The Added Value of VEs to Network Management

For internal use only at KPN

R&D-RA-96-1040

Author: Jeroen Dijk

Date: September 1996

The Added Value of VEs to Network Management

For internal use only at KPN

R&D-RA-96-1040

KPN Research

Information sheet issued with Report R&D-RA-96-1040

Title:	The Added Value of VE to Network Management
Abstract:	The purpose of this thesis is to prove the added value of Virtual Environments (VE) to Network Management. On the base of the description and definition of VE, the description of an Network Management system and a list of guidelines, a demonstrator is designed and implemented. This prototype system demonstrates the Performance Management of a PABX. With a usability test the added value can be proved.
Author:	Jeroen Dijk
Reviewers:	Ir. J. van Wijk, Prof. dr. ir. L.J.M. Nieuwenhuis, Dr. ir. A.J.S. Hin (R*u*G)
Department:	SDS
Project:	Devine
Project manager:	Ir. R.W. Nijenhuis
Project number:	81411
Commissioned by:	Raad van Bestuur
Date:	September 1996
Classification	For internal use only at KPN
Person responsible at KPN Research:	Drs. E.P. van Aagten
Key words:	Virtual Environments, Network Management (NEMS)
Mailing list:	Aagten, drs. E.P. van (SDS); Bergh, drs. C.A.A. van den (SDS); Biegstraten, B. (SOA); Coolen, ir. L.A.A.M. (DIR); Dijk, J. (R*u*G) (4x); Essen, ir. H.P. van (TTS); Hin, mw. dr. ir. A.J.S. (R*u*G); Jager, ir. E.J. (SDS); Klok, mw J. (SDS); Liempd, ir. E.P.M. van (SDS); Luljken, mw. ir. M.J.E. (SDS); Nieuwenhuis, prof. dr. ir. L.J.M (CAS, R*u*G); Nijenhuis, ir. R.W. (SDS); Peper, ir. J.C.A. (SDS); Reijmerink, ir. R.A.J. (TTS); Thieme, ir. W.J. (STR); Verhoeven, ir. T. (SOA); Wijk, mw ir. J. van (SDS).

Contents

List of Figures

List of Tables

Management Summary

The SOA Centre of PTT Telecom manages PABXs of customers by remote. The SOA Centre uses the SOA system for this management. This system has a two dimensional, text based interface and a lot of technical knowledge is needed to use this system. A new user interface technology are Virtual Environments. These are three dimensional worlds in which users can interact through devices. These Virtual Environments are very suitable for tele-operations, a medium trough which people can interact with remote places. This new technology seems to be very suitable for the remote management of PABXs. In this research we want to prove the added value (on functionality and usability of a new application with regard to the old situation) of Virtual Environments to Network Management. We try to do this on the basis of a prototype system that demonstrates the Performance Management of PABXs via a Virtual Environment.

This research delivered a number of results. First it delivered a description and definition of Virtual Reality and Virtual Environments and a light is put on the Human Factors side of Virtual Environments. Secondly a description is given of the Network Management of PABXs and the SOA system of PTT Telecom, which will be the starting point of the design of the Virtual Environment demonstrator "Virtual SOA". Third there is composed a list of guidelines that can serve as starting point for the design of Virtual Environments. Finally there is designed a demonstrator called "Virtual SOA", which demonstrates the Performance Management of PABXs via Virtual Environments.

We recommend to complete this research with a usability test among the target group of the demonstrator. This test is needed to prove the added value of Virtual Environments. We did not have enough time to do such a test. Still there are enough indicators to continue the research on Virtual Environments and application areas of Virtual Environments like Network Management. The user interface is very intuitive and suitable to novice users, and users with no technical knowledge of PABXs. Virtual Environments also look very suitable for the management tasks. A lot of data can be visualised easily and conclusions can be easier drawn than with text based interfaces. The guidelines from the Requirements phase are a good starting point for the design of Virtual Environments and it is recommended to use such design guidelines.

Abstract

The subject of this report is the use of Virtual Environments In telecommunications management. This research serves as master thesis in Computer Science at the University of Groningen, and takes place at KPN Research, Groningen. This research has the following problem definition:

"Does a Virtual Environment add value to remote management of PABXs?"

We want to answer this problem definition on base of the following objective:

"Prove the added value of Virtual Environments to Network Management on the basis of a demonstrator which visualises a number of performance data of a PABX."

The research is divided into the phases *Definition*, *Requirements*, *Design* and *Implementation*. In the Definition phase the necessary information and knowledge for this research are studied. First information is gathered from literature on Virtual Reality and Virtual Environments. Second information is gathered from literature and practise on Network Management in general and on the management of PABXs, especially Performance Management, performed by PTT Telecom with the management system SOA. A PABX is a telecommunication switch for internal communication and communication with the public telephone network. In the Requirements phase the requirements for the design and implementation are gathered. These include design guidelines, which can be used as a starting point for the design of a Virtual Environment. In the Design phase the user interface of the Virtual Environment is designed. Besides the user interface also a generator for input data and an object oriented simulation are designed. The data is needed to start a simulation of the performance of a PABX. The requirements from the Requirements phase are applied on the design. In the Implementation phase we implemented the design we made. The design is implemented with Superscape VRT, a 3D world modelling tool, on a PC platform. The simulation is implemented with the script language of Superscape and the data generator in Pascal.

To prove the added value of Virtual Environments to the Performance Management of a PABX, it is necessary to perform a user test. In this research there was not enough time to perform a user test and to prove the added value on base of the test results. Though there are indicators that Virtual Environments do add value to Network Management. The user interface is very intuitive and suitable to novice users and users with no technical knowledge of PABXs. Virtual Environments also look very suitable for the management tasks. A lot of data can be visualised easily and conclusions can be easier drawn than with text based interfaces. The guidelines from the Requirements phase are a good starting point for the design of Virtual Environments and it is recommended to use such design guidelines.

Samenvatting

Het onderwerp van dit rapport is het gebruik van Virtual Environments In telecommunicatie management. Dit onderzoek dient als afstudeer onderzoek in de Informatica aan de Rijks *universiteit* Groningen, en is extern volbracht bij KPN Research te Gronlngen. De probleemstelling voor dit onderzoek is:

"Geeft een Virtual Environment toegevoegde waarde aan het op afstand beheren van een PABX?"

We willen deze probleemstelling beantwoorden aan de hand van de volgende doelstelling:

"Het bewijzen van de toegevoegde waarde van Virtual Environments aan Netwerk Management op basis van een demonstrator die de performance van een PABX weergeeft."

Het onderzoek is opgebouwd uit de fasen *Definitie*, *Requirements*, *Ontwerp* en *Implementatie*. In de Definitie fase is de nodige informatie en kennis verzameld. Eerst is er een literatuur onderzoek op het gebied van Virtual Reallty en Virtual Environments. Daarna Is de nodige informatie verzameld uit literatuur en de praktijk over Netwerk Management in het algemeen en over management van PABX-en, en dan vooral Performance Management, zoals die door PTT Telecom wordt uitgevoerd met het management systeem SOA in het bijzonder. Een PABX is een bedrijfs telefooncentrale die zorgt voor de Interne communicatie en de communicatie met het publieke telefoon netwerk. In de Requirements fase zijn de vereisten en behoeften opgesteld die nodig zijn voor het ontwerp en de implementatie. Deze bevatten onder andere een lijst met ontwerp richtlijnen, die als uitgangspunt kunnen dienen voor het ontwerpen van een Virtual Environment. In de Ontwerp fase is de gebruikers interface van de Virtual Environment ontworpen. Naast deze interface zijn ook een data generator en een object georiënteerde simulatie ontworpen. De data dient als input voor de simulatie van de performance van een PABX. In de Implementatie fase is het ontwerp geïmplementeerd. Hiervoor is het 3D ontwerp pakket Superscape VRT gebruikt, op een Personal Computer platform. De simulatie is geïmplementeerd in de script taal behorende bij Superscape, en de data generator is geschreven in Pascal.

Om de toegevoegde waarde van Virtual Environments aan de Performance Management van een PABX te kunnen bewijzen, is het nodig om een gebruikers test uit te voeren. In dit onderzoek was hier geen tijd meer voor, en kon de toegevoegde waarde dus niet bewezen worden aan de hand van de test resultaten. Toch zijn er indicaties dat Virtual Environments waarde toevoegen aan Netwerk Management. De gebruikers interface is namelijk erg intuïtief en geschikt voor onervaren gebruikers zonder de technische kennis van PABX-en. Daarnaast lijken Virtual Environments geschikt voor management taken. Veel data kan duidelijker worden gevisualiseerd en er kunnen makkelijker conclusies uit de data worden getrokken dan bij op tekst gebaseerde interfaces. De ontwerp richtlijnen uit de Requirements fase zijn een goed uitgangspunt voor het ontwerp van Virtual Environments en het is aanbevolen om zulke richtlijnen te gebruiken.

List of Abbreviations

CDR	Call Detail Record
HMD	Head Mounted DIsplay
ISDN	Integrated Services Digital Network
LED	Light Emitting Diode
MAC	Meld- en Analyse Centrum [dutch]
PABX	Private Automatic Branch Exchange
PSTN	Public Switched Telephone Network
SMS	Service Management System
SOA	Service Op Afstand [dutch]
VE	Virtual Environment
VR	Virtual Reality

1 Introduction

The subject of this report is the use of Virtual Environments (VEs) in telecommunications management. This research serves as master thesis in Computer Science at the University of Groningen, and takes place at KPN Research, Groningen. First an introduction is given of KPN Research and the University of Groningen. Next a description is given on the assignment.

1.1 KPN Research

KPN Research is the research organisation of Royal PTT Nederland NV (KPN). KPN Research undertakes research in the fields of technology and the social sciences. It advises in the selection and implementation of new technologies, systems and products, it answers questions on the applications and consequences of relevant technical developments, it offers solutions for the application of business processes and the integration of telecommunication systems, and it processes new services and products. And it prepares these products and services for implementation. The financial, quantitative and qualitative needs of customers are always the driving force. Through an extensive network of international contacts and projects, KPN Research has access to the expertise of international partners.

Among other objectives, R&D activities are used for supporting client decision making by research and consultancy. Knowledge and information also are of vital importance to the decision making process. When, where and in what form or language needed, comprehensible and useful knowledge and information are offered. The needs of PTT Telecom, PTT Post and other KPN daughter companies and partners are the central driving force. The research institute answers questions in the fields of customer care, enabling the operating companies to fully direct their attentions at the customer.

