the pressure of the helmet onto their heads.

### 3.5.4 WorldUp and WorldUp player, and the virtual worlds

Neither the therapist’s, nor the patient’s computer needs a full development version of WorldUp to be installed; this is only necessary for a system where one is developing new, or modifying existing worlds. The development version compiles a world into a single executable file, which contains all data for that respective world. These compiled worlds can be opened and run by the free players that Sense8 distributes without the need of the development version. The versions of this player used by VRET are an OpenGL player for the patient’s computer and a Direct3D player for the therapist’s computer; the choice for this depends on the underlying operating system as stated earlier in this paragraph, and the players do not very differ in functionality. The latest updated versions are from 2002 and 2004 respectively and are suitable for Windows 98 and Windows NT.

The virtual worlds are constructed using the modeller of WorldUp. WorldUp is also used to add movements, events and interactivity to the world. Navigation and orientation is added using the WorldToolKit API, a transparent layer that can communicate with a large range of VR-devices. Adding advanced elements in a world is done using a Visual Basic-like scripting language.

When a virtual world is ready, it can be compiled into an executable file that contains all necessary data. This file can be read and run by the free players. Players are available for both the OpenGL and
The current system

the Direct3D platform, but their functionality is similar.
As any other virtual reality developing environment, WorldUp takes a while to learn and to get used to.
Regarding students and graduates that worked with WorldUp, the program is complete in its
functionality but is too unstable: worlds changing without reason the next time you open them and
crashing the system are two examples that have frustrated people many times.
Maybe the lack of further development was the first sign of the discontinuation of this product line of
Sense8. At this moment, Sense8 is no longer available for requests of any form. Due to this, instances of
WorldUp or WorldToolkit can no longer be installed on new machines, therefore pausing development.
Another consequence is a really outdated player program.
In paragraph 3.3, the virtual worlds have been discussed in more detail.

3.5.5 Delphi user interface
An application written in Delphi 5.0 is used on both computers to control the rendering of the virtual
worlds. The patient’s computer has no visible user interface but several actions are handled rather by
Delphi than by WorldUp. The therapist’s computer clearly has a user interface; with this user interface
the therapist is able to control the therapy used to treat a patient. A lot of research has been
performed after the separate elements of this by Schuemie (Schuemie, 2003) in his thesis: usability,
effectiveness and efficiency are a few of the aspects he studied. This user interface is still used and
even further developed by different students as described in the history section above. Most parts of it
will be included in the new system again.
In paragraph 3.6.3 the Delphi user interface will be discussed in more detail.

3.5.6 Communication and network connections
For a therapy to succeed it is necessary that all relevant data is sent from therapist to patient, but also
from patient to therapist. For this communication a TCP/IP-connection is used.
The first data to be exchanged is a trigger to open a world: the patient should not be concerned with
these non-functional actions, so the patient’s computer has a little program running that listens to the
therapist’s computer: as soon as the therapist opens a virtual world, it sends a command to the
patient’s computer to open the same world. This program is called the listener application and is
discussed in more detail in paragraph 3.6.2.
After the world has been started on both computers, synchronization data is sent both ways:
orientation and position data coming from the motion tracker is sent to the therapist, so the exact
patient’s views can be re-drawn, and actions initiated by the therapist, like changing parameters, are
sent to the running world on the patient’s computer.

3.6 The framework
WorldUp and WorldToolkit are powerful applications that, especially when using the scripting language,
allow more complex applications to be written, but for more complicated programming that enables
development of a graphical user interface and connections to for example a database, a more high-
level programming language is recommended. A Delphi 5.0 application has been developed as a skin
around the VRET virtual worlds and this application runs these worlds and coordinates the interaction
between them.
There are three different Delphi applications used within the VRET system: a listener program to
synchronize the starting of worlds on the patient’s computer; a user interface for the therapist’s
computer with connections to WorldUp and the patient’s computer; and a non-visible skin to coordinate
actions done on the patient’s computer. In the next paragraphs these programs will be discussed in
more detail.

3.6.1 Phobia.ini
Both computers need a configuration file, that contains some variables, like the location of programs
and virtual worlds, and network information. The variables to be initialized are listed in table 3.2,
which also indicates whether the variable is required for the therapist’s computer, the patient’s
computer or both computers. Comments can be added in the file by starting a line with a # character.
Developing a general framework for the Delft VRET application

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Therapist’s computer?</th>
<th>Patient’s computer?</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThisCompu</td>
<td>+</td>
<td>+</td>
<td>Console VRstation</td>
<td>Indicates whether ‘this’ computer is the therapist’s computer (Console), or the patient’s computer (VRstation)</td>
</tr>
<tr>
<td>DLLdir</td>
<td>+</td>
<td>+</td>
<td>Directory</td>
<td>Full directory path that contains the DLL files</td>
</tr>
<tr>
<td>VEdir</td>
<td>+</td>
<td>+</td>
<td>Directory</td>
<td>Full directory path that contains the virtual world files</td>
</tr>
<tr>
<td>WUP</td>
<td>+</td>
<td>+</td>
<td>File location</td>
<td>Full directory path and filename of the WorldUp player to use</td>
</tr>
<tr>
<td>IPconsole</td>
<td>+</td>
<td>+</td>
<td>IP address</td>
<td>IP address of the therapist’s computer</td>
</tr>
<tr>
<td>PortConsole</td>
<td>+</td>
<td>+</td>
<td>Integer</td>
<td>Port number on the therapist’s computer that will be opened for listening</td>
</tr>
<tr>
<td>IPVRstation</td>
<td>+</td>
<td>+</td>
<td>IP address</td>
<td>IP address of the patient’s computer</td>
</tr>
<tr>
<td>PortVRstation</td>
<td>+</td>
<td>+</td>
<td>Integer</td>
<td>Port number on the patient’s computer that will be opened for listening</td>
</tr>
<tr>
<td>VEfilenamexx</td>
<td>+</td>
<td>+</td>
<td>File location</td>
<td>Directory paths relative to the directory set in the variable VEdir and filenames of the virtual worlds; the numbers (xx) are related to the numbers used in the WorldUp scripts</td>
</tr>
<tr>
<td>NumberofBirds</td>
<td>+</td>
<td></td>
<td>Integer</td>
<td>Number of motion trackers connected to the patient’s computer</td>
</tr>
</tbody>
</table>

Table 3.2: Overview of all variables to be initialized in the VRET configuration file phobia.ini.

An example of the phobia.ini file on the therapist’s computer:

```
# File initialization
ThisCompu = Console
DLLdir = c:\DLLs\
VEdir = d:\Worlds\
WUP = C:\WuPlayD3D.exe

# Network initialization
IPconsole = 130.161.157.192
PortConsole = 999
IPVRstation = 130.161.157.198
PortVRstation = 998

# Virtual worlds
VEfilename02 = Hoogtevrees\Brandtrap\ophoogtevreesv09.wup
VEfilename03 = Hoogtevrees\Dakterras\hduni06.wup
VEfilename04 = Hoogtevrees\Bouwplaats\hbouwplaatsuni05.wup
VEfilename05 = Neutral\Binnenplaats\Nbinnenplaats04.wup
VEfilename06 = Claustrofobie\Gang\UniGang2 v05.wup
VEfilename07 = Claustrofobie\Kast\ckast2uni02.wup
VEfilename08 = Claustrofobie\Lift\cliifun104.wup
```

An example of the phobia.ini file on the patient’s computer:

```
# File initialization
ThisCompu = VRstation
DLLdir = c:\DLLs\
VEdir = D:\Worlds\
WUP = C:\WuPlayD3D.exe

# Network initialization
IPconsole = 130.161.156.156
PortConsole = 999
IPVRstation = 130.161.157.198
PortVRstation = 998

# Number of motion trackers
NumberofBirds = 2
```

3.6.2 Listener

A little standalone program is called the Listener: this program is run as background program on the patient’s computer, and it does exactly what its name suggests: it listens to incoming commands from the therapist’s computer. If the therapist starts a session, a command containing the data necessary to
The current system

open the same world is sent to the patient’s computer. The listener catches this command and performs the required actions: starting the WorldUp Player, opening the correct world, and initializing the network connections. After initialization, the virtual world application takes over all responsibilities, including sending and receiving network data: the listener finished its job.

The reason for using this listener application is minimizing the interruption during the therapy: the patient is concentrated and should not be disturbed by a therapist opening a program on his computer.

3.6.3 Therapist’s user interface

Program flow

One would expect the graphical user interface is the main program, which starts the required virtual world when requested - the main program that is used by the therapist as a parent of its children, the virtual worlds. But, due to the structure and limitations of WorldUp, the real relationship is different: the graphical user interface is started as a child, a subprocess, of the virtual world. For this reason the technical structure of the graphical user interface is a dynamic linked library (DLL) file rather than a standalone executable program. Instead of having a general ‘VRET application’ that manages all virtual worlds, the therapist has a lot of virtual worlds to manage. To make the program flow more clear, refer to the flowchart diagram in figure 3.8.

![Flowchart diagram of the initialization of a new session in the current system.](image)

Figure 3.8: Flowchart diagram of the initialization of a new session in the current system. After opening a virtual world, the DLL file is included and initialized; a notification is sent to the patient’s computer, where the same virtual world is loaded, also including the DLL file. At last, the network connection is established.

The first action is opening a virtual world (1) at the therapist’s computer. The WorldUp program includes (2) the dynamic linked library that contains the graphical user interface for the therapist and after inclusion calls the initialization function (3); this function reads and sets some parameters and notifies the listener program of a new world being opened (4). The listener application starts the WorldUp Player with the corresponding virtual world (5), which in turn includes (6) the dynamic linked library containing the user interface for the patient and also calls the initialization function (7). After both virtual worlds have been started correctly, a TCP/IP-connection - which sends synchronization data ten times a second - is established (8) between the DLL files on the therapist’s and patient’s computer, and the therapy can commence.

Interface elements

The main user interface for all therapies, of which an example has been placed in figure 3.10, is similar: the therapist has to be able to control the environment, has to see the world as the patient sees it, and should be able to create notes and remarks and process other feedback from the patient. The main elements in the user interface always are equal, and only the world-specific elements like the environment controls change. This functionality is achieved by placing layers on top of each other and only show that layer that belongs to the chosen world; see for a clarification figure 3.9.
Although the idea of a standardized user interface is perfect – a therapist that is known to one world, can easily switch to another world – there were a few drawbacks discovered by practice. The layout is too static, especially regarding the size, causing complex worlds not to fit within the limited space of control elements. An example of this is the Airplane world: Gunawan (Gunawan, 2003) extended the functionality of the airplane world and found too many new control elements to fit into the space available in the former implementation. Another drawback has a more technical nature: each new world has to be programmed into the framework source code, implying also a recompilation of the program.

All this resulted in three slightly different versions of the framework application: the original PhobiaServer.dll and a copy that, as far as I could investigate, is exactly similar to this file, called PhobiaServerNew.dll. I suppose this file was created by one of the students working on the application in order not to overwrite the original source. Gunawan’s work resulted in AirplanePhobiaServer.dll.