This master thesis is done at the Service Development & Support (SDS) department of KPN Research. SDS has the following mission:

"Research and development in the area of telecommunication services (based on generic platforms for communication and data processing), users aspects of services and management of products and services."

SDS covers the following appliance fields:

- Service Development is developing services in its broadest sense.
- Business System Support is working on improving the intern business processes.
- Customer Support is improving the knowledge about user supporting activities.
- User Centred Engineering is thinking from the users point of view to make user-friendly interfaces.
- Information & Process Engineering focuses on knowledge management and knowledge controlling systems.

1.2 University of Groningen

The University of Groningen is a modern classical University - classical, in the sense of universal, because of its wide range of disciplines - modern, because of its research, education, organisation and facilities. The University holds a staff of five thousand people. The researchers among them are engaged in high-quality fundamental research- which often crosses the traditional boundaries of departments and faculties with striking results. Together they teach a student population of twenty thousand in more than fifty fields of study, and are responsible for some six hundred trainee research assistants who are working on their PhD theses.

The University of Groningen is the second oldest in the Netherlands. It was founded on 23 August 1614 on the initiative of the "Statenvergadering van Stad en Ommelanden van Groningen" (Groningen States General). The University began with six professors in four faculties: Theology,

Law, Medicine and Philosophy. Ubbo Emmius was the first rector magnificus. The first seventy-five years of the University's history were prosperous. The University enrolled one hundred students per year, a respectable number in those days. Nearly half of them came from abroad. In 1914, when the University celebrated its three hundredth anniversary, there were 611 students. Ten years later the magic number of one thousand students was reached. After 1945 the number of students increased rapidly and currently it stands at almost twenty thousand. There are now ten faculties, housed in 150 buildings on various locations in and around Groningen. The number of professors has increased to three hundred. Over 3000 students enrol each year.

1.3 Assignment

PTT Telecom, the dutch telecommunications company, delivers Private Automatic Branch Exchanges (PABXs) to companies. The management of these PABXs can be done by the company itself, or by PTT Telecom. In the latter case the management is done by the department SOA (Service Op Afstand [dutch]). SOA-engineers can log in by remote in the PABXs and make changes in the configuration. The current SOA application is a text-based one, and the users need to have a lot of technical knowledge to work with the application.

This research takes place at KPN Research, Groningen for the department Service Development and Support (SDS). The project Devine (**De**mo **V**irtual Environments on **N**etwork Management) has the assignment to prove the added value of Virtual Environments applied on Network Management by means of a demonstrator. The network management system SOA is the starting-point for this demonstrator. With this system PABXs can be managed by remote. This research handles the performance management of PABXs and the demonstrator "Virtual SOA" visualises performance data of a PABX in a Virtual Environment.

This research is a continuation of the former research "Telepresence In Shared Virtual Environments" at KPN Research. The guidelines for the design of Virtual Environments which are derived in this former research serve as starting point for this research.

In this research we use the following definitions of a "Virtual Environment" and of "added value". A virtual Environment is

"An interactive, computer-simulated, three dimensional environment which is so realistic that the user has the feeling that he or she is inside this environment."

and the added value is defined as

"The added value on functionality and usability of a new application with regard to the old situation."

Problem definition
Does a Virtual Environment add value to remote management of PABXs?

Objective
Prove the added value of Virtual Environments to Network Management on the basis of a demonstrator which visualises a number of performance data of a PABX.

1.4 Approach

Within this research will be tried to prove the added value of Virtual Environments to Network Management. Therefore there has to be build a demonstrator, called "Virtual SOA". There will be used a standard design path, as illustrated in Figure 1-1, for the design and evaluation of "Virtual SOA".

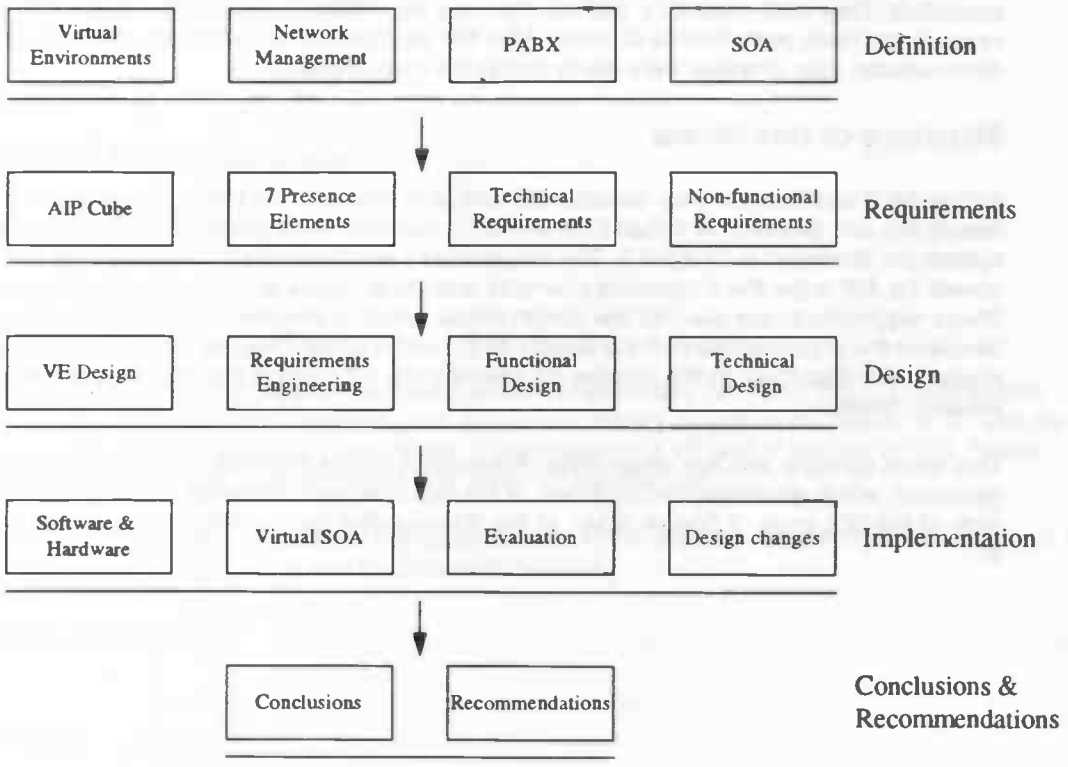


Figure 1-1: Development phases within this research

The first phase is the "*definition*" phase, in which the necessary information and knowledge are studied for this research. The subjects in this phase are *Virtual Environments*, *Network Management*, *the PABX* and *SOA*. The Virtual Environments part is to get insight in what a Virtual Environment (VE) is and which Human Factors are involved in VEs. An introduction to Network Management and PABX is given to provide base information. The system SOA has to be analysed to get insight in the functionalities of the demonstrator which is developed in this research. The analysis of the performance data of PABXs and Network Management provide the necessary information for the performance data which is presented in the demonstrator.

The second phase, the requirements, is very important. In this phase there will be derived guidelines which are used for the design and implementation of "Virtual SOA". First there is a description of the AIP Cube of Zeltzer. This cube defines a co-ordinate system, based on three salient components: *autonomy*, *interaction* and *presence*, which can be used as a qualitative measure of virtual environments. The presence axis has been deepened by Peter Hofstede to get an indication of the measure of presence. This is done from the viewpoint of VEs. In this research we supplement these presence guidelines with guidelines from literature on the subjects of technical and non-functional requirements, to create a more complete list of guidelines.

Before a demonstrator can be implemented, there has to be made a design. The first part of the design is the design of the VE. We design three environments, and select one of these for "Virtual SOA". After the VE design, the requirements engineering is performed. We selected three use cases of tasks to perform with "Virtual SOA" and drafted the user and software requirements for these use cases. The VE design is supplemented with the appliance of the presence guidelines on the SOA system. The functional and technical design then complete the design of "Virtual SOA". The functional design covers the translation of the goal into the demonstrator, based on the use cases. In the technical design this functional design will be translated to a technical specification. In this section the generation of data and the simulation of the data is designed.

After the design is completed the design can be implemented. First a description of the software and hardware is given. The implementation will be done with the 3D modelling tool Superscape VRT. This tool is currently the best commercial available on the Personal Computer platform. The experiences with this tool are described. The resulting demonstrator "Virtual SOA" is presented,

including screen captures. The demonstrator is evaluated with help of Human Factors and VE specialists. They putted remarks, that can increase the understanding of the VE and therefore increase the tasks performance of users. After this evaluation some changes are made to the demonstrator. Also changes were made during the implementation.

1.5 Structure of this thesis

In Chapter 2 and Chapter 3 we describe the definition phase of this thesis. Chapter 2 covers the description and definition of Virtual Environments. Network Management, the PABX and the SOA system are described in Chapter 3. The requirements are described in Chapter 4. This chapter covers the AIP cube, the 7 presence elements and the technical and non-functional requirements. These requirements are used for the design phase, which is described in Chapter 5. Chapter 6 describes the implementation of the design. In the last chapter, Chapter 7, the conclusions of this research are described. In this chapter we describe the subgoals of this research and answer the research question.

This report contains also two appendices. Appendix A covers the source code of the CDR generator, which generates the input data of the demonstrator "Virtual SOA". The most important parts of the SCL code of "Virtual SOA", of the 3D modelling tool Superscape, is given in Appendix B.

2 Virtual Environments

2.1 Introduction

This chapter gives background information about Virtual Reality (VR) and Virtual Environments (VEs). First an introduction on VR is given that among others describes the history of VR and the several types of VR systems. Then follows a description of a VE and of the use of VEs. Finally several examples are given of current and future VE applications.

In Chapter 3 the subjects Network Management and PABX and the system SOA are described, to complete the literature study and the information search.

2.2 Virtual Reality

2.2.1 History

Virtual Reality (VR) is not a new concept. Ivan Sutherland (1965) introduced the key concepts of immersion in a simulated world, and of complete sensory input and output, which are the basis of current VR research [Balaguer].

VR was not introduced to the general public until June 6, 1989 at two trade shows by VPL Research Inc. and Autodesk Inc.. Both companies presented devices and head-mounted displays for interacting with virtual worlds [Balaguer].

VR became one of the hottest buzzwords for the early 1990's. Coined by Jaron Lanier, the term promises hidden new worlds and alternate visions of reality. VR would change the world and how people interacted in it [Choe].