In the description of the user interface elements, I refer to the original user interface. The Airplane world user interface is quite similar and can be projected to this description, too. The numbers refer to the numbers in figure 3.10.

Figure 3.9: Each virtual world has a different set of control elements, placed in a single layer for each world. All these layers are put in the same position in the user interface form; the code makes only the required layer visible.
1. **Patient’s viewpoint**

The patient’s viewpoint is an exact copy of the world the patient sees. All parameters are equal, as is the viewport and viewing direction. This viewpoint is one of the most important elements of the user interface, for it is necessary to be able to see what the patient is experiencing. Another way to copy the patient’s viewpoint is using an additional monitor that duplicates the video stream sent to the head mounted device. The Cybermind Visette Pro supports this by adding a “VGA out” on the control box, that can be connected to a monitor. The view of the left eye is copied in this way.

2. **Free viewpoint**

Schuemie added, besides the copy of the patient’s viewpoint, a viewpoint in which the therapist himself can navigate. The viewpoint can be locked to the patient’s position, or be a completely free one. Reasons for adding this free viewpoint are twofold: aligning the world when working with the railing that surrounds the patient, and adding a bird eye’s view to the world. Schuemie experimented (Schuemie, 2003) with this viewpoint, and the results are that the free viewpoint is perfect for exact positioning and thus aligning the two reference points (the railing in the virtual world and the railing in the real world), but less efficient for navigation - using the patient’s viewpoint resulted in quicker and shorter navigation paths. A further detail in Schuemie’s conclusion is that experienced users will be able to combine both viewpoints in an optimal fashion.

These conclusions are proved by the experiences of therapists in real treatments: most of them do not use this free viewpoint, but the few of them that do, are very positive about it.

3. **Map**

Especially as the virtual worlds become more complex or larger, a map that shows the position of the patient in the world can prove very useful. Each virtual world has a single BMP image file which is a simple map of the world, and each world also has a few lines of code in the file Translator.pas. This code translates world coordinates to a coordinate in the bitmap image - at this location, a dot is projected.

---

*Figure 3.10: Example of the therapist’s user interface, showing the ‘closet’ virtual world.*
4. **CONTROL ELEMENTS**
All virtual environments have certain parameters that can be changed to increase or decrease the patient's fear. Examples of these are the level to stop the elevator at, closing or opening doors in the closet-world, calls in the airport or turbulence in the airplane.

The controls section of the user interface is different for each world (if available), which is implemented by using a different layer for the different worlds. Depending on the world chosen, the correct layer is shown and all the other layers are hidden. This implies that for each new world or each change in controls the complete DLL-file should be recompiled.

To send commands to the WorldUp player, all parameters that can be initiated through the control elements are stored into a data array and sent on a regular base (ten times a second).

5. **AUTOMATIC PILOT**
On the fly navigation of the patient is done using the joystick. This requires a lot of experience and concentration, and for most therapies the therapist has a sort of preplanned route in the world that he would like to use. These routes have been implemented in the automatic pilot. The therapist can send the patient to the right place, being able to adjust the speed of movement. The initial version of Schuemie has an option to transport the patient without delay to a specified location, but this functionality is not available in the used DLL-files. The main reason for building the automatic pilot is removing tasks from the therapist, who has more time for the more important things of the therapy and doesn't need to hassle with navigation.

Another reason for using the automatic pilot is the increment of fear: when the patient is forced to move in a constant way, his SUD will increase. As a possible explanation is given (Schuemie, 2003) that the patient can not avoid fearful situations when guided by the therapist.

6. **FILE**
The file section offers the therapist a digital notebook to write down comments and remarks. All data of one session is saved into a single text field, so they cannot be linked automatically to a certain moment in the session.

7. **SUD RECORDING**
An important part of a session is determining the level of fear the patient experiences. With certain time intervals the patient is asked to express his level of fear on a scale of 0 to 10. The SUD recording section can be used to enter these data into the system and shows the results in either a table or a graph. Using this information the therapist can examine if the patient progresses and the therapy is succeeding.

8. **ALARM AND CLOCK**
The clock section shows the total session time and offers the possibility to pause the session for a while. The alarm section can be used by the therapist to be notified after the set amount of time, for example to go to a next phase of the therapy. The alarm can be turned off by setting the time interval to 0.

9. **SYSTEM DATA**
The system data block shows the running virtual world and the status of the network connection. It is primarily used for debugging unexpected errors, like a missing network connection.

3.6.4 **PATIENT'S USER INTERFACE**
Just like on the therapist's computer, as soon as a WorldUp file is opened on the patient's computer (upon an incoming trigger), it includes the DLL-file that has been written for this computer. No user interface is present in this DLL file (the only output is that of the WorldUp player to the screen - or, in fact, the head mounted display).

The functionality of the DLL file is sending and receiving data over the network connection. It provides two functions to transmit and receive coordinates, joystick parameters and additional variables to and from the therapist's computer.

3.7 **EVALUATION**
Different students reviewed the current VRET system, either for adding new worlds to the system or changing existing worlds. Another level in analysing and evaluating the code is performing research to
the possibility of migrating the existing system to more recent hardware and software. This has been the main focus of Roorda's research assignment (Roorda, 2005a) and Aslan's project (Aslan, 2006a).

In this section I will describe the positive and negative points found by the different students, and I also take into account comments of therapists about their experiences with the VRET system. These two sources (students and therapists) overlap each other, and the features can not be distinguished easily between students or therapists.

3.7.1 **Positive points**

The functionalities below are experienced in a positive way by developers and/or therapists, and should certainly be implemented in the new version of the VRET system:

- The initial idea and implementation of one basic user interface for all worlds has proved to shorten the time needed for learning to use the interface. An advantage for the developer is that no new user interface has to be developed and implemented, and all focus can be upon development and integration of the virtual world.

- Two separate viewpoints, one exact copy of the patient’s view and a freely navigable viewpoint, offer the therapist a better orientation of the patient in the virtual world. The projection map can even be seen as a third viewpoint.

- Most other user interface elements have proved to be very useful for the therapist: the SUD and comments sections make recording of the session possible, the automatic pilot decreases the press upon the therapist in navigating the patient, and the alarm clock notifies the therapist when the next phase of the therapy should start or a specific action has to be performed. The system data part is mainly used to check the network availability, and could be replaced by something less technical.

3.7.2 **Negative points**

The negative points mainly rose from the view of development: apart from some tiny comments, the therapists are content with the system as it is now. The following points should be prevented and taken care of in the new version of the VRET system. Comments about missing elements and functionality are not taken into account here, but are included in the requirements section of the next chapter. Where a short hint towards a solution is possible, it is added.

- Each new world or adaptation to an existing world (i.e., the user interface for it) requires a new compilation of the complete user interface file – including all code for all other worlds. There is always the danger something is adapted that should not be changed, causing the new world to work perfectly, but existing worlds giving problems. This problem can be solved by developing in a modular way, creating a general framework, linking to application-specific include files.

- The graphical user interface is put into the front of all other running programs, preventing other windows to overlap it. To prevent the WorldUp Player windows to be overlapped by the user interface, a third party class is used to create two transparent ‘holes’ in the user interface, exactly at the position of the WorldUp viewports. This class is called CoolForm and contains code that digs deeply into the system functions. This makes the class system-dependent, and results into program failures when running the application on another computer than the one it was compiled on. For each new installation of the VRET system, the PhobiaServer class has to be recompiled.

- Another problem rising from the fact that the graphical user interface exists of three different program windows (Delphi user interface and two WorldUp Player windows) is that the position of everything on the screen is fixed: resizing the window is not possible, nor is moving. This can be solved by triggering the WorldUp windows upon resize or movement of the main application window.

- Overall, the system is quite unstable by times. Problems during development (sudden crashes combined with loss of work) and running the application (crashes, difficulties when migrating to another computer) are reported by developers and users more than once.

3.8 **Further development**

While more and more companies and institutes show interest in the virtual reality treatments of VRET, it is becoming ever more important to be able to continue development. For several reasons it is not
possible to keep developing on the current system. Of course the discontinuation of the graphical software of Sense8 is an important factor, but also the above mentioned difficulties with development and management of the framework turns against future expansion.

In the papers of Roorda (Roorda, 2005b) and Aslan (Aslan, 2006a) two attempts are described to migrate the original VRET system to a new hardware environment: newer and faster computers with Windows XP. The results of these studies were similar: it is possible to a certain extend, but takes a long time and requires a thorough knowledge of the way of working of WorldUp and the Delphi-code of the framework.

In the next three paragraphs I will review the possibilities of further usage and development when sticking to the current system. Both the graphical software and the framework and the hardware are discussed.

### 3.8.1 Graphics and VR
The worlds that are available now can be run by the latest versions of the WorldUp players. Maybe a little ‘code-hacking’ is needed to prevent some unwanted behaviour (an example as found in (Roorda, 2005b) is the calibration of the joystick: some threshold values must be adapted in the WorldUp scripts for the joystick to work in the right way due to driver differences between Windows 98 and Windows XP), but up to now (tested up to Windows XP) no real problems with the WorldUp software are discovered.

Development of new worlds or even adaptation of the existing ones is not possible. The WorldUp development environment is installed on only a few computers and these ones are only meant as a backup of the complete system. New installations cannot be activated.

### 3.8.2 Framework
Where the worlds can mostly be run without much troubles, the user interface is another case. The use of the CoolForm class is very system dependent, and each new computer on which VRET should run forces a recompilation of the DLL-file. Sometimes other units should be installed with Delphi for the compilation to succeed.

Other than with the worlds and despite the recompilation problems, it is possible to expand the number of worlds the GUI can handle, just by adding some basic functionality.

### 3.8.3 Hardware
The currently used hardware is, even after six years, of very high quality. Maybe it is worthwhile looking for a more light head mounted display, as the Visette Pro is a little heavy, causing patients to become tired after a short time of therapy. As the system itself, most commonly available desktop computers can be used without problems.
4 THE NEW FRAMEWORK

This chapter describes the new framework and all its related parts in detail. After the requirements listing, the graphical part is handled in the form of a modelling application and the virtual reality application. Next, the framework itself is explained in all its aspects.

4.1 REQUIREMENTS

Requirements for the new framework arise from several aspects. A lot of requirements from the current system remain relevant, so I will not repeat these. But the current system arose new requirements, both in functional way and in technical way. In this section I split them out.

The requirements are collected via own experiences, colleagues working or having worked on the VRET application, interviews done with therapists (mainly Paul Emmelkamp, Jenneke Wiersma and Merel Krijn) and of course the assignment description. The functional requirements mostly arose from the therapist’s input, the technical requirements from the other’s.