Since then, VR has captured the public imagination and much work has been done to explore the possibilities of VR. So far, it seems that the entertainment market has been the most successful at applying the technology for the mass market. But VR is more often introduced in more serious areas of application such as medicine, chemistry and scientific visualisation [Balaguer].

2.2.2 Definition

The term "Virtual Reality" has different meanings to different people. Most people associate VR with head-mounted displays, data gloves, and position trackers. These are the devices, which often receive the most attention in the media. Some other people stretch the term to include conventional books, movies or pure fantasy and imagination. The devices, while the most visible component of VR, are really but a single component of VR. The devices are the medium of communication with the user, the hardware used to implement a human-machine interface.

At the moment there are several definitions of VR. In Table 2-1 we present some of them.

Source	Definition
US Government Office of Technology Assessment	"Virtual Reality is the popular name for an absorbing, interactive, computer-mediated experience in which a person, or persons, perceives and interacts through sensors and effectors with a synthetic (i.e., simulated) environment, and with simulated objects in it, as if they were real."
Norman Choe in his article "No crash and burn here: A study of the users of virtual world entertainment's Battle Tech"	"Virtual Reality is a simulation in which a computer-generated environment is manipulated in real-time by the observer."
Steve Aukstakalnis & David Blatner in the book "Silicon mirage: The art and science of virtual reality"	"Virtual Reality is a way for humans to visualise, manipulate and interact with computers and extremely complex data."

Table 2-1: Definitions of Virtual Reality

The three definitions all cover the terms "computer", "person" and "interact(ion)", though not always mentioned literally. The definition of Norman Choe only adds the term "environment" to these terms, while the definition of the US Government also adds that VR is an absorbing experience where persons also interact with objects as if they were real. The definition of Steve Aukstakalnis and David Blatner introduces the visualisation and interaction with complex data. For this research, we choose the definition of the US Government Office of Technology Assessment. This definition underlines the media and devices we want to use in this research and the interaction between the user and environment. Although the definition of Aukstakalnis and Blattner covers the indication of visualisation of data and this also will be done in this research, we do not choose this definition. This definition is not general, and the definition we use does not exclude the visualisation of data. Therefore the definition we use in this thesis is:

"Virtual Reality is the popular name for an absorbing, interactive, computer-mediated experience in which a person (or persons) perceives and interacts through sensors and effectors with a synthetic (i.e., simulated) environment with simulated objects in it, as if they were real"

2.2.3 Types of VR systems

In his article *"What is Virtual reality? A home-brew introduction and information resource list"* Jerry Isdale discussed some types of VR systems [Isdale]. He said that a major distinction of VR systems on user interaction is the mode with which they interface to the user. This section describes some of the common modes used in VR systems.

1. ***Window on World Systems***
   Some systems use a conventional computer monitor to display the visual world. This concept traces it's lineage back through the entire history of computer graphics. In 1965, Ivan Sutherland laid out a research program for computer graphics in a paper called "The Ultimate Display" that has driven the field for nearly the past thirty years. "One must look", he said, "as a window through which one beholds a virtual world. The challenge to computer graphics is to make the picture in the window look real, sound real and the objects act real." [Computer Gr.]. So a Window on World System is a system where you hold a window on a virtual world.

2. ***Video Mapping***
   A variation of the Window on World approach merges a video input of the user's silhouette with a 2D computer graphic, called Video Mapping. The user watches a monitor that shows his body's interaction with the world. Myron Kruger has been a champion of this form of VR since the late 60's. He has published two books on the subject: "Artificial Reality" and "Artificial Reality II".

3. ***Immersive Systems***
   The ultimate VR systems completely immerse the user's personal viewpoint inside the virtual world. These "immersive" VR systems are often equipped with a Head Mounted Display. This is a helmet or a face mask that holds the visual and auditory displays.

   A nice variation of the immersive systems use multiple large projection displays to create a 'Cave' or room in which the viewer(s) stand. An early implementation was called "The Closet

Cathedral" for the ability to create the impression of an immense environment within a small physical space. The Holodeck used in the televlsion series "Star Trek: The Next Generation" is a far term extrapolation of this technology.

4. ***Teleoperations***
   Teleoperations links remote sensors in the real world with the senses of a human operator. The operator receives enough sensory feedback to approximate actual presence at the remote side and is able to do different kinds of tasks. In supervisory control modes, a VR interface provides the operator with multiple viewpoints of the remote task environment in a multi-modal display format that can be easily reconfigured according to changing task priorities [Balaguer].
   For example:
   - Fire fighters use remotely operated vehicles to handle some dangerous conditions.
   - Surgeons are using very small instruments on cables to do surgery without cutting a large incision in their patients. The instruments have a small video camera at the business end.
   - Robots equipped with Telepresence systems have already changed the way deep sea and volcanic exploration is done.
   - NASA plans to use telerobotics for space exploration.

5. ***Mixed Reality (Augmented Reality)***
   Merging the Telepresence and Virtual Reality systems gives the Mixed Reality systems. The term Augmented Reallty is also often used and maybe more known. Here the computer generated inputs are merged with Telepresence inputs and/or the users view of the real world.
   For example:
   - A surgeon's view of á brain surgery is overlaid with Images from earlier CAT scans and real-time ultrasound.
   - A fighter pilot sees computer generated maps and data displays inside his fancy helmet visor or on cockpit displays.

6. ***Fish Tank Virtual Reality***
   The phrase "fish tank virtual reality" was used to describe a Canadian VR system reported in the 1993 InterCHI proceedings. It combines a stereoscopic monitor display using LCD Shutter glasses with a mechanical head tracker. The resulting system is superior to simple stereo-Window on World systems due to the motion parallax effects introduced by the head tracker [InterCHI '93].

7. ***"Simulator-platforms"***
   A simulator-platform is a cabin that can move, slant and shake and In combination with screens let people experience travels like a space travel or deep-sea travel [Sala]. In Figure 2-1 an example is given of a motion platform, build by CAE-Link for use with the black Hawk flight simulator. Inside there Is place for about 10 to 40 people, or just a few in case of a flight simulator, who get the illusion that they are really making that trip. Examples of such systems are "Tour of the universe" in Toronto and "Star Tours" in Disney Land or EuroDisney.

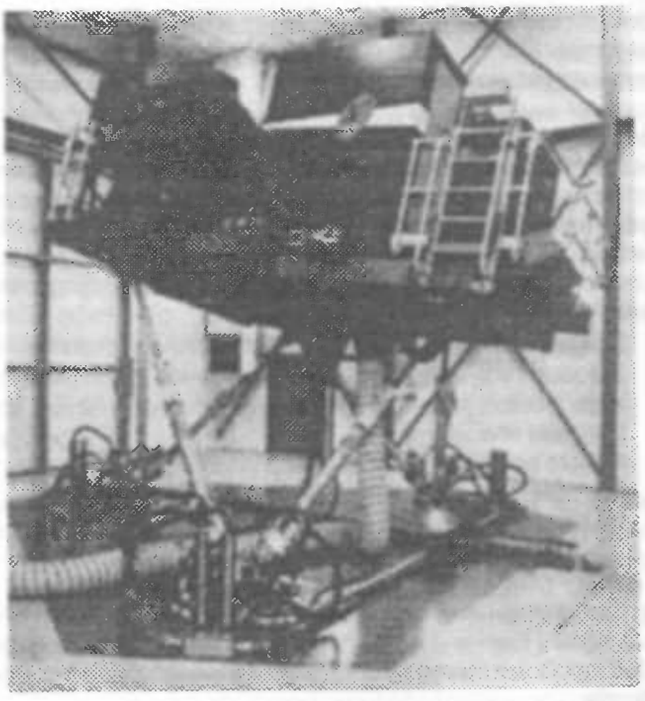


Figure 2-1: An example of a sophisticated motion platform

2.2.4 Devices

As already said in Section 2.2.2, VR is associated with several interaction devices like HMDs and Gloves. This kind of devices provides the interaction between the users of VR applications and the computers. Currently many research labs are working on defining and developing new devices or on improving existing devices. The devices can be globally divided in the groups *control devices*, *tracking devices*, *Head Mounted Displays* and *hand measurement devices*, as done by Francis Balaguer et. al.. These groups are described in the next subsections, and the information is gathered from [Balaguer] and [Pimentel].

2.2.4.1 Control devices

Control devices are used for navigation in VEs. Navigation means that the user of VE-applications can move his viewpoint of the VE by use of control devices. Control devices have different degrees of freedom which cover translations and rotations.

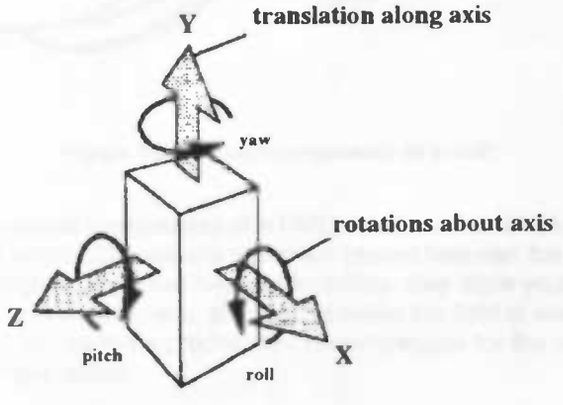


Figure 2-2: The 6 degrees of freedom

Figure 2-2 gives the 6 different directions or rotations in which objects normally can move or be moved. They can move forward or backwards (X-axis), up or down (Y-axis) and left or right (Z-axis), known as translations. In addition objects can rotate about the X, Y and Z-axis, also called roll (Y-axis), yaw (Y-axis) and pitch (Z-axis). All together this results in 6 degrees of freedom.

Known devices here are the simple devices like mouse and joystick, which are 2 degrees of freedom devices, and devices like the SpaceMouse and the Spaceball, which have 6 degrees of freedom.

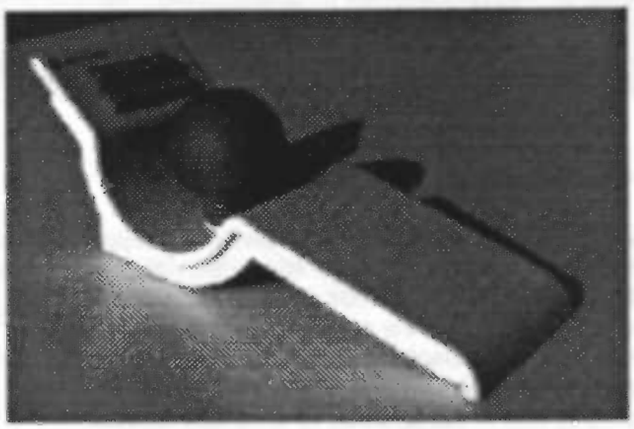


Figure 2-3: The Spaceball by Spatial Systems

The Spaceball, designed by Spatial Systems Inc., is shown in Figure 2-3. It is composed of a rigid sphere containing strain gauges mounted on a plastic base in order to offer a comfortable rest position for the operator hand and arm. The position and orientation are specified by rotating or pushing the sphere in a certain direction. The Spaceball is well suited to moving objects in 3D environments. The metaphor based on the fact that the ball you grab is the object you want to move is easy to integrate, thus requiring reduced training.