4.1.1 TECHNICAL REQUIREMENTS

CONTINUATION
One of the strongest requirements is continuation, which is the main reason for this thesis project. Over a longer period of time the VRET system should be usable, maintainable and extensible. Usable on ever-changing hardware and operating systems: the average computer system doesn’t last longer than a few years, neither does an operating system including drivers and other dependent applications. Maintainability and extensibility are two subjects that overlap each other: no software is perfect, and no software will be fully complete, so developers should be able to view the source code, make changes and recompile the application, and aiming on the grow of the VRET project, new functionalities and virtual worlds need to be implemented and connected.

MODULARITY
The last two points mentioned above, maintainability and extensibility, imply a modular structure, that makes it possible to change or add parts without touching other parts and especially without touching the main core of the system.

SECURITY
All data stored in the VRET application is very personal, mainly medical based. This fact makes security an important aspect: not only the use of the application should be restricted (only the authorized and trained therapists may use the program), but the data itself should be safe too: therapists can only see and modify their own patient data, and the data must not be accessible from outside the VRET application. A system administrator that can grant or deny access to (parts of) the system must be chosen.

The VRET application is an online application, in which a network or internet connection is established between therapist and patient. This connection should be secured, too, to prevent eavesdropping.

LOGGING
Sessions should be fully logged. At this moment only SUD-information and notes are logged and stored into a database, but the complete session should be replayable, including movements, actions and comments. All data must be easily retrievable again from within the VRET application.

4.1.2 FUNCTIONAL REQUIREMENTS

ONLY ONE APPLICATION
At this moment, the VRET application is part of the virtual worlds instead of vice versa. This means the number of worlds determines the number of applications that can be started by the therapist. The new setup should contain only one application that is able to include and start different worlds. This application also contains all other aspects of the VRET system, like configuration, logging and retrieving stored data.

PATIENT MANAGEMENT
Following the previous point, patient management should be available. In the current application the therapist has to enter the patient’s number into the system, which is erroneous, unclear and requires
Developing a general framework for the Delft VRET application

An additional action from the therapist. To make a more complete application, it should be possible to manage patient data. In the future, an automated import and export functionality to the main patient database can easily be implemented.

**Flexible User Interface**

At this moment, the user interface is fixed with respect to position, size and lay-out. Even with the currently available worlds, this proved to be a problem - there was a good reason to re-write the user interface for the airplane world: there was not enough space to put all controls. The more the VRET application will be extended, the more this problem blocks development.

With more therapists working with the application, the wishes for a customizable user interface grow, for example hiding or re-arranging interface elements.

The new application should provide a fully configurable user interface, which enables showing and hiding different functionalities and moving them around. In this way, the most often used elements can be grouped together, for example. Each world has a default user interface setting, and newly configured settings can be saved per therapist and per world.

### 4.2 3D Modeller

As investigated by Roorda (Roorda, 2005a), there are only few virtual reality application development tools that include a modelling application. While a built-in modeller has the advantage that objects can be defined, stored and used in a most optimized way, the risk is dependency on that specific application. As soon as the software is no longer used, either because another application comes into view, or stopped support and development by the manufacturer, the objects and models will no longer be useful. Even when the software provides export functionalities, there will be tiny differences and data loss compared with the original objects, and the advantage of optimization is gone.

For these reasons I recommend using a separate modelling application, especially one that is commonly supported. Among the best known and most widely spread and supported applications are Maya and 3D Studio Max. The wide range of usage does not guarantee infinite continuation, but makes it less likely the product is stopped at once. Also a lot of people master these programs and documentation is available for all levels of professionalism, making maintenance more reliable and maybe even cheaper.

Furthermore, using separate, external objects will make the VRET application more modular and a future change of virtual reality development environment will be a lot easier than it currently is. The final choice for the modelling application still has to be made, but few studies have been done with (re-)building virtual worlds in Maya.

### 4.3 Virtual Reality Application

Before starting this master thesis project, I studied the question which virtual reality development application to use. The whole report can be found in (Roorda, 2005a), which is included in appendix D; here I will summarize the main outcomes of it.

Out of an initially larger number of applications, eight programs have been studied on seven criteria to be able to compare them and rate them accordingly for the use within the VRET application. The programs have been rated on the following technical and non-technical aspects:

- Supported hardware (mainly motion tracker and head mounted device);
- Can the idea of the VRET application be implemented; is code re-use even possible?
- Internal or external modeller?
- Importing and exporting features, especially existing VRET files;
- Pricing policy;
- Licensing conditions;
- Continuation guarantee.

All these aspects were rated onto a scale from 1 to 5 (best). The final conclusion leads to table 4.1. The biggest differences strike in the licensing conditions and continuation precautions, which was one of the most important aspects. Also pricing and the options for conversion are spread over a wide range. The development functionality itself was never a problem, whereas the hardware was also very commonly supported.
Table 4.1: Comparison of eight virtual reality development applications on several subjects, in order to be able to choose the application used for further development of VRET.

<table>
<thead>
<tr>
<th></th>
<th>CAVElib</th>
<th>CaveUT</th>
<th>EON Reality</th>
<th>MetaVR</th>
<th>MultiGen</th>
<th>Quest3D</th>
<th>Virtools</th>
<th>Vizard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware support</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Software development</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Scene development</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Conversion and import</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Price</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Licensing conditions</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Continuation</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>21</td>
<td>29</td>
<td>19</td>
<td>19</td>
<td>23</td>
<td>31</td>
<td>29</td>
<td>31</td>
</tr>
</tbody>
</table>

Of the eight fully studied applications, three candidates showed to be good ones to use in the further development of VRET: Act3D’s Quest3D, Virtools, and WorldViz’s Vizard. All of these applications have the required basic functionality, but on certain criteria they differ from each other.

Despite of the conclusions in my research assignment - where Quest3D was chosen - I will compare Quest3D and Vizard once again. The main reason for this is practical experience by some students actually working with Quest3D: integration within a larger application proved to be difficult and the way of programming is characterized by a steep learning curve. Virtools has already been dropped earlier for its strict and expensive licensing conditions. I reviewed my time schedule and added a few days to get practical experience with both applications to be able to make a more founded choice.

4.3.1 QUEST3D

**Background**

Quest3D is a product developed by the Dutch company Act-3D (Quest3D, internet). In 1998, it was developed as an in-house production tool for enlightening the production process of the different projects. When customers saw this tool, they were impressed, and the idea grew that Quest3D could become a well-sold application. In 2000, the first commercial version of the program was released. Regarding the reactions on internet fora and Act-3D’s own website, the quality of the program, compared to its price, was of a high level. In 2005, the product line was for the first time extended with a virtual reality edition, which is used here. In this evaluation, I will, besides own experiences, use some outcomes of Hooplot (Hooplot, 2005) and Aslan (Aslan, 2006a and 2007).
Developing a general framework for the Delft VRET application

EVALUATION

PROGRAMMING STYLE

Starting the program, the first thing that strikes is the way of programming: this is completely different compared to most other programming languages or 3D environments (Quest3D, internet). Literally all implementation has to be created using predefined building blocks.

While Quest3D has basic primitives and other objects built-in, creation of a real 3D environment should be done using an external modeller. Afterwards, the properties of the object (e.g. size, position, texture and material) can be modified. Quest3D includes plug-in modules for most general modellers, including Maya and 3D Studio Max. Besides the objects, also light sources and cameras should be placed in the virtual world.

The next step is defining all dynamics, dependencies and interaction between elements within the virtual world, and between the virtual world and the user. An example of the first type is moving characters that shouldn’t collide, an example of the latter is navigation or head tracking. Quest3D works with building blocks, called channels, rather than plain code, and this is one of the main reasons the learning curve is very steep: it takes a long time to learn the basics, even when following the introductory tutorials. Figure 4.3 shows a simple if-then-else statement to illustrate this different implementation method. The channels Action true and Action false are again the start of a new web of interconnected channels to perform the right actions.
The new framework

Quest3D provides ways to arrange the mess of channels and the links between them, but far too soon your screen becomes full of squares and lines and it is difficult to keep structure in it as the images below show - the term *spaghetti code* might come in mind! Below a series of screen shots showing a specific Quest3D application (the market place of Delft, used for treating agoraphobia patients (Hooplot, 2005)), ranging from fully zoomed in (detail view), to fully zoomed out (structure view).

Apart from the channels provided by default, it is possible to create your own channels. The easiest way to do so is by combining a number of channels and treat it as one new channel. But this way, you are bound to the limitations that the existing channels know. A more sophisticated way of creating your own channels is using the C++ API, which makes all C++ functionality available - which is only limited by the skills of the programmer. Quest3D provides some examples for this way of creating new channels and states its requirements with respect to input and output format.

**Program Flow**

Each Quest3D program has a so-called ‘Start channel’. This channel is the root to which all other channels are linked, directly or indirectly. Simply said, the program flow consists of looping recursively through all linked channels, from left to right (visually spoken). After the last action is performed, the program begins from the first channel again.

An example is shown in figure 4.5: taking the Start channel as starting point, the first thing done is calling the Render channel, which in turn defines and calculates camera, light and 3D environment. After having rendered completely, the Check input channel reads from an input device if a signal is available (and if so, call the underlying channel, which is not shown here). The last step in this loop is writing some data to a database in the last channel. After having completed this step, the loop rewinds and the Render channel is called again.
Developing a general framework for the Delft VRET application

The number and complexity of channels influences the speed of the program: complex programs will have a lower frame rate. This implies the speed of Quest3D applications depends fully on the used computer system. Another very depending factor is the screen resolution - the lower the resolution, the faster the rendering. Without additional programming, it is not possible to express movements in a relative way, like 'move 20 meters during the next 5 seconds'. The only way to avoid this unwanted behaviour is to manually calculate the positions using the true frame rate using the microseconds-value of the system clock. More disturbing are relative movements, like moving legs under a walking person: the legs move too fast or too slow related to the forward moving speed.

USER INTERFACE AND INTEGRATION
One of the more important requirements is integration within a fully functional user interface. The only way in which a user interface can be implemented in Quest3D is placing user interface elements (buttons, labels, ...) into the 3D world and looking at them through an additional camera; events are triggered by continually checking the mouse position (does it collide with the object) and the mouse trigger (is the mouse button pressed?). Another option is to use only key short cuts, of which an example is placed in figure 4.6.

Besides difficulties in implementation and re-usability, the user interface is always part of the virtual world. Connection to a separately running user interface (compare the VRET system with the Delphi user interface) is allowed only by use of own channels.

3D OBJECTS – IMPORTING AND HANDLING
Quest3D ships with exporters for (among others) 3D Studio Max, Maya and Lightwave; native support for the file format conversion program Polytrans is also included. Objects are imported as separate channels, and can in that way be addressed by the other channels in order to add interactivity.

Figure 4.5: Development example of Quest3D: program flow of a simple Quest3D program.
4.3.2 Vizard

Where in Quest3D the imposing graphical user interface directly jumps into view, Vizard looks like a simple, even Spartan, command line tool. It is true most actions in Vizard are programmed in the Python programming language; only a few tasks can be performed visually, including the import of objects and setting global parameters. But exactly this non-visual programming makes it far more easily accessible, especially for people that are used to programming.

A C++ version of Vizard was said to be under development, but during the time it took to finish this project, it remained in a beta stadium. For that reason I have chosen the Python version, and I will refer to this version throughout this report.
Developing a general framework for the Delft VRET application

EVALUATION

PROGRAMMING STYLE

Programming is done using the Python programming language that can be learned very quickly if some basic knowledge about programming is present. Python is an object oriented programming language that has a similar programming syntax as for example C++ and PHP. The main components are the Vizard classes, that range from basic classes (the viz class) to classes that enable high-level programming (e.g. the use of tracking devices), but also the basic Python classes and all own or third party classes may be used.

Each Vizard program starts with including the viz class, and optionally additional classes for more functionalities. General components of the program are importing objects, setting viewpoint parameters and adding interactivity (mouse and keyboard).

The most simple Vizard program can be implemented as follows:

```python
import viz
viz.go()
world = viz.add('world.wrl')
```

These only three lines import the main viz class, start the render loop, and add a world, described in a .WRL-file, to the scene. Running the program, it is possible to navigate through the world using mouse or keyboard.

Adding user interactivity by keyboard is done in another few lines:

```python
viz.callback(viz.KEYBOARD_EVENT, handleKeyboard)

def handleKeyboard(key):
    if key == 'a':
        do.action1()
    if key == 'b':
        do.action2()
```

Figure 4.7: Vizard - a typical work-in-progress screen, showing the user interface with a summary of general settings and a raw preview window (1), the application code (2), the output window (3) and the debugging messages (4).
Animation can be added by using timers. Like the way of catching keyboard events, a timer is initiated by a callback function. Multiple timers can be added in a Vizard program.

```python
callback(viz.TIMER_EVENT, handleTimer)
starttimer(1, 0.5, viz.FOREVER)
starttimer(2, 0.5, viz.FOREVER)
```

```python
def handleTimer(id):
    if id == 1:
        do.timerEvent1()
    if id == 2:
        do.timerEvent2()
```

A Vizard installation includes a default network handling module, that can be implemented in quite the same way as keyboard, mouse and timer events. When multiple network connections are required (as is the case in the VRET application), one must fall back to the basic Python commands and functions in combination with a timer event:

```python
timer_id_network = 1

def ReceiveData():
    try:
        return AddSocket.recv(1024)
    except:
        pass

def handleTimer(num):
    if id == timer_id_network:
        data = ReceiveData()
        if data:
            handleData()

callback(viz.TIMER_EVENT, handleTimer)
starttimer(timer_id_socket, 0, viz.FOREVER)
```

3D objects - importing and handling

Vizard can handle a lot of different 3D formats, including 3D Studio Max, Maya and WRL files; WRL files are used in all tutorials and also for the prototype belonging to this research. WRL files or parts of a WRL file can be addressed directly by Vizard to add interactivity.

4.3.3 Which application to choose?

Quest3D and Vizard are two applications that are very different in all aspects. Both applications offer a lot of functionality that is accessible in a more or less easy way. In fact, ‘everything’ will be possible with both applications, just if you know how to achieve it. Vizard is in this aspect more intuitive, as you stay in the same programming context (Python). With Quest3D, you will have to switch from the channel based style to C++ programming.

The final conclusion of the comparison report (Roorda, 2005a) was to choose Quest3D, for in theory this application has the best papers to be the new development environment. Frans Hooplot started building his agoraphobia world into this program (Hooplot, 2005). His practical experiences made me to re-look after my choice for the VR application: as described above, Quest3D has a very specific way of working, that is not the most convenient. Table 4.2 sums the aspects that struck me most.

<table>
<thead>
<tr>
<th>Quest3D</th>
<th>Vizard</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming style</td>
<td>Python-based; imperative programming.</td>
</tr>
<tr>
<td>Channel-based; interconnecting pre-programmed building blocks.</td>
<td>As a result of channel-based programming, a simple program already exists of a lot of channels, including connections between them. This causes a confusing mess of visual elements if you are not very careful.</td>
</tr>
<tr>
<td>The tutorial provides nice examples, but in a stepwise way, without much explanation about used channels. The help files are rather complete, but quite frequent they are based upon a former version of Quest3D, resulting in missing documentation.</td>
<td>Simple, but very helpful tutorials teaching mainly the basics, but sometimes covering the more advances features of Vizard. The help files are reasonable, but can be better.</td>
</tr>
</tbody>
</table>
Developing a general framework for the Delft VRET application

<table>
<thead>
<tr>
<th>Quest3D</th>
<th>Vizard</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learning curve</strong></td>
<td>As Vizard is in fact plain Python, a programmer with knowledge of imperative programming will be able to master Vizard quite quickly.</td>
</tr>
<tr>
<td>The experiences of several people are that the learning curve is very steep: it takes a lot of learning time before you are able to build a simple program. Most developers will not have a reference window to this programming style.</td>
<td></td>
</tr>
<tr>
<td><strong>Extensibility</strong></td>
<td>n/a: all functionality can be written inline in the Python programming language.</td>
</tr>
<tr>
<td>Quest3D provides a channel Lua script which can be used for small pieces of code in the inline Lua scripting language (C++ look-a-like). An optional SDK (Microsoft Visual Studio C++ 6.0) is available for a larger degree of freedom. With this SDK, customer-made channels can be created.</td>
<td>By default, one network connection can be made using Vizard elements. Variables are not automatically shared, which implies a send and receive function. Python provides ways to implement multiple networking connections.</td>
</tr>
<tr>
<td><strong>Networking</strong></td>
<td>By default, one network connection can be made using Vizard elements. Variables are not automatically shared, which implies a send and receive function. Python provides ways to implement multiple networking connections.</td>
</tr>
<tr>
<td>Networking can be achieved using built-in channels. Once a ‘server’ has been created, up to 31 ‘clients’ can connect to this server and share its variables transparently. However, communicating with external programs (like the user interface) is only possible writing a new channel with the SDK.</td>
<td></td>
</tr>
<tr>
<td><strong>Flock of Birds support</strong></td>
<td>Vizard uses the open source VRPN library that enables a lot of tracker types and other hardware in a standardized way. I experienced some start-up problems, but the tracker did work.</td>
</tr>
<tr>
<td>With the first release of the VR edition, the flock of Birds was not supported yet, but with in relative short time we could use a development version of it. I experienced several problems with it. Now, more than one year later, there is still no fully correct support of the Flock of Birds: different trackers lead to different results on different or even the same computer (Bui, 2006).</td>
<td></td>
</tr>
<tr>
<td><strong>Stereo projection</strong></td>
<td>Built-in feature for two different types of stereo projection: real stereo (as we want to have it) and anaglyphic stereo ('red-green' projection).</td>
</tr>
<tr>
<td>No built-in feature; can be achieved by placing two cameras and calculating both viewpoints. After choosing for Vizard, this functionality has been added to Quest3D.</td>
<td>Examples in the portfolio of Vizard seem to me of a lower graphical quality than Quest3D, but reasonable enough for the purpose of the VRET project.</td>
</tr>
<tr>
<td><strong>Graphics quality</strong></td>
<td>Examples in the portfolio of Vizard seem to me of a lower graphical quality than Quest3D, but reasonable enough for the purpose of the VRET project.</td>
</tr>
<tr>
<td>The graphics quality will primarily depend on the skills of the modeller and the choice for modelling application. According other projects in Act-3D’s portfolio and applications developed by fellow-students, the level of graphics seems to be higher than that of Vizard, maybe because Quest3D is in principle a 3D application.</td>
<td></td>
</tr>
<tr>
<td><strong>Distribution</strong></td>
<td>Both a .VIZ- and a .PY-file (Vizard info resp. Python source code) are needed, as is the Vizard player. Source code can be encrypted. In the used version, it is not possible to publish a single executable file. The latest version, however, supports this.</td>
</tr>
<tr>
<td>The following types of executable can be chosen: executable file, installer, Quest3D viewer file, screen saver or Winamp plugin file, all at no additional costs.</td>
<td></td>
</tr>
<tr>
<td><strong>Support</strong></td>
<td>As far as I can see now, all questions on the forum are answered in at most a few days. Mostly examples are included. If a feature does not (yet) exist in Vizard, it is simply written and the respective file is attached. The forum is a great addition to the help pages.</td>
</tr>
<tr>
<td>Most, but not all, questions are answered on the forum. Answers often are abstract and more hints in the right direction than concrete examples.</td>
<td></td>
</tr>
<tr>
<td><strong>Price</strong></td>
<td>The educational license is priced $ 5,200; a commercial license costs $ 8,000. Depending on the number of distribution licenses, these are priced $ 75 up to $ 450 (bulk discounts).</td>
</tr>
<tr>
<td>The educational price of only € 230 was the decisive reason to prefer Quest3D above Vizard. The commercial license is priced € 9,000. Distribution of the programs is free of costs.</td>
<td></td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td>Backwards compatibility is not 100%.</td>
</tr>
<tr>
<td>Not always backwards compatible: switching to a newer version will sometimes mean revising the program.</td>
<td></td>
</tr>
</tbody>
</table>

*Table 4.2: A quick comparison of technical, functional and other aspects of Quest3D and Vizard.*
As might be obvious following from the statements above, I changed the choice of VR application early. I do not pretend to know for sure that Quest3D will be a less choice than Vizard, but with the experiences and the comparison in this paragraph (all from non-professionals, must be admitted) I am not able to implement a framework as required with Quest3D.

During my research and implementation, Schuemie has done one step forwards and started a new company, which aims at putting VRET in a clinical environment. Schuemie chose for Quest3D, surprisingly just because he experienced opposite results regarding usability and ease of learning. This company was abandoned however after a short period.

As a finishing remark in this question, I would like to state that in fact the choice for a VR environment is not as final as it might sound: as the virtual environments are ‘modular’ with respect to the framework, a good implementation will be able to handle multiple VR environments without too much troubles. No insuperable problems will occur at a later switch to regardless which VR application.

4.4 FRAMEWORK

Although virtual reality exposure therapy cannot exist without the virtual environments, maybe the most important part of the complete VRET system is the framework that regulates everything. The framework enables management of the VRET system in total, covering all virtual worlds and the information required to work with.

In this section, the framework is described regarding the different aspects it contains. I will not dig too deep into the code; for those interested, the source code is available at the included CD-ROM; refer to appendix C for more information. The following aspects will be covered here:

- Which programming language to use for the development?
- How are privacy and security guaranteed?
- How is the communication with Vizard established?
- How can the user interface be made dynamic?
- How can the user interface be made extensible?
- What methods of configuration will be provided?
- How is logging done?

Please note that not all of the features described in this chapter will be implemented in the prototype, which is described in chapter 5.