2.2.4.2 Tracking devices

Tracking devices are 3D position tracking devices which use *acoustic*, *magnetic*, *mechanical* and *optical* methods for reporting 3D position and orientation. Acoustic systems use the time-of-flight principle to estimate the position of an object in space. Because of the speed of sound varies if ambient air density changes, these systems have poor accuracy over a large range. Furthermore, it is difficult to use them to measure orientation. Mechanical linking systems have been built in the past. Not only have they a limited working range, but the friction inertia of the system and the mechanical linkage attached to the user greatly restrict its motion [Sutherland]. Magnetic tracking devices have been the most successful and the Polhemus 3Space Isotrack, although not perfect, is the most common one.

Figure 2-4: Polhemus source and sensor

A source generates a low frequency magnetic field detected by a sensor. The main problems with the Polhemus, is that its performance is affected by any conducting materials present in the environment. Although most optical tracking systems seem to have failed as commercial products for VR, the method is appealing mainly because it is relatively insensitive to environmental distortions and has a large working environment. The optical tracking system presented by Wang et al (1990) is composed of three cameras on the helmet and an environment consisting of a room where the ceiling is lined with a regular pattern of infrared light-emitting diodes (LEDs) flashing under the system's control. The 2D positions of the LED images inside the field of view are reported in real-time and the 2D image position and the known 3D positions of the LED are used to compute the position and orientation of the helmet in space. Multiple users can be present in the same environment without interfering. The main problems are the weight of the camera assembly, and the fact that the system needs a room where LEDs are installed.

2.2.4.3 Head Mounted Displays

A Head Mounted Display (HMD) is a kind of device which places small video displays in front of each eye, with special optics to focus and stretch the perceived field of view, and has head-tracking facilities. The HMD may be free ranging, tethered, or it might be attached to some sort of a boom armature. They present the general following characteristics:

- headgear with two small display devices (generally LCD colour screens), each optically channelled to one eye, for binocular vision.
- special optics in front of the screens, for wide field of view.
- a tracking system for precise location of the user's head in real time.

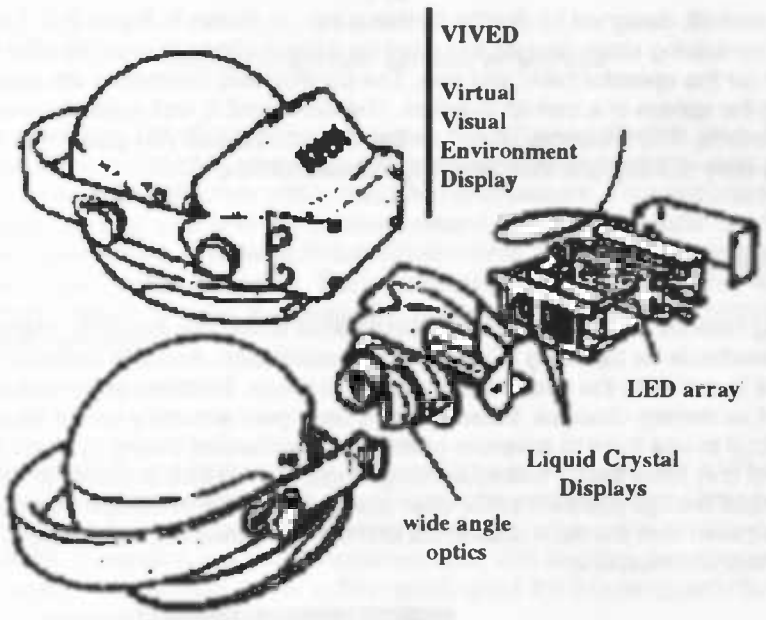


Figure 2-5: Internal components of a HMD

Figure 2-5 presents the internal components of a HMD developed at NASA Ames for their VIVED project. The displays are small LCD screens which are placed between the wide angle optics and the led array. The wide angle optics have two functionalities: they allow you to focus on a display screen two to three inches from your face, and they increase the field of view of the displayed image. The computer output has to be predistorted to compensate for the radial distortion introduced by the wide angle optics.

Figure 2-6: A see-through HMD (left picture)

Figure 2-7: VPL Eyephone (right picture)

Figure 2-6 presents a see-through HMD, designed by the University of North Carolina. A see-through HMD is used for augmented reality. The user sees a VE and also his own environment. The VE is merged with the real environment. The VPL Eyephone is showed in Figure 2-7. A user sees now only the VE.

2.2.4.4 Hand measurement devices

Hand measurement devices are glove devices for the measurement of hand movements. They must sense both the flexion angles of the fingers and the position and orientation of the wrist in real-time. Image based techniques have been used to measure both channels directly from observation of user movement. However, since interpretation of the moving images is required, this technique tends to be slow and imprecise, even when LEDs are used to identify key points on the user to simplify image processing task. The Z-glove from VPL Research Inc. performed position tracking with electronic transducers mounted on side of the palm. This method of tracking fails if there is not a direct line-of-sight between the transducers and the set of three receivers.

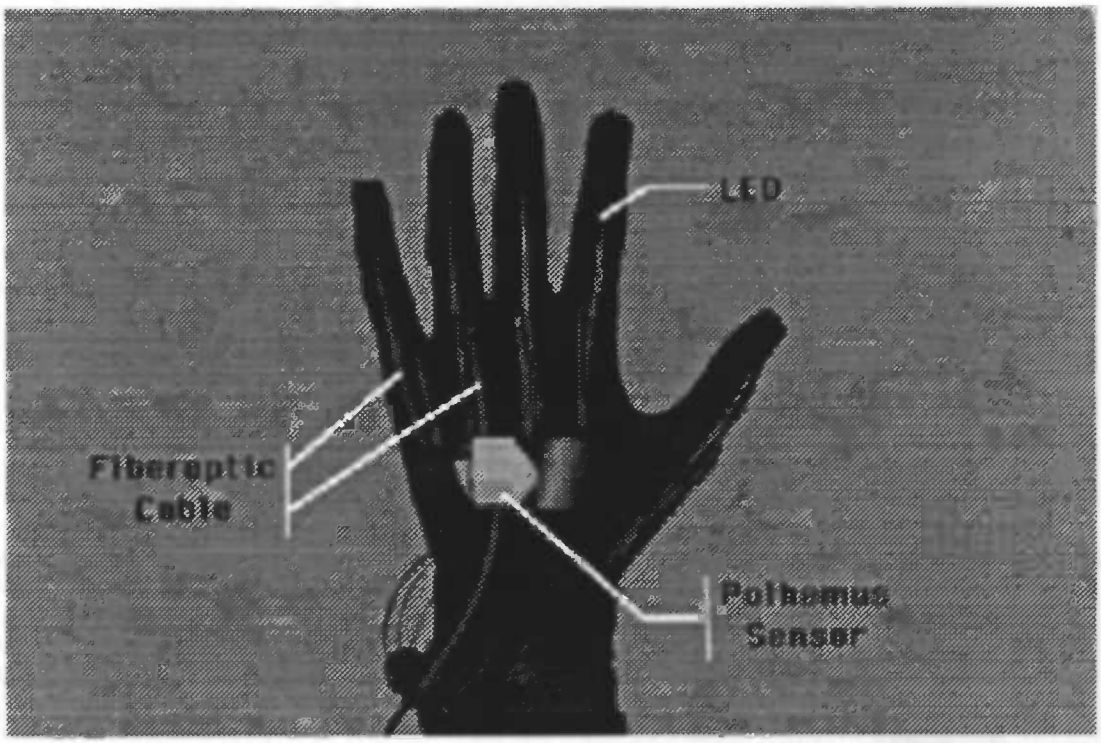


Figure 2-8: VPL DataGlove

The most common hand measurement device currently is the DataGlove from VPL research Inc. The DataGlove consists of a lightweight nylon glove with optical sensors mounted along the

fingers. In its basic configuration, the sensors measure the bending angles of the joints of the thumb and the lower and middle knuckles of the other fingers, and the DataGlove can be extended to measure abduction angles between the fingers. Attached to the back is a 3Space Isotrack system to measure orientation and position of the gloved hand. This information, along with the ten flex angles for the knuckles is transmitted through a serial communication line to the host computer.

2.3 What is a Virtual Environment?

In the previous section we described what Virtual Reality is and what kind of systems exist. Now we first give a definition of what a VE is, and then we describe the features *immersion* and *presence*.

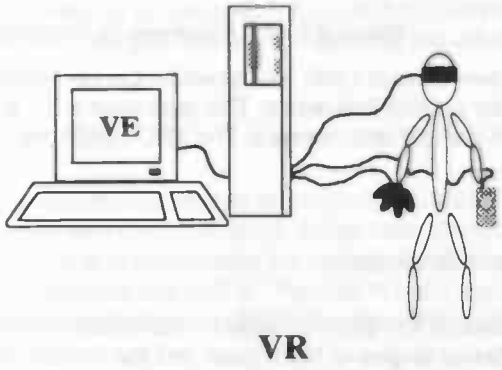


Figure 2-9: Relation between VR and VEs

A VE can be seen as a part of VR, as shown in Figure 2-9. VR is a collective noun of and popular name for any absorbing, interactive, computer-mediated experience in which users perceive and interact through sensors and effectors with a simulated environment. Four important components are then *computer*, *devices*, *user* and *Virtual Environment*. The user interface of a VR application, i.e. the simulated environment, is called a Virtual Environment. The Virtual Environment is displayed on the monitor and on the HMD. The user interacts through the devices with the Virtual Environment. The computer handles the interaction between the user and the Virtual Environment and displays the environment.