4.4.1 PROGRAMMING LANGUAGE

Regarding the requirement of maintainability, the programming language to choose should be a programming language that is widely known and is not expected to ‘die’ in the nearby future. The programming language should provide ways to create graphical user interfaces, network connections and database management – some requirements that most common programming languages already meet in their default installation.

There are a few programming languages that directly come into view, when looking at the applications described in previous chapters. Delphi is the programming language that the VRET system currently is written in. Some parts of the code might be re-usable. C++ is used by Quest3D for the development of your own channels with the SDK. Several other VR applications also use C++ in an SDK. Vizard will be available once in a C++ version. Python is the programming language that is used at the moment while working with Vizard.

All of the above mentioned programming language contain the basics from the first paragraph, and all of these languages are supported by a broad public, C++ maybe most of them all. For this project, I chose to use Delphi. Two reasons are behind this choice, the first one being that the current VRET system is written in Delphi. Another reason is personal, Delphi is the language I know best. The use of Delphi is not irreversible: the end product is a prototype that will be far from its final version. Using the ideas and programming structure, a switch to another programming language is not that hard.

In case Vizard will be the definite choice for the VR application, the use of Python rather than another programming language should be considered: Python is also a fully functional programming language, and for Vizard uses Python internally too, this makes it possible to program your code more efficiently.

An important difference between the three mentioned programming languages (C++, Python and Delphi)
Developing a general framework for the Delft VRET application

is that Delphi is the only one that is not platform independent. Borland (Borland Kylix, Internet) once
developed a re-implementation of Delphi for the Linux platform, called Kylix, but due to missing
interchangeability (not all code was usable between the different platforms), Borland stopped the
development. However, this missing platform independence is no reason not to choose it: due to the
very specific type of software VRET is, we can put requirements to the end user what hardware and
software to use. Let alone the fact that most hardware only (or primarily) runs on the Windows
platform.

4.4.2 PRIVACY AND SECURITY

As stated in the introduction, VRET is a system dealing with medical and therefore personal data. This
data is available at various points in the system, or during the use of the system: storage on the hard
drive, logs of old sessions in the database, and data sent over the network during a session. In the
Netherlands, security of data is formalized in several laws (e.g. the Personal Data Protection Act - WBP)
and personalized by the Dutch Data Protection Authority (CBP). Two of the main points in these
regulations are blocking unauthorized access to electronic data, and encrypting medical data as soon as
they are sent over the Internet.

We can split up the VRET application in several layers, each knowing a specific type of security. Figure
4.8 shows these layers, which will be described in the following paragraphs. In the current setup in
which the patient is always guided by a therapist, only the data connection should be secured; if in the
future therapy-at-home will be provided, the patient’s computer should be set up with more security.

![](image)

**Figure 4.8:** Overview of the different security layers of the VRET system. On all layers, a certain type of security
should be implemented. If VRET is used as therapy-at-home, also the patient’s computer should be secured in
more ways than only sending the session data; this is indicated by the gray boxes.

OPERATING SYSTEM AND FILE SYSTEM

As soon as somebody has access to the computer (physically or over the network), he can read the local
files. The VRET application itself does not work with medical files - all data is stored in the database -
but the therapist can decide to save some own files on the hard disk. As soon as these files tend to be
personal, like patient dossiers, it is advisable to save them encrypted. For this a lot of software exists,
for example TrueCrypt (TrueCrypt, Internet).

The same counts for the patient’s computer if more is done there than only running the VRET
application in a clinical environment. One can think hereby of keeping logs of therapy-at-home.

DATABASE

The database is the part of the VRET system that contains most private information. All sessions are
logged here, and the patient’s personal data is stored for easy access. Protection can be implemented
using good user management, or even store the data encrypted in the database, but be sure the
performance will not drop.

38
User rights within the application

The VRET application knows two levels of users: the system administrator and the ‘other users’. The system administrator has the right to manage the worlds and can gain access to all data of all therapists and patients. The system administrator also is the only one that can access the user management: assigning other users the administrator status, and adding, editing or removing users. The other users will mainly be the different therapists. These users only have access to their own patient management and session data of their own therapies.

At start up, the application is blocked until the user authenticated him- or herself. In the current prototype, a simple username-password combination is implemented for this access validation, but the final version might be equipped with a more sophisticated method, like biometric checks (finger prints, iris scan, ...).

This different security levels are implemented throughout the whole program, and define the functionalities and data visible and usable for the user.

Network connections

At the moment a session is being performed, data is sent between two computers. This opens the risk of eavesdropping. As long as the computers are using a direct wired network connection and are not connected to the internet, security is no real issue, as intercepting data is almost impossible. A wireless connection increases the risk, and obviously using an internet connection for data transmission is ‘open to all’. The VRET application will therefore always use secure SSL connections, built-in in most operating systems and programming languages.

4.4.3 Communicating with Vizard

Obviously, the Vizard applications on the therapist’s and the patient’s computer are transmitting data to synchronize the worlds. However, as the VRET application uses more viewpoints of the virtual environment, all these instances should be synchronized. This is done by four network connections. Figure 4.9 shows a scheme of these available network connections.

The four viewpoints of the VRET application are the following:

- the patient’s computer shows a stereoscopic view of the virtual world;
- an exact copy of this view (but monoscopic view) is available for the therapist - the ‘patient’s viewpoint’;
- the therapist also has the disposal of a free viewpoint;
- and the location map is the last viewpoint usable by the therapist.
All these four viewpoints should show exactly the same world, which implies that all parameters should be shared between all instances. This happens between two different computers (the patient’s and the therapist’s), but also locally on the therapist’s computer.

Vizard provides by default a clustering functionality, that synchronizes the output on two or more computers automatically. The problem with respect to the VRET application is that all these computers should be physically different; the VRET application, and more specific the therapist’s computer, uses multiple instances on one and the same computer. The result of this is that the clustering function can not be used, and we must fall back to normal network connections.

The Vizard classes include a networking functionality, which is very easy to set up and use. This class limits the connections to only one, so the other three connections are implemented as separate network sockets. For an implementation example of both, please refer to the Vizard analysis in paragraph 4.3.2.

The framework listens to only one network connection: the one on the patient’s computer. From here, the framework continually fetches information about location and orientation of the patient’s head mounted display. The other way, the framework sends all parameters to all other Vizard instances, resulting in a synchronized environment.

4.4.4 Dynamic user interface

One of the wishes and requirements is customization of the user interface. Where the current VRET application has a static user interface with all elements on fixed positions - even the window on a fixed position on the screen to allow the WorldUp players to be visible - and of fixed size, a therapist might prefer another arrangement of elements, or hiding and showing more or less elements. The addition of new elements is described in the next paragraph - but all these new elements can be configured as described here.

In the display elements a distinction is made between default display elements and dynamic display elements. The default display elements are the ones that are built-in in the application, and include the viewpoints, SUD recording, timer, and others. In short, everything that is available for all worlds, and is too specific to be implemented as ‘add-on’. The dynamic display elements are those, that are externally defined for each world, and include mainly the world controls. Both types of display elements can be shown or hidden via the application menu.

Additionally, all display elements can be moved to any position in the application window by dragging the element to another position.

Combining both features, a therapist can customize the user interface completely, showing only that elements that are required for his types of sessions, and arranging them as efficiently as possible.

The display settings are on a per-world basis: each therapist can customize his worlds to his own needs, and this settings can be saved for the next session. In case a therapist does not yet have customized a world, the display elements are shown and placed following the defaults set in the code or XML file.

I will not go into details about all display elements again, since their rationale and functionality was already covered in paragraph 3.6.3, and did not change in most cases. Some display elements however have been improved or otherwise altered; these will be handled with in this paragraph.

File

The patient file has been changed from a single text field in which comments could be added about the complete session, to an event-based notepad. The file does no longer exist of only one entry, but small notes can be made linked to a certain moment in time.

The therapist can browse through his notes and if necessary, adapt the contents or even delete the entry.

File notes are stored together with the session log, making retrieval of the notes at the exact moments in time possible.

![Figure 4.10: The File section has been changed from a single text field to an event-based notepad.](image)
The new framework

Session

The session block shows the duration of the session, which can be paused and resumed, or reset to start at zero again. This last functionality is an improvement of the existing session block, that showed only the session time. The alarm function has been preserved. The therapist can indicate an interval in minutes at which he will be notified by visual or audio events. Changing the interval will reset the timer; setting the interval at zero will turn the functionality off.

Webcam

Currently, there have not been any therapy sessions in which the patient is in an other room than the therapist. Therefore, the therapist was always exposed directly to the behaviour of the patient. One of the goals of the VRET project is ‘treatment at home’, or at least not in the same room as were the therapist is.

In this case two-way communication between therapist and patient is very important. The communication from therapist to patient will be auditive, vice versa the communication will be established by a webcam connection, for this is the only way in which the therapist can both see and hear the patient. Especially the sight can be a very important modality to see the condition of the patient.

The implementation of the webcam extension is outside the scope of this prototype, so for now a third party webcam application (for example the Microsoft MSN webcam available in all new Windows installations) should be used.

4.4.5 XML FORMAT FOR WORLD CONTROLS

Creation of user interfaces using an XML format is an area that several companies already explored. The main reason to do so, is separation of data and visualization: if you want a button to be showed, you only need to define what the button ‘says’, and what event should be triggered. The real ‘drawing’ of the button is a task of the program itself.

One of the more sophisticated standards is Glade (Glade, internet), a free and open source user interface builder for the (Linux) GTK+ and Gnome desktop environments. The library can be called in a great variety of programming languages and XML-files that once are developed, can be used fully platform- and programming language independent.

For the development of an XML-format for the VRET project, I looked at two simpler specifications, that are mainly developed for use within web applications: XForms (W3C XForms, internet) and XUL (XUL, internet). XUL is a Mozilla project that primarily aims at the development of Mozilla applications and extensions, but is also suitable for web applications. It’s main limitation is lack of support: only Gecko-based browsers can show these .xul-files, and this doesn’t include Internet Explorer, Opera, Konqueror or a lot of other browsers. XForms is developed as a new standard by the W3C, and is meant to be the successor of HTML/XHTML-forms. Development is continuously ongoing, and XForms is more likely to be supported than XUL, because even for Internet Explorer, there is a third-party plugin available.

In relation with the VRET application and the requirements for the user interface, all above described standards-to-be are far too extended: the VRET application only needs a few types of user controls, using very strict specifications. Mainly for this reason, I chose to develop a dedicated XML-format, specifically for VRET application user interfaces. This XML format is very simple, both in structure as in readability, covers all required elements from the current VRET system (and more) and is easily extensible.

Global Description

Each world can be extended with one or more XML files containing the description of the control elements. The XML files are placed into a directory named XML located directly under the directory containing the world files. The VRET application reads this directory and investigates all XML files in it. These files are included in the menu View and can be chosen to show or hide, and if shown, can be dragged to another position, just as described in paragraph 5.4.3.

An XML file describes a combination of control elements and can be as extended, or as minimized, as desired. Each file is handled by the VRET application as a single block of control elements, which means
it will be treated as a whole when showing or moving. An example of different control element blocks is
given in a next paragraph.

The global contents of an XML file is general data at the top (e.g. title, size, position), followed by the
different controls (e.g. type, contents, action). The sizes and positions are the default value, used
when no customized values are present. Customized values overrule the default values.

**XSD Schema**

The XML format is described below in XML Schema format (W3C XML Schema, internet). Examples of all
different control elements are placed in the next paragraph.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="vretui">
    <xs:complexType>
      <xs:sequence>
        <!-- Title of control group; size and position of control group -->
        <xs:element name="title"/>
        <xs:element name="left"/>
        <xs:element name="top"/>
        <xs:element name="width"/>
        <xs:element name="height"/>

        <!-- Arbitrary number of control elements -->
        <xs:element name="elements" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <!-- Button -->
              <xs:element name="button" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="label"/>
                    <xs:element name="left"/>
                    <xs:element name="top"/>
                    <xs:element name="width"/>
                    <xs:element name="height"/>
                    <xs:element name="action"/>
                    <xs:element name="parameter"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>

              <!-- Slider -->
              <xs:element name="slider" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="label"/>
                    <xs:element name="left"/>
                    <xs:element name="top"/>
                    <xs:element name="width"/>
                    <xs:element name="height"/>
                    <xs:element name="minimum" minOccurs="0"/>
                    <xs:element name="maximum" minOccurs="0"/>
                    <xs:element name="action"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>

              <!-- Radiogroup -->
              <xs:element name="radiogroup" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="label"/>
                    <xs:element name="left"/>
                    <xs:element name="top"/>
                    <xs:element name="width"/>
                    <xs:element name="height"/>
                    <xs:element name="action"/>
                    <xs:element name="option" maxOccurs="unbounded">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="label"/>
                          <xs:element name="parameter"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
The XML file starts with the declaration of some global values.

```
<vretui>
  <title>Weather conditions</title>
  <left>450</left>
  <top>56</top>
  <width>270</width>
  <height>185</height>
  <elements>
    <list>
      <label>Sunny</label>
      <left>450</left>
      <top>56</top>
      <width>270</width>
      <height>185</height>
      <elements>
        <option>
          <label>Thunder and lightning</label>
          <parameter>
            <action>
              <option>
                <label>Turbulence</label>
                <parameter>
                  <option>
                    <label>On</label>
                  </option>
                  <option>
                    <label>Off</label>
                  </option>
                </parameter>
              </option>
            </action>
          </parameter>
        </option>
      </elements>
    </list>
  </elements>
</vretui>
```
These first lines contain the title of the control block, and the default parameters for position (left and top) and size (width and height); the position is relative to the main user interface window. The elements tag precedes the declaration of the different control elements.

If no elements are added and the XML is closed right after this initialization, this code will create an empty box as shown in figure 4.14.

![Figure 4.14: Example of an empty control element.](image)

**ELEMENTS DATA**

After these general declarations, an arbitrary number of control elements can be placed into the control block, surrounded by the elements tag. Although different elements can be created, there are a few general parameters that are present in all types. These are, like the globals defined above, a (sub)title, position and size. Position is once again relative, this time to the control element box.

A new parameter is the action tag, included in all control types. This parameter identifies the action to be performed by the VRET application. The values passed to this action are different for all control types and will be described further on in the respective section.

All descriptions of control types will include a graphical example, together building the control box as shown in 4.15.

![Figure 4.15: Visualization example of a control box containing different control elements.](image)

**CONTROL TYPE: LIST**

```xml
<List>
  <Label>Weather type</Label>
  <Left>5</Left>
  <Top>30</Top>
  <Width>120</Width>
  <Height>100</Height>
  <Action>changeWeatherType</Action>
  <Option>
    <Label>Sunny</Label>
    <Parameter>1</Parameter>
  </Option>
  <Option>
    <Label>Cloudy</Label>
    <Parameter>2</Parameter>
  </Option>
</List>
```

44
The new framework

The list type is visualized by a list of options, out of which the user can choose one. All options are defined within an option tag, which include the label of the option, and the parameter passed to the VRET application together with the action trigger. In this example, the action changeWeatherType is performed with one of the five options. All parameters can be numeric or alphanumerical.

```xml
<option>
  <label>Rain</label>
  <parameter>3</parameter>
</option>
<option>
  <label>Snow</label>
  <parameter>4</parameter>
</option>
<option>
  <label>Storm</label>
  <parameter>5</parameter>
</option>
</list>
```

Figure 4.16: Visualization example of the list type.

**CONTROL TYPE: BUTTON**

```xml
<button>
  <label>Thunder and lightning</label>
  <left>140</left>
  <top>30</top>
  <width>120</width>
  <height>25</height>
  <action>doThunder</action>
  <parameter>Thunder</parameter>
</button>
```

The button type may be considered as the most simple element: a label is set, and an action including trigger parameter value is defined. When the therapist clicks the button, the respective action will be performed.

Figure 4.17: Visualization example of the button type.

**CONTROL TYPE: SLIDER**

```xml
<slider>
  <label>Wind force</label>
  <left>5</left>
  <top>150</top>
  <width>260</width>
  <height>25</height>
  <action>changeWindForce</action>
  <minimum>1</minimum>
  <maximum>12</maximum>
</slider>
```

The slider type allows gradually changing the intensity of the parameter between two limits using a slider. These limits are optional, and if not given, the default values for minimum (0) and maximum (100) will be used. Each time the slider changes position, the action is triggered with the new value.
Developing a general framework for the Delft VRET application

CONTROL TYPE: RADIO GROUP

```
<radiogroup>
  <label>Turbulence</label>
  <left>140</left>
  <top>70</top>
  <width>120</width>
  <height>60</height>
  <action>doTurbulence</action>
  <option>
    <label>On</label>
    <parameter>on</parameter>
  </option>
  <option>
    <label>Off</label>
    <parameter>off</parameter>
  </option>
</radiogroup>
```

The radiogroup type is similar to the list type regarding functionality - a list of options is shown and the therapist can choose from it. The only difference is the way of visualization.

```
Turbulence
  On
  Off
```

Figure 4.19: Visualization example of the radiogroup type.

An arbitrary number of control types can be defined in one XML file. After all controls have been defined, the XML file is closed by placing end tags. The visualization of this example is shown above in 4.15.

4.4.6 USING THE CONTROL GROUPS TO MANIPULATE THE VIRTUAL WORLD

Between the framework application and the different running virtual worlds, the commands the therapist gives must be shared. A UDP network connection enables this. As soon as an event takes place in the user interface, for example pressing a button, a CSV-format command is sent to the different Vizard instances, for example, if we take the above defined button, `doThunder`.

As commands are sent not continuously, but ‘only’ ten times a second, performing multiple commands very quickly after each other will result in a combined command, for example

```
doThunder~Thunder~changeWindForce~8
```

A developer creating a new virtual world will have to catch these commands, and perform the right actions. The prototype uses for this the `mynetwork`-functions. In this way, a new control group can be added really quickly in the user interface, and implemented in the virtual world (at least, catching the command is easy, performing the right graphic action can be hard, though!).

4.4.7 LOGGING

Each session is logged - date, time, session number and patient, navigational data, actions performed and notes made in the patient file. In this way, it is possible to re-play a session identical to the original session, either with or without the patient. This logging is achieved by continually writing data to the database - in the current implementation ten times a second. This provides a fluent play-back when
reviewing a session. An example of re-viewing is given in the walk-through of the prototype in chapter 5.

### 4.4.8 DATABASE STRUCTURE

A database is used to store all configuration details (visual, but also patient, therapist and world management), and all session data in order to provide the logging and play-back functionality. The database structure is shown in 4.20.

![Database structure of the VRET application.](image)

### 4.4.9 CONFIGURATION PANEL

The configuration panel allows the therapists (with restricted rights) and the administrator (with full rights) to configure the VRET framework.

Patient management includes adding, changing or removing patient data, either restricted to patients under own management (therapist rights) or without restriction (administrator rights). The administrator can also link a patient to a therapist and change this relation later on.

Therapist management is allowed only with administrator rights. Therapist data can be added, changed or removed. Password management is done by the administrator, too.

World management is also allowed only for those users with administrator rights. Details of the virtual worlds can be altered, including location on the hard disk or network.

### 4.4.10 EXTENSIBILITY

One of the improvements regarding the original VRET application is the transparent way of adding control elements. Instead of digging into the code, creating new layers and implement new functionality, using the XML format as described above allows a developer to add control elements in a very easy way. The only technical implication will be in the virtual world, where the trigger, sent by the framework, should be caught.

More sophisticated new functionalities that are not related to a single virtual world but rather affect the framework itself, cannot be set up as easily as the control elements. One can think of implementing a webcam connection, artificial intelligence agents, or external sensors (heartbeat, voice recognition, etc). These additions are too complex and too specific to make accessible in a default way. The current Delphi code is clear and general enough to make implementation of this type of additions possible.

A future research project can be investigating how the framework can use separate modules that add the above described functionality. In this way, the original, core source code is kept clean and unchanged, and an externally compiled module is included for additional functionality. A second advantage is minimizing the use of resources: only the modules that are really used, have to be added.
5 Prototype

A first version of the new framework including a simple virtual environment to test with has been developed as part of this research. A first version, for it is an illusion to think that it is possible to develop a complete application in the short timespan of a master thesis project. In this version the basics of the program have been implemented without focusing too much on a fancy lay-out or a complicated virtual world. The clear structure of the code creates the basis for easy further development and extension.

This chapter starts with a quick walk-through, analogue to the description of the current system in chapter 3. This walk-through summarizes the functionality of the prototype in an easy-reading way without focussing too much on the technical details. These details will be covered in the following paragraphs.

5.1 Quick Walk-through

5.1.1 Preparing the Session

Before the patient arrives, the therapist ensures that the system has been set up correctly, so that the patient does not have to wait for this during the session. Likewise to the current system, this includes connecting and turning on all hardware devices, turning on both computers and running the listener application on the patient’s computer.

![Login Form]

As a preparation to the upcoming session, the therapist wants to review the previous sessions shortly to refresh his mind, and to allow himself to optimize today’s session. He starts the VRET application, chooses his own name from the drop down menu and enters his secret password (figure 5.1).

![Figure 5.1: A screenshot of the prototype showing the log in form.]

As a preparation to the upcoming session, the therapist wants to review the previous sessions shortly to refresh his mind, and to allow himself to optimize today’s session. He starts the VRET application, chooses his own name from the drop down menu and enters his secret password (figure 5.1).
The list of logged sessions (figure 5.2) is a short one yet - only a few sessions have been performed. But the SUD diagrams show a small, but clear decrease in level of fear. The last session was a tough one; a lot happened. Let’s review that session quickly… (figure 5.3).

Figure 5.2: A screenshot of the prototype showing the list of logged sessions for a patient.