2.3.1 Definition

A Virtual Environment (VE) is a kind of user interface. It is a user-interface of a VR simulation and users get immersed in it, which means that one or more of a user's sensors (eyes and ears generally) are isolated from the surrounding environment and fed only information coming from the computer [Pimentel]. Immersion is a main characteristic of VEs and this, together with the definition of VR which is mentioned earlier in this report, gives the following definition of a VE which we will use in this research:

"An interactive, computer-simulated, 3D environment which is so realistic that people get immersed in it"

2.3.2 Presence

Immersion into the data space provided by a computer, and the feeling of really being there or "presence", are commonly acknowledged as the uniquely important features of virtual reality environments [Psotka]. Most people who have experienced different VE systems state that they felt either "part of a synthetic experience" (presence) or that they felt "immersed in the experience" (immersion).

To feel presence means that although one feels immersed to some extent in the experience, one is also aware of the outside world [Kalawsky]. The user has the feeling of moving through the VE and interacting with the objects or persons within the VE, but also notices changes in the outside world.

For example, a user notices when another person enters the room. If the user is completely immersed, he will not notice that someone enters the room. There are two kinds of presence. The first is to experience presence within a completely simulated environment, for example in a simulator platform like "Star Tours", when you have the idea of travelling through space. The second is Telepresence. Telepresence aims to simulate presence of an operator in a remote environment to supervise the functioning of the remote platform and perform tasks controlling remote robots. This is already mentioned in Section 2.2.3, at the Teleoperations mode. You can think here of the control of nuclear plants or other dangerous operations.

2.3.3 Immersion

How immersed one feels appears to be determined by a complex set of physical components and affordances of the environment, and as yet poorly understood psychological processes [Psotka]. Pimentel and Teixeira say that the experience of being immersed in a computer-generated world involves the same mental shift of "suspending your disbelief for a period of time" as "when you get wrapped up in a good novel or become absorbed in playing a computer game." [Pimentel, p. 15].

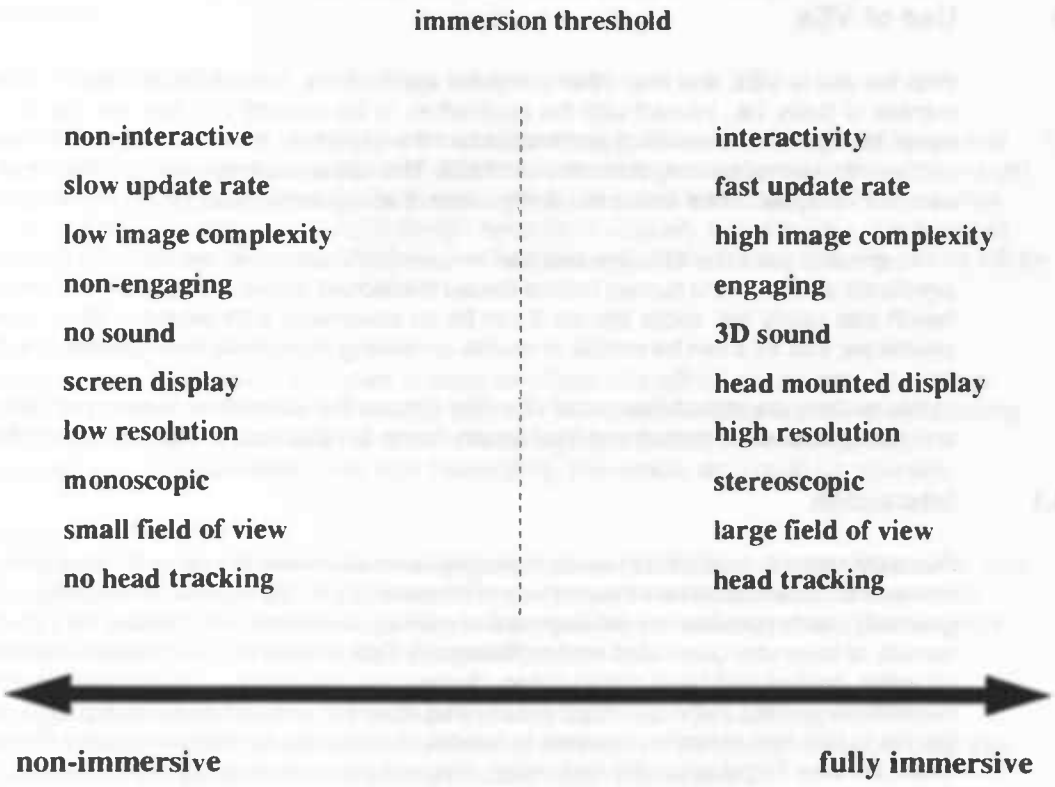


Figure 2-10: degree of immersion

Immersion is a measure of the degree of realism achieved by a certain virtual reality device. The more the user can not discriminate between the VR experience and the reality, the more the immersion is obtained. Figure 2-10 gives a diagram showing factors that govern the degree of visual immersion, according to Pimentel and Teixeira [Pimentel, p. 105]. On the left side are the factors that do not influence the degree of immersion in a positive way. The factors on the right side of the diagram do influence the degree of immersion in a positive way. Below we describe these factors.

Humans feel immersed in their real life. In the real world we *interact* with other people or machines. Our visual system does not notice changes in the environment because all interactions are continuous processes. A car moves continuously closer to you and not with steps of for example 10 meters. In VEs a *fast update rate* is needed to achieve the same effect. The complexity of our view is very detailed at a close distance and less detailed at a long distance. A

tree at a long distance is a green sphere on a brown stick, at a close distance you see the leaves and the bark of the trunk. In VEs a *high image complexity* is needed to let users see all these details. Sounds come from all distances, think of a train or a car which is passing by. It sounds very different when it is moving towards you or away from you (*3D sound*). Humans also have a large field of view in the real world. With a *HMD* users of a VE-application can only see the VE and not the outside world. The *high resolution* of the images, the *stereoscopic view* and the *large field of view* make the VE look closer to reality than if they are not provided with the VE. With a high resolution images can be more detailed and look more realistic. The stereoscopic view makes the user see depth in the images. A large field of view corresponds with the large field of view we have in real life. At last the *head tracking* of a HMD gives the user the freedom of looking around, just like in the real world.

Pimentel and Telxeira say that it might not even be possible to understand immersion by reducing it to its components, which are then independently studied. Instead, a holistic technique might be necessary because the experience of immersion is more than just the mere sum of its parts. The point is that you can't just focus on one factor while ignoring the others. An effective VR application balances all the issues.

2.4 Use of VEs

With the use of VEs, and also other computer applications, comes that the user has to perform a number of tasks, i.e., interact with the application. In the case of VEs the user has to navigate and travel through the environment and manipulate the objects in the environment. The use of VEs involves the use of sensory devices like HMDs. The use of such devices can influence the health of users of VE applications and is not always safe. It also can have social impact on users.

If VR systems are to be effective and well received by their users, researchers need to focus significant efforts on the human factors issues mentioned above, *interaction*, *human performance*, *health and safety* and *social impact*. It can be an advantage, with regard to other *human computer interfaces*, that VEs can be worlds or worlds containing metaphors derived from the real world.

In this section, we look at the use of VEs. We discuss the interaction, human performance, health and safety and social impact and the Human Computer Interface in the next subsections.

2.4.1 Interaction

Recently, various specialised research programs on advanced concepts for human-machine interaction have focused on the problem of interacting with 3D worlds. These programs have gradually made possible the development of new input devices and displays for interacting with remote or computer generated worlds [Balaguer]. Rather than keyboard input, interaction is based on voice, gesture and hand manipulation; displays are rethought to closely match human vision capabilities yielding in the development of head mounted or head coupled display concepts. The goal is to simulate operator presence in remote or computer synthesised worlds [Balaguer] and to make the user forget about the technology involved and immerse you in the simulation itself [Choe].

In exploring a virtual world, the user must perform a number of tasks, including navigation, travelling and object manipulation [Baker]. These three tasks are described below.

- ***Navigation in VEs***
  VR differs fundamentally from the desktop in that the user can travel inside the data. Just as it is easy to get lost within the forest, immersion complicates navigation because we lose our overall sense of location. A fundamental task is to develop a mental model of the overall virtual space , that is a "cognitive map", for successful travel and interaction with the environment. A cognitive map encodes the relative locations as well as the attributes of phenomena in spatial environments. A good map supports all the other spatial navigation tasks, such as searching for a target object, or travelling to a target spot. And it aids in maintaining situation awareness, "where am I" in the overall context.

- ***Travel***
  Travel through a virtual environment presents a number of challenging cognitive and perceptual-motor, i.e. the process of perception, issues, because most of the constraints that

accompany locomotion through physical space are removed. Travel metaphors should consider what we know about human judgements of motion and orientation in the natural world. In designing travel mechanisms, the stability of the perceptual-motor loop, i.e. the continues process of perception, must be maintained. The process of perception should continue. Disruption or distortion of the natural coupling between body movement and view leads to disorientation and confusion for the user. Minimising the latency between user movement and change in view is crucial. John M. Airy et. al. found that maintaining a frame update rate of at least 6 frames per second was necessary for users to stay immersed in the VE [Airey]. Chris Esposito suggests a frame rate of at least 12 frames per second [Esposito].

- ***Object manipulation***
  The perceptual and motor skills involved in reaching for, grasping, and manipulating objects are considerably different from those involved in travel. Correspondingly, the design choices for mechanisms supporting object manipulation differ from those supporting navigation and travel. Travel demands an accurate representation of motion, an accurate depiction of spatial relations, and presentation of visual features rich enough to convey a sense of spatial extent. On the other hand, motion fidelity is less important for object inspection or recognition. Texture gradients and stereoscopic views play relatively less important roles than do light and shadowing.

2.4.2 Human Performance

In her article Kay M. Stanney said that human performance in VEs will be influenced by several factors, including: task characteristics; user characteristics; design constraints imposed by human sensory and motor physiology; integration issues with multi-modal interaction; and the potential need for new visual, auditory and haptic design metaphors uniquely suited to VEs. It is a goal to maximise the human performance efficiency in VEs [Stanney]. In the following these factors will be described.

- ***Task characteristics***
  It is important to determine the types of tasks for which VEs will be appropriate. In order to obtain this understanding the relationship between task characteristics and the corresponding Virtual Environment characteristics which effectively support their performance (e.g., stereoscopic 3D visualisation, real-time interactivity, immersion, etc.) must be attained.

- ***User characteristics***
  An important aspect influencing human VE performance is the affect of user differences. User characteristics that significantly influence VR experiences need to be identified in order to design VR systems that accommodate the unique needs of users. Important characteristics here are level of experience and spatial orientation.

- ***Design constraints imposed by human sensory and motor physiology***
  The physiological and perceptual issues which directly impact the design of multi-modal VEs include visual perception, auditory perception and haptic perception.

⇒ *Visual perception*
The design of visual presentations for VEs is complicated because the human visual system is very sensitive to any anomalies in perceived imagery, especially in motion scenes. In general, human visual perception needs to be better understood in order to ensure that the most effective visual scenes are developed for virtual worlds.

⇒ *Auditory perception*
In order to synthesise a realistic environment it is important to obtain a better understanding of how ears receive sound, particularly focusing on 3D audio localisation. Because not everything is yet understood, for example the discrimination of sounds from front to back, much work is needed in order to effectively synthesise 3D auditory environments.

⇒ *Haptic perception*
A haptic sensation is a mechanical contact with the skin. Three mechanical stimuli produce the sensation of touch: a displacement of the skin over an extended period of time, a transitory (few milliseconds) displacement of the skin, and a transitory displacement of the skin which is

repeated at a constant variable frequency. Even with this understanding of global mechanisms, however, the attributes of the skin are difficult to characterise in a quantitative fashion. This is due to the fact that the skin has variable thresholds for touch (vibrotactile touch) and can perform complex spatial an temporal summations which are all a function of the type and position of the mechanical stimuli.

- ***Integration Issues with multi-modal Interaction***
  While developers are focusing on synthesising effective visual, auditory and haptic representations in virtual worlds, it is also important to determine how to effectively integrate this multi-modal interaction. With multi-modal is meant the use of several modalities, like visual, auditory and haptic perception, at a time. This multi-modal interaction may be a primary factor that leads to enhanced human performance for certain tasks presented in virtual worlds. Early studies have already indicated that sensorial redundancy can enhance human performance in virtual worlds. There is currently, however, a limited understanding on how to effectively provide such sensorial parallelism.

- ***Virtual Environment design metaphors***
  It is known that well-designed metaphors can assist novice users in effectively performing tasks in human computer interaction. Thus, designing effective VE metaphors could similarly enhance human performance in virtual worlds. Such metaphors may also be a means of assisting in the integration of multi-modal interaction.

2.4.3 Health and safety

There are several health and safety issues which may affect users of VEs. These issues include both direct and indirect effects [Viirre]. The direct effects can be looked at from a microscopic level (e.g., individual tissue) or a macroscopic level (e.g., trauma). The indirect effects are primarily psychological. It is a goal to minimise these effects.

Virtual Reality with Virtual Environments seems grandiose, but there are also some safety problems because of side effects of these applications. To date there is still little published literature regarding the safety issues. All of the harmful reactions to VR may be classified under the term cyberpathology. Reported harmful reactions include *physical pathologies* and *Cybersickness* [Wantland]. We describe these two reactions below.

Physical Pathologies:
- *Repetitive stress* injury is a rather common result of extensive VR use and results from continually repeated movements.
- An *immersion injury* is any physical trauma occurring during VR use. Users may become so involved and engrossed in their virtual world that they become unaware and even possibly disoriented with their immediate surroundings.
- *Transmittable disease*: VR applications are potential fomites (objects which are able to harbour pathogenic organisms and thus may serve as an agent of transmission of an infection, see [Dorland, p 647] , because the equipment is used by multiple individuals without disinfecting or hygienically wiping.

Cybersickness: a variant form of motion sickness which has harmful effects upon the visual, neural and psychological status.
- *Visual effects*: There have been reported 3 primary visual effects: nausea, asthenopia (eyestrain) and the impact of electromagnetic fields upon the eye structures.
- *Neural effects*: The impact of electromagnetic fields upon the Central Nervous System (CNS) and the body and the occurrence of "flicker vertigo", a condition in which individuals suffer from seizure when a flickering light is observed, in the CNS [Viirre].
- *Psychological effects*: The problem of anxiety, for example experiencing acrophobia when a VR user is on a top of a mountain in a VR simulation or claustrophobia when wearing a darkened HMD before the immersion begins.

2.4.4 Social impact

Virtual Reality is a technology, which like its ancestors (e.g., television, computers, video games) has the potential for negative social implications through misuse and abuse. Through a careful analyses, some of the problems of VEs may be anticipated and perhaps prevented. A proactive ,

rather than reactive, approach of analysis may allow researchers to identify and address potentially harmful side-effects related to the use of VE technology [Stanney].

There are many open Issues , such as: how will interaction in the virtual world modify behaviour, will people turn their backs on the real world and become "contented zombies" wandering around synthetic worlds which fulfil their whims but disregard their growth as a human being, will people avoid reality and real social encounters with peers and become addicted to escapism. These issues need to be proactively explored in order to circumvent negative social consequences from Human Virtual Environment Interfaces [Stanney].

We must not underestimate the enormous effect that this kind of media has on the society. Some people may see it as a terrific tool, others do not want anything to do with it and are possibly cut of of the influences that VR has on our daily life. It can influence the way we work and also the entertainment at night and the way we react when we meet other people. It is possible that VR will change our behaviour considerably [Sala].

2.4.5 Human Computer Interface

The last decade has been marked by the development of the computer as a tool in almost every domain of human activity. One of the reasons for such a development was the introduction of human-friendly interfaces which have made computers easy to use and learn [Balaguer].

The world we live in is a three-dimensional (3D) world. Because we are used to live in this 3D world you could say that we are experienced in interacting with 3D worlds. Our sense of sight enables us to orient ourselves in space, to move forward and to communicate. Our mind is highly trained to interpret images and communication is often more effective through images, graphs, diagrams than through words [Balaguer]. VEs are simulated 3D worlds, so they have a 3D interface. In VEs, head-tracking or hand-tracking is often used to establish position. This makes navigation within the 3D model world particularly natural, and dramatically easier than trying to manoeuvre around three dimensions using screen-based, 2D controls [Baker]. Interaction with objects in the data should also be more familiar in the physically 3D spaces provided by most environments.

You could say that it is a goal to create a Human Computer Interface that reflects the way we interact with the real world by using our senses sight, hearing and touch. Virtual Reality is a powerful new technology that is pointing in this direction. In a Virtual Environment the participant establishes a direct connection with its virtual senses. The participant does not have to learn how to extract information from the medium [Jacobson].

2.5 Examples of VE-applications

Below follow a number of examples of VE-applications and areas in which they can be applied, both current and future applications.

Current applications:

Architectural design: When MIT's new School of Computer Science building was designed, a virtual version was created. It was used to explore the architectural structure prior to constructing the building. It was determined that a wall was not right and a visit to the Virtual World convinced the architect to make some adjustments to the design. Other virtual buildings have been used for advertising. Exploration of Virtual Worlds has limitless applications.

Virtual Wind Tunnel: A Virtual Wind Tunnel is an environment for visualisation and interface for exploring and understanding the output of Computational Fluid Dynamics. Virtual fluid flow around a body is simulated. Tasks that are simply impossible to do in a real wind tunnel can now be experimented with in the Virtual Wind Tunnel. This application is being implemented at NASA Ames Research Centre.

Simulators: Virtual Reality was born as a flight simulator for improving AIR FORCE pilot's abilities in missile launching. Another such environment called SIMNET, is a tank for combat simulation. This is the most widely distributed environment today. Advanced aeroplane cockpits are being developed at Human Interface Technology Laboratory at University of Washington.

Future applications:

In *scientific visualisation*, the VR interface can help scientists explore the multi-dimensional graphical representation of their data at various levels of detail using interactive camera control and stereotopic displays [Balaguer]. The scientist is free to fly within his simulation data and therefore to focus interest on any part of its simulation, to study it in detail or to allocate supercomputer time for a more precise simulation.

Finger-painting in 3D: Humans will be able to put on the virtual reality equipment and paint/sculpt something in mid-air. There will be an unlimited supply of paints and tools to create any work of art.

Keyboard class without keyboards: No more worrying about crumbs in the keyboard! The same equipment used for the 3D painting will be used to teach keyboarding. The students will simply type on the desktop while viewing a keyboard in virtual reality. Their work will be monitored by the computer and a tutor will be inside the VR world to provide personal instruction to each student.

Walking the moon from Earth: On July 20, 1969, man walked on the moon for the first time. With virtual reality, man will be able to walk on the moon at any time. Virtual reality will give humans the opportunity to experience the moon's gravity, surface, and more.

Zooming around an atom's nucleus on an electron: How fast does an electron travel around an atom's nucleus? What type of orbit does it take? Does this orbit stay the same or does it change? People will be able to hop on an electron and travel with it to answer these and many other questions. Virtual reality will allow humans to conceptualise (not just attempt to conceptualise) what goes on in the atomic world.

Travelling through the human body on a platelet: The biological sciences will never be the same again! Travel through the human body on a platelet (in the bloodstream) and see what goes on inside you! Wouldn't it be great to see what happens to that cookie you just ate?

Participating in WW I or WW II: Do you even remember what war this battle was a part of? If you fought in it you would surely know. Humans will be able to fight in World War I or II. No harm will come to the people (in real life), but they will have a better understanding of the conditions of the time.

2.6 Conclusions

Since the introduction of VR to the general public much work has been done to explore the possibilities of VR. It seems that the entertainment market has been the most successful at applying the technology for the mass market, but VR is more often introduced in more serious areas of application such as medicine, chemistry and scientific visualisation.

The uniquely important features of VR applications are immersion and presence. With presence is meant that although one feels immersed to some extent in a virtual world, one is also aware of the outside world. With real immersion one is not aware anymore of the outside world.

To avoid overall sense of location because of immersion, it is fundamental that users can develop a cognitive map for successful travel and interaction with the environment. Travel through a VE presents a number of challenging cognitive and perceptual-motor issues because most of the constraints that accompany locomotion through physical space are removed. Therefore travel metaphors should consider what we know about human judgements of motion and orientation in the natural world.

It is a goal to maximise the human performance efficiency in VEs. Therefore it is important to determine if VEs are appropriate for the kind of tasks you want to perform. In general you should take the kind of user interface which suits best with the task. Also you should consider the influence of user characteristics on VR experiences. These characteristics should be identified in order to design VR systems that accommodate the unique needs of users. On the subject of human sensory and motor physiology still a lot of work has to be done, especially on the visual and auditory perception. These subjects are very complicated and not everything is understood yet.

Also there is still a limited understanding on how to effectively provide sensory parallelism. Early studies have already indicated that sensorial redundancy can enhance human performance in virtual worlds. What already is known is that well-designed metaphors can assist novice users in effectively performing tasks in virtual worlds.

On the subject of health and safety work has been done, but still little is published. It is preferable that this work continues and that the health and safety effects will be minimised or disappear. A lot of work has to be done on the social impact of VEs on users. There must be tried to avoid negative social consequences. This can be very hard. The consequences will probably become clear after the introduction of VR in the society, companies and households.

An advantage of VEs is that we are used to interact with 3D worlds. It is a goal to create VEs that reflect the way we interact with the real world by using our senses sight, hearing and touch. A disadvantage is that the input and output of interaction devices not always correspond with reality. Fortunately these devices are still improved and new devices are designed.

3 Network Management, PABX and SOA

3.1 Introduction

PTT Telecom manages Private Automatic Branch Exchanges (PABXs) of many of her customers and strives as much as possible to do this management by remote via Public Switched Telephone Network connections. This is cheaper and faster than management at location. According to PABX-engineers about 80 percent of the management can be done by remote with software.

For the management of PABXs, PTT Telecom uses a Network Management system called Service Op Afstand [dutch] (SOA). This is a technical tool for and a protected interface to the PABX. Currently the system is extended and renewed on functionality and capacity within the project "Redesign SOA". The renewed system SOA is the starting point for the demonstrator in this research. In the "Redesign SOA" project PTT Telecom has the following vision on this system SOA:

> *"SOA is a flexible technical tool for the protected data communication between all of the by remote to manage PABXs and the Telecom organisation, in spite of this organisations' structure."*

This chapter contains background information on Network Management , PABXs and SOA. First we describe Network Management and we explain what a PABX is. We can use this knowledge to describe what SOA is and how it works. Finally in Section 3.5 we chose a target group for "Virtual SOA" and the management category we support with "Virtual SOA", Performance Management. In Chapter 4 research is done to guidelines for the design of a VE and these are applied in the design phase of "Virtual SOA".

3.2 Network Management

In this section we give information on Network Management and why it is necessary. Also a classification of the functionality of Network Management systems is given.

The management of an integrated telecommunications network is of crucial importance to the organisation that depends on that network [Abrahams]. The technology for the design and implementation of large, integrated networks is becoming available but managing these networks is a serious challenge. In order to manage a major network appropriately, the planners must adopt an active, rather than a reactive, attitude to problem-solving. Network Management itself must be both integrated and centralised. Network Management must provide a common point of reference for user problems. It must have the capability of system-wide monitoring from a central location and it must eventually have the ability to anticipate problems before these reach alarm levels [Abrahams].

The OSI Standard[1] is a Network Management Standard. This standard is used increasingly in the telecommunications industry. The OSI Standard has classified the functionality required of Network Management systems into the 5 categories of: *Configuration Management*, *Fault Management*, *Performance Management*, *Security Management* and *Accounting Management* [Black]. Below we give a description of these categories, to let you construct a map of these categories.

[1] OSI Standard stands for Open Systems Interconnection Standard. This standard is a Network Management standard for telecommunications networks.

Configuration Management covers identifying the functionality of the network, gathering data about it and providing it with data. These activities are related to the data of all network components, so at every moment the network configuration is known. Tasks to distinguish are:

- Take care of network modifications and - expansion.
- Adjust the network parameters.
- The managing of the status of network components.
- The registration and representation of the network components and the status of it.

Fault Management is the detection, the report, the analysis and correction of abnormal functioning of the network. It is recommended that faults are detected in an early stage and handled quick and accurate, so that the network stays In optimal condition. Tasks to distinguish are:

- The detection of faults (alarms) (threshold crossings, test results).
- Make a fault diagnosis.
- Registration of faults.
- Controlling of the removal of faults.
- The prevention of faults, for example by testing the suspicious parts of the network.
- The initialising of activities and procedures after an escalation.
- The assigning of faults to the departments which serve to solve the problems.

Performance Management contains the activities for reporting and evaluating of the performance of the network. It includes gathering and next analysing of the data about the functioning of the network. The result of these activities is that insight is obtained if the performing of the network Is adequate or not. Tasks to distinguish are:

- Gathering of network statistics (response time, traffic behaviour, average recover time after alarms/fault reports)
- Analysing and presenting of network statistics (reports)

Accounting Management covers the activities of registration of the use of the telecommunication network and fix the costs of it. Owing to this the network can be exploited in a economic responsible way. Tasks to distinguish are:

- The calculation of the costs of use of the telecommunications network.
- The generation of surveys of costs.
- The promotion of the awareness of costs among the users.
- The guarding of budgets of organisation units.
- The Impute of fixed and variable costs.
- The adjustment of cost parameters.

Security Management covers the management of the access to the telecommunication network. Unauthorised use of the network is impossible or restricted and the data in the network are secured. Tasks to distinguish are:

- The establishing of identification and authentication possibilities (distribution, control of passwords or other keys).
- The establishing of authorisation rules (who is authorised for what).
- Making backups of data via the network.
- The control of security supplies.
- Drafting procedures and looking on the observance of it.

Current user interfaces typically consist of hundreds of forms that need to be filled in order to configure and update the network. SOA has such an interface. Users get lost in piles of forms In the absence of effective organisation and information visualisation tools. "A fundamental feature of an advanced Network Management station is the capability to present to the human manager a comprehensible picture of the relevant scenarios" [Cons].

3.3 PABX

In this section the PABX is described. Also CDRs are discussed and when they are recorded.

A Private Automatic Branch Exchange (PABX) is a telecommunications switching and control system that is installed on the customer's premises to provide internal electronic communication services and is also connected to the public telecommunications networks. With this Automatic Exchange the users dial their own calls. In some case the abbreviation PBX is used. We do not

use this abbreviation because there can be confusion with the older, manual, exchange (sometimes abbreviated to PMBX).

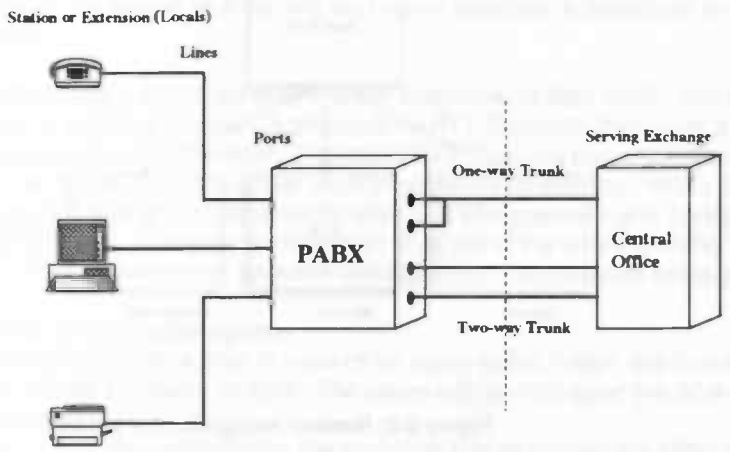


Figure 3-1: Basic parts of a PABX

In Figure 3-1 the most important connections with a PABX are presented. A PABX is connected with the public telecommunications network by so called *trunks*. These trunks are analog or digital. The trunks are digital if they are connected to a digital network or ISDN (Integrated Services Digital Network). The trunks are one-way trunks, in case the communication data only goes from the PABX to the public network, or two-way trunks, if the data can go both ways. Internally, i.e. connections with the PABX that do not come from the public network, several *stations* are connected with the PABX. These links between stations and the PABX are called *lines*. A station is a voice or data terminal, like a telephone, a computer, a printer, a data workstation or facsimile machines. There is also a possibility for connecting a PC directly to the PABX with a RS232 serial connection. Such a PC can be used for the management of the PABX at location.

PABXs record so called Call Detail Records (CDRs). A CDR is a data structure with fields containing information about the call which is made. The most important fields are the *A-Number*, *B-Number*, *date*, *starting time*, and *duration*. The A-Number Is the number of the person who makes the call, the caller. The number of the person who receives the call, the receiver, is stored in the B-Number. The date is the date when the call is made, the starting time is the time when the call is answered and the duration is the time the answering takes.

These CDRs are not always recorded. It depends on the result of the dialling process. Below we first describe how the PABX "recognises" a number and then how the flow of the dialling process continues after recognition of the number.

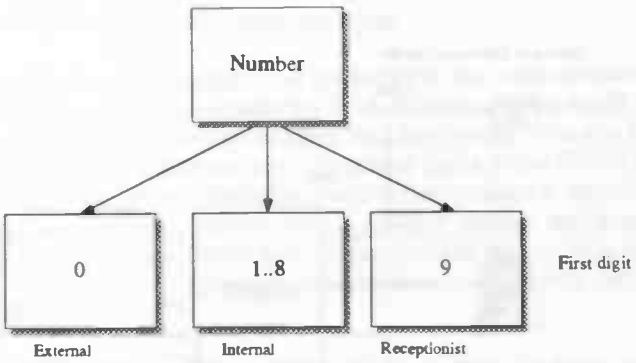


Figure 3-2: Number recognition by a PABX

The recognition of a number from a station connected to a PABX is done as presented in 1st. If the number starts with a digit in the range 1 to 8 it is an internal number, if it starts with a 0 it is an external number. A 9 is to call the receptionist. If the number is an external number, it is passed to the public network.

If the caller dials a non-existing number he has to hang up and dial again, because the PABX can not make a connection with that number. When the number exists the PABX can locate the receiver. Below the dialling process of a PABX is described when a number is existing.

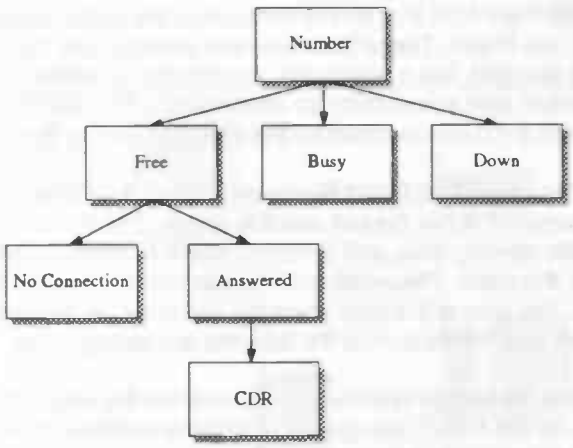


Figure 3-3: Dialling process of a PABX

In Figure 3-3 the dialling process is visualised when the number exists. There are three possibilities. The line of the receiver can be *down*, *busy* or *free*. In case the line is down and the number you dialled is external, the only thing that can be done is checking if the number still exists at 06-8008. If the line is down and the number is internal, the internal PABX manager or the MAC (Meld en Analyse Centrum [dutch]) has to be contacted. The MAC is described in Section 3.4. If the line is busy you just have to wait and try it again later. In case the line is free the call can be answered or not. If nobody is at home or at the office, the only thing you can do is try it again later. In case it is answered you have the call you wanted and a CDR is kept.

3.4 SOA

In this section the SOA system environment is briefly described. Also the operating environment of the SOA system and the flow of fault reports and report handling is described, to give a complete impression of SOA.

An organisation that buys a PABX for their internal telephone or data traffic, can also decide to subscribe to SOA. A SOA-subscription is provided by PTT Telecom. With such a subscription the PABX will be managed by PTT Telecom. Therefore PTT Telecom has a special department called the SOA Centre. At the SOA Centre Local SOA Engineers manage the PABXs by remote with the SOA-system. The SOA-system is designed for the Fault Management and Configuration Management, by software. The management with SOA will in the future possibly evolve to management on other management domains among which Performance Management.

There are two kinds of SOA-subscriptions:
1. *A subscription with a Service Box*: in case of an alarm in the PABX, the Service Box automatically passes the alarm to SOA. The alarm will be noticed at the SOA-centre and the fault will be handled.
2. *A subscription with a Service Modem*: the customer has to contact the MAC (Meld- en Analyse Centrum [dutch], Report and Analyse Centre) when an alarm appears and the MAC will take actions to solve the fault. It still can be handled remotely.

3.4.1 SOA system environment

As already mentioned in the introduction, the SOA system is currently extended and renewed. The functionality and capacity of the SOA system will be improved. The project "Redesign SOA" also makes changes to the architecture of SOA to achieve its goal. The new SOA system environment is given in Figure 3-4. The SOA System is represented shaded in this figure.

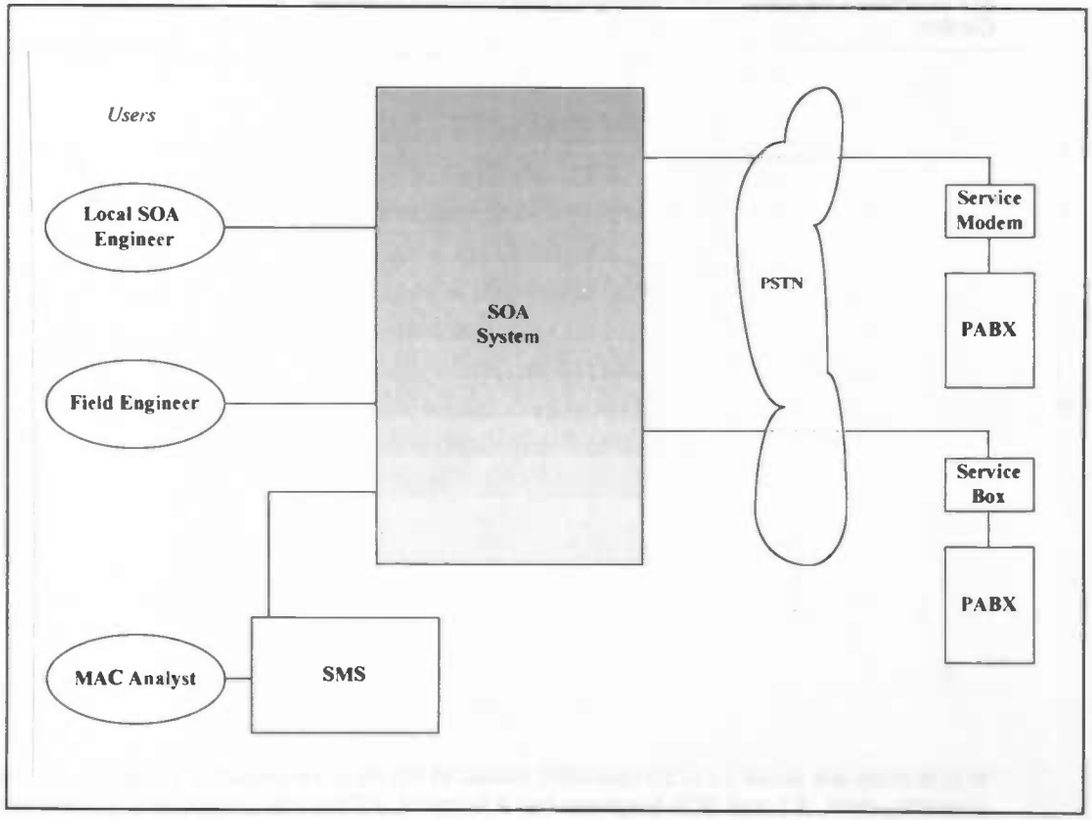


Figure 3-4: Planned SOA-system environment after "Redesign SOA"

As shown in Figure 3-4 several users can log on on the SOA System. After logging on on SOA the user can build PSTN connections (connections with the public network) with PABXs of customers.

These connections are made to a Service Box or Service Modem. When the connection is made the user can start up the management system corresponding to that kind of PABX. These management systems are delivered by the PABX-suppliers. With these tools the user can search for faults, solve them and/or make configuration changes.

The access to the SOA System is protected. The privileges of the various user-groups are recorded in the SOA System. The contact with the SOA System can be made both local as via a telecommunication connection. The Service Modem and the Service Box are the interface between the PABX and PSTN. The Service Box signalises faults in the PABX and automatically generates reports to the SOA System. The SOA System then makes a report in SMS of the alarm. SMS stands for Service Management System, and is a report archive and management system. All performed activity's and services of SOA are passed to SMS by the SOA System. SMS adds priority to the reports on base of the customer's subscription, and spreads the reports over the available SOA-engineers.

The users of the SOA system are the MAC analyst, the Local SOA Engineer and the Field Engineer. The MAC-analysts are in general low educated and have the direct phone contact with the customers. Currently they can not execute actions on the PABXs of customers with the current SOA/SMS-systems. They only can retrieve, in a limited and indirect way, information about a PABX. The engineers, Local SOA Engineers and Field Engineers, are in general higher educated technical experts. They can perform actions on the PABXs of the customers, like for instance locate errors, solve small faults, change the number of a telephone and retrieve the state of the PABX. The current SOA-system provides in a lot of functionalities for these kind of actions. At the moment the services are limited to Fault Management but in the future also other kinds of management will be offered.

3.4.2 SOA operating environment

In this section we describe the operating environment of a Local SOA Engineer, at the SOA Centre.

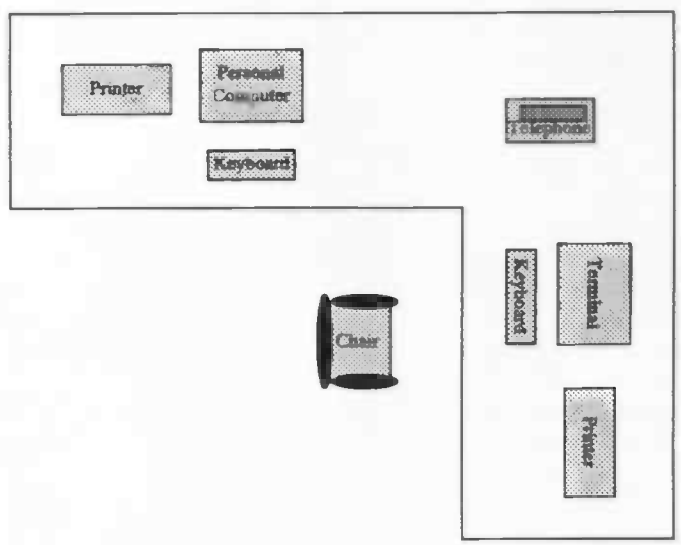


Figure 3-5: Operating desk of a Local SOA Engineer

In one room are about 15 to 20 operating desks. In Figure 3-5 a picture is presented of one operating desk. A Local SOA Engineer has a terminal, a Personal Computer (PC), two printers and a telephone on his desk. On the terminal is the main menu of the SOA system with items like file management, fault handling, access security, a word processing tool and fault reporting. On this screen the Local SOA Engineer receives the fault reporting with SMS, via a MAC analyst or via a Service Box. With the PC the Local SOA Engineer can make connections with PABXs and start the corresponding management tool for that PABX. This PABX-specific management software is delivered by PABX-suppliers for the management of their PABXs.

Now the SOA-system and operating environment have been described. In the next subsection the flow of fault reporting and handling is described.

3.4.3 SOA report flow

In this section the flow of fault reporting and handling is described. The fault reporting and handling are currently the only possible functionalities of SOA. First the general description of the flow from customer to Field Engineer is described as shown in Figure 3-6, followed by a more extensive description of how a Local SOA Engineer handles the report.

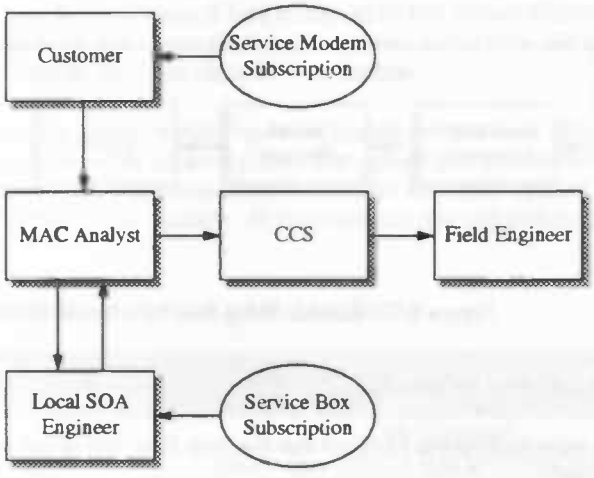


Figure 3-6: Flow of fault reporting and handling

When a customer with a Service Modem subscription detects a fault in his PABX, he contacts the MAC and is served by an MAC-analyst. The MAC-analyst identifies and analyses the fault by asking several questions to the customer and decides if the fault can be handled with SOA. If that is the case, he reports the fault with SMS to the SOA-centre. Faults of customers with a service-box, are already automatically directly passed to the SOA-centre and registered automatically in SMS. If a customer calls in such a case, the MAC analyst can see in SMS that the Service Box already reported the fault. The Local SOA Engineer tries to solve the problem and reports his result in SMS to the MAC. In case the fault could not be solved by remote it will be noticed by an MAC-analyst in SMS on base of the state of the fault report. The MAC-analyst