Figure 5.3: A screenshot of the prototype showing the details of a logged session.
The notes are telling a lot, but the therapist prefers a live view upon the session, and starts the playback (figure 5.4). He saw an interesting note at minute 4, so he waits a while (complaining that this prototype does not yet support scrolling forward and backward in the session replay). There is the fragment. At this moment, he increased the number of people walking around the patient, and at the same time in the ‘real’ world, a door accidentally slammed. The patient was terribly frightened by this unexpected event, which is proved by the SUD level jumping to 10 for a minute or two.

The therapist decides that the surprise effect of unforeseen events was too much for the patient. Maybe he can use that fact in this new session.

5.1.2 Opening a virtual world
After the usual non-virtual start of the session, the therapist prepares the virtual world; he chooses the right world, looks for this patient in the list and adds a general comment: Main focus in this session: observe reaction of patient upon unexpected events.

Both computers start to rattle and after everything started completely, the therapist sees his user interface exactly as he left it last time, showing only the elements that he needs. Today he will observe the reaction upon unexpected events, so he adds the movie controls to the user interface, and saves it for next time.

The user interface is quite empty, as the therapists doesn’t want to be distracted by a lot of unused elements. The interface is shown in figure 5.5.
Performing the session is not really different from performing a session in the current system. The main differences are creating separate notes on different timestamps rather than writing in one big text field, and the way the world controls can be visually managed.

During the whole session, playback data will be written to the database in order to be able to review the session at a later moment.

After the session has ended and the patient has left the room, the therapist has a quick view on the logs for the patient. He sees there is a little improvement regarding the SUD charts, and that is in keeping with his practical findings during the session.

5.2 The virtual world

The virtual world that I developed for this prototype is a very simple one based on examples shipped with the Vizard application. It is just a world to show the possibilities and features of the framework, and not meant to be used in a real therapy. The world is based on the currently going on research on agoraphobia and other social phobias, and shows a large, empty room in which the ‘patient’ is facing one other person in the room. The following parameters are included, to create a certain degree of fear:

- Up to hundred people can be placed into the room. These people start walking randomly. To create a more dense feeling, the people move in a close range around the patient, making the scene look more crowded.
- The projection screen can be raised or lowered, and a film can be shown on the screen.
- The facing person can perform several actions, like shout, wave or run.
Although the virtual world is meant to be run in a VRET environment, it is possible to run it stand alone. To achieve this, one has to add a trigger that acts upon a keyboard or mouse event to the Vizard source code. This can be useful if, for example, no motion tracker is available while testing or demonstrating. The source code of the Vizard world can be found at the included CD-ROM; refer to appendix C for more information.

5.3 Setup of Hardware and Software

The hardware requirements are similar to that of the current system: two state-of-the-art personal computer suffice, and the Ascension Flock of Birds and Cybermind Visette Pro can be used for motion tracking and head mounted device. As Vizard supports multiple input and output devices, the motion tracker and head mounted device might be of different brand or type.

A Vizard license is required for both the therapist’s and the patient’s computer. Without license, one can use the 90-days trial version, but this is time-limited and shows a watermark through the video output.

The VRET application itself can be run on Windows XP; if this is a default installation, no dependencies should break the program.

Be sure to modify both the phobia.ini configuration file, and the in-program configuration via the Settings tab, so that the correct location of the virtual worlds is entered.

5.4 The Framework

Opening a single executable file and taking care of a running Listener application is all that is required to get the VRET application running. To actually see the application in action, one or more virtual worlds should be available and configured, as are one or more therapists and patients.

The screenshots shown in this chapter are in many cases detail views of a screenshot, to improve readability. The complete screenshots can be found in appendix B, and are referenced in the figure-captions.

5.4.1 Phobia.ini

The configuration file phobia.ini, to be placed in the directory C:\, contains general runtime information for the framework and Vizard application. This file indicates which computer is running (therapist’s or patient’s), the location of important files and network configuration. Table 5.1 shows an overview of all configurable variables and their restrictions.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Therapist’s computer?</th>
<th>Patient’s computer?</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThisCompu</td>
<td>+</td>
<td>+</td>
<td>Console VRstation</td>
<td>Indicates whether ‘this’ computer is the therapist’s computer (Console), or the patient’s computer (VRstation)</td>
</tr>
<tr>
<td>VEdir</td>
<td>+</td>
<td>+</td>
<td>Directory</td>
<td>Full directory path that contains the virtual world files</td>
</tr>
<tr>
<td>Executable</td>
<td>+</td>
<td>+</td>
<td>File location</td>
<td>Full directory path and filename of the Vizard application</td>
</tr>
<tr>
<td>IPconsole</td>
<td>+</td>
<td>+</td>
<td>IP address</td>
<td>IP address of the therapist’s computer</td>
</tr>
<tr>
<td>PortConsole</td>
<td>+</td>
<td>+</td>
<td>Integer</td>
<td>Port number on the therapist’s computer that will be opened for listening</td>
</tr>
<tr>
<td>IPVRstation</td>
<td>+</td>
<td>+</td>
<td>IP address</td>
<td>IP address of the patient’s computer</td>
</tr>
<tr>
<td>PortVRstation</td>
<td>+</td>
<td>+</td>
<td>Integer</td>
<td>Port number on the patient’s computer that will be opened for listening</td>
</tr>
</tbody>
</table>

Table 5.1: Overview of all the variables to be initialized in the configuration file phobia.ini.

An example of the phobia.ini file on the therapist’s computer:

```
# File initialization
ThisCompu = Console
VEdir = h:\vret\ve
Executable = C:\Program Files\Vizard25\bin\winviz.exe

# Network initialization
```
An example of the phobia.ini file on the patient’s computer:

```
# File initialization
ThisComp = VRstation
VEdir  = h:\vret\ve
Executable = C:\Program Files\Vizard25\bin\winviz.exe

# Network initialization
IPconsole = 169.254.227.221
PortConsole = 4950
IPVRstation = 169.254.227.222
PortVRstation = 4951
```

5.4.2 **Listener**

The Listener application does not differ from the Listener application in the current VRET system. The only functionality of this application is catching incoming requests to open a world. The Listener application reads the phobia.ini configuration file and with this data together with the data sent by the therapist’s application, it opens the requested world. After initialization, Vizard takes over all responsibilities, including sending and receiving network data: the listener finished its job.

The source code of the Listener application can be found at the included CD-ROM; refer to appendix C for more information.

5.4.3 **Therapist’s user interface**

**Program flow**

The framework is the central application in the VRET system - all data, sessions and virtual worlds are managed by and started from this central application. The flowchart in figure 5.7 describes the program flow, starting with the execution of the program. The only two actions after starting the program are immediately exiting the program, or log in. After having logged in, one can choose for log out, performing a session (which contains of initializing and actually running the session), view the historical data (including optional play-back) and manage the configuration parameters. From all these four functionalities (between which can be switched without restrictions), the program can be quit or the user can log himself out.

Figure 5.6: Flow chart diagram of the prototype. After logging in, a choice must be made out of the four options. At each time, one can return to the main menu. The external Vizard application is started at the start of a new session, or at the playback of a recorded session.
Developing a general framework for the Delft VRET application

The flowchart of a session to be performed is shown in figure 5.7. After having initialized the session, the user can close the world, change the user interface, save the current settings of the user interface, send commands to the virtual world, and use the SUD registration, notepad, free viewpoint, and the alarm and session timer.

Figure 5.7: Flow chart diagram of the prototype, performing a session. When the session runs, the virtual environment can be altered by acting controls, or the therapist’s user interface can be changed by adding, removing or moving user interface elements. This lay-out can be saved.

‘Log in / Log out’ Tab
The start screen of the application shows a log in form, or, if already logged in, a log out form. If not logged in, the only available actions are to log in, or to exit the program (via the menu). As soon as a user has logged in, the other tabs are revealed, and the menu is activated.

As a simple implementation of a user credential system, a username and password combination is required which will be checked against a credentials file (the database). The username is to be chosen from a list - this list contains all therapist’s configured in the application. A person that is not yet present in this list of users, will have to ask the system administrator to add him as a new therapist to the application.

Figure 5.8: Login form (full screenshot in Appendix B, figure 8.3). Before being able to use the application, you have to log in into the application using your username and password.
If no session has been started yet, an initialization form is shown. The therapist should enter the session data (patient, virtual world, session number, and optional comments) in this form; after that, the session is started. An example of the initialization form is shown in figure 5.10. The session number is pre-entered. Based upon the last session number for that patient, the application makes a suggestion. Of course the therapist is free to alter the session number.

The user interface in the VR session tab to control the virtual session is fully customizable. All control elements can be shown, hidden and moved to an adequate position. Showing and hiding elements is established through the menu View. Moving can be performed by dragging the mouse, while pressing the Alt key. For more details about this feature, refer to paragraph 4.4.4. Through the menu View, the lay-out can be saved. Currently, this happens on a per-world base, meaning that next time the world is opened, the saved lay-out is shown, regardless of the patient for whom the session is performed. A distinction is made between the built-in control elements (default controls) and the control elements that come with the virtual world (dynamic controls). Figures 5.11 and 5.12 show examples of the submenus under the menu View.

**Dynamic user interface**

The user interface in the VR session tab to control the virtual session is fully customizable. All control elements can be shown, hidden and moved to an adequate position. Showing and hiding elements is established through the menu View. Moving can be performed by dragging the mouse, while pressing the Alt key. For more details about this feature, refer to paragraph 4.4.4.

Through the menu View, the lay-out can be saved. Currently, this happens on a per-world base, meaning that next time the world is opened, the saved lay-out is shown, regardless of the patient for whom the session is performed. A distinction is made between the built-in control elements (default controls) and the control elements that come with the virtual world (dynamic controls). Figures 5.11 and 5.12 show examples of the submenus under the menu View.
The default interface elements are fully described in 4.4.4. The dynamic interface elements that come with each world, are described here for the prototype world.

The auto pilot is only a dummy control, with no added functionality. It shows the use of a slider and buttons.

The avatars controls the number and the actions of the avatars in the virtual world: up to 100 other people can be added to the virtual world, and can walk around the patient or stand loitering. Adding or removing people is done by moving a slider; the actions are connected to buttons.

The movie controls the white screen (that can be moved up or down) and the playing of the film on the screen.

The avatar controls the facing person in the world. This person can perform different actions, like shouting, waving or falling down.

The report contains the example XML as described in paragraph 4.4.5.

'Logs' tab

Information about all performed sessions is stored in a database and accessible by the respective therapists. The logs give the opportunity to review the sessions a patient has had, see statistical information and even replay the session if required.
After a patient has been chosen from the drop down list (figure 5.14), the left part of the screen shows a quick summary of all sessions that patient has had (figure 5.15). The general session data as entered in the initialization form is shown, as is a small SUD chart.

Clicking on a specific session shows the details for that session in the right part of the screen (figure 5.16). These details include information on the virtual world, general session data, a full SUD chart, all recorded notes including timestamps and a button to perform a session play back (figure 5.17).
Developing a general framework for the Delft VRET application

Figure 5.16: Details about a previous session.

Figure 5.17: Each session can be played back at a later moment.
“Settings” tab
The last tab, Settings, is accessible to all users of the application (figures 5.18 and 5.19). However, not all users have the same rights here. A ‘normal’ therapist can only view and manage his own patients, and can view (but not edit) the configuration of the therapists and the virtual worlds.

A user with administrator rights is allowed to see and edit (or even remove) all patient and therapist data, and also the data regarding the virtual worlds.

The source code of the user interface can be found at the included CD-ROM; refer to appendix C for more information.

5.4.4 Patient’s user interface
As the Vizard application handles all communication to and from the therapist’s computer, the framework contains no user interface for the patient’s computer. One could say the user interface is the combination of the Listener application (triggers the start command) and the Vizard virtual world. Both are described earlier in this chapter.

5.5 Evaluation

5.5.1 Evaluation by a ‘therapist’
The prototype has been evaluated by one of the other master students involved in the VRET project, who has gathered some experience with the psychological view upon the project due to multiple attended real VRET sessions by therapists and patients in three different clinics using the current VRET system.

The main remark is the lack of information: the prototype has been developed from a (one) developer’s point of view, and this resulted in implicit assumptions of the functionality of the application. Examples of this are the way in which to configure the program (adding or editing patient data), or changing the interface lay-out.

Some of these shortcomings should be solved in the application itself, others (like the description of the different world controls) will need to be documented. In this, it strikes that the current VRET application neither does include much documentation about the functionalities or the virtual worlds. This is likely to be a result of the fact that only a few people use the VRET application, people that already were very involved in the development of it. As soon as new users will have to work with the current VRET application, the lack of documentation will reveal itself.

A disadvantage of the current Vizard version is the visualization of it: the trial edition shows a
Developing a general framework for the Delft VRET application

watermark and a debug ‘command line’ window, that appears in front of the application. The development and enterprise editions allow executables to be created, the latter even protected executables.

5.5.2 Evaluation regarding requirements
Regarding paragraph 4.1, the framework and the prototype can be held against the requirements stated.

Continuation
It is always difficult, if not impossible, to predict the chance that the application as it is now, can be used in two, five, ten or thirty years in the same way. Regarding the rapid developments in the IT, I tend even to state that this chance is zero. But the modular setup enables partially renewal of elements of the application: the worlds can be built in another modeller, as long as the VR application supports this modeller’s format. The VR application can be linked to the framework, as long as the connection schemes (see paragraph 4.4.6) are followed.

Modularity
The text above implies that the requirements regarding modularity were met.

Security
The prototype is really not secure: nor the data, nor the application is stored in a secured way; neither is the network connection between therapist and patient. However, the ways of securing the application are stated in paragraph 4.4.2.

Logging
The prototype allows full logging of the sessions: SUD’s, therapist’s notes and the session itself are being logged at all times. This information is easily retrievable, too. Some things need a little fine-tuning, but the raw idea has been implemented.

Only one application
The prototype allows configuration of all virtual worlds - all sessions can be started from the prototype. In fact, there is no other way to start a session. More specifically, as long as the specifications are met, there is no reason why virtual worlds of complete different VR applications can’t be included in this single framework.

Patient management
The prototype allows basic patient management, as well as basic therapist (user) management and management of the worlds. In an ideal future situation, the framework application is linked to the patient and therapist database of the institute, allowing full integration of data in both ways.

Flexible user interface
As one of the main goals, the application was set up with a flexible user interface in mind. This has been established by allowing the user of the application defining how he or she wants the interface to be visualized. This lay-out can be stored on a per-world basis; a future version can work with a lay-out stored for each patient and each world.
6 **CONCLUSION**

Renewal of the current implementation of the VRET system is a necessity, having in mind new research goals and usage in clinics. Other reasons for a study of the options for renewal are changed requirements, and continuation problems with the current software. This study is meant to analyse and evaluate the first steps in this process.

6.1 **CURRENT SYSTEM**

In the course of the research, it became clear the initially set research goal was incomplete, and an additional goal was introduced: the description of the current VRET system. The reason for this was, that this study will be one of the latest studies in which the current system is described (and one of the few that describes it in a technical and complete way), and also because the main goal is to re-write the current system to a new one. A considerable part of this report is dedicated to a detailed description of this system, in all its aspects: hardware and software setup, communication channels and the parts of the framework itself. All of these aspects prove not to be ready for future development, apart from some hardware.

Although the chapter about the current system mainly aims at being describing, a few paragraphs include comments and recommendations for improvements. These are formalized in the requirements analysis for the new system.

6.2 **NEW SYSTEM**

Another change with respect to the original research goals was the use of Quest3D as development environment. Practical experiences of several students showed that Quest3D was not as easy and complete as it looked on the first sight. This made me decide to review the choice of Quest3D, resulting in the use of another application for the prototype: Vizard. This kind of switches will not be a big problem, even in the future when the system is running in a clinical environment, as the framework and the virtual worlds are different ‘modules’ of the complete application. One thing to keep in mind is the use of a ‘general’ modelling tool, such as Maya or 3D Studio Max, whose models can be imported to a lot of different VR applications.

The foundation has been laid for a new framework, which serves as a centralized application in which all different virtual environments for the VRET therapies can be managed and accessed. This framework has been inspired by the current system regarding basic functionality and lay-out, but is set up modular, so that extension is easily possible. Part of the extensions are the controls for the virtual worlds, which can be developed without changing the source code of the framework. For this, a special XML format has been developed.

The centralized application offers other benefits, like patient management and session management. The session management keeps track of previous sessions and includes play back of the performed sessions.

One of the key points in the design was a flexible user interface, that can be adapted on a per-patient base. The prototype supports modification of the user interface by showing, hiding and moving the interface elements, and supports saving of the lay-out on a per-world, per-therapist base.

6.3 **RECOMMENDATIONS**

This report describes the framework in a general way. The prototype covers most of the requirements, but of course a lot of improvements can be made here.

The prototype handles a fictive world, with fictive therapists and patients. The real benefits of the framework can only be shown with real data and ‘real virtual’ worlds.

Requirements that have not been implemented in the prototype are the security layers and the audio/video connection. The security layers are not implemented as this is a complete different research subject; the audio/video connection has been omitted in the prototype because common applications like Skype or MSN offer enough functionalities in this area for good connections.

Future extensions can include sensors (heartbeat, voice analysing), artificial intelligence agents and level based sessions. All of these are brand new subjects in the field of VRET and require new research.
7 REFERENCES

1. B. Aslan - Porting the Delft VRET system to Windows XP - internship report, Delft University of Technology and Rotterdam University - July 2006a

2. B. Aslan - Manual for the Delft VRET system, written for the VALK foundation - internship report, Delft University of Technology and Rotterdam University - July 2006b


5. College Bescherming Persoonsgegevens - Omgang met uw medische gegevens - December 2006


11. C.A.P.G. van der Mast - Elaborated case for MKE project 1 (IN1810) - Delft University of Technology - November 2004


14. S.J. Roorda - VRET - Ready for the future! - Research assignment report Delft University of Technology - 2005a (this report has been includes in appendix D)

15. S.J. Roorda - Getting the VRET-system working on Windows XP-computers - Intermediate report Delft University of Technology - 2005b (this report has been includes in appendix D)


18. M.J. Schuemie - Phobias and Virtual Reality, the computer system - Intermediate report, Delft University of Technology - 2002


20. R. Sopacua - VRET for agoraphobia - Research assignment report Delft University of Technology - 2004
INTERNET REFERENCES

    Wikipedia article concerning the discontinued Borland programming tool Kylix

    Glade, a User Interface Designer for GTK+ and GNOME - Gnome project

    The creators of Quest3D - background of Quest3D (October 2005)

    True Crypt - Free Open-Source On-the-fly Encryption

25. W3C (XForms) - http://www.w3.org/TR/2006/REC-xforms-20060314
    XForms 1.0 (Second Edition) - W3C Recommendation - 14 March 2006

26. W3C (XML Schema) - http://www.w3.org/TR/xmlschema-1

27. XUL - http://www.mozilla.org/projects/xul
    XML User Interface Language (XUL) - Mozilla project
APPENDICES

APPENDIX A - HARDWARE CONNECTION DIAGRAM OF THE OLD VRET SYSTEM

Figure 8.1: Hardware connection diagram of the old VRET system for the therapist's computer.

Figure 8.2: Hardware connection diagram of the old VRET system for the patient's computer.
Appendices

Appendix B - Full Screenshots of the Prototype

Figure 8.3: Prototype screenshot: login screen
Figure 8.4: Prototype screenshot: logout screen
Figure 8.5: Prototype screenshot: entering data to start a new session
Figure 8.6: Prototype screenshot: running session, showing in the menu the default control elements
Figure 8.7: Prototype screenshot: running session, showing in the menu the dynamic control elements.
Overview of the results

Patient: Hans de Vries
Therapist: Sistem Records
Birthdate: 1973.01.02

Session 1
- World: Agoraphobic Room
- Time: 12:00 - 13:15
- Duration: 01:15
- Comments: "Hans was very calm and the session was very successful."

Session 2
- World: Agoraphobic Room
- Time: 13:15 - 14:30
- Duration: 01:15
- Comments: "Session was successful, Hans showed improvement."

Session 3
- World: Agoraphobic Room
- Time: 14:30 - 15:45
- Duration: 01:15
- Comments: "Hans was feeling good, continuing improvement."

Session 3 - General information

World: Agoraphobic Room
Session started: 25-11-2007 15:02:46
Duration: 01:30
Comments: "Session was successful, Hans showed good progress."

SUD chart

File records

00:10 Hans was feeling very calm and was very enthusiastic about the session.
01:15 "I feel much better today. I am ready to face the world again."

Figure 8.9: Prototype screenshot showing the session log of a patient, including details of one of the sessions.
Figure 8.10: Prototype screenshot: showing the session details of a specific session, while playing back that session
APPENDIX C - CONTENTS OF THE CD-ROM

This master thesis report includes a CD-ROM with the digital version of this thesis and the source code of the prototype, together with some other reports.

The included CD-ROM contains:

– All source code of the prototype:
  – the Vizard application
  – the listener application
  – the framework itself
– All external files required for this source code
– PDF versions of the following reports:
  – This master thesis report
  – VRET - Ready for the future! (Roorda, 2005a)
  – Getting the VRET-system working on Windows XP-computers (Roorda, 2005b)

More information about the contents of the CD-ROM can be found on the README.TXT file in the root folder of the CD-ROM.
APPENDIX D - INTERMEDIATE REPORTS

The following intermediate reports have been attached to this report and follow immediately hereafter:

– S.J. Roorda - Getting the VRET-system working on Windows XP-computers - Intermediate report Delft University of Technology - 2005
– S.J. Roorda - VRET - Ready for the future! - Research assignment report Delft University of Technology - 2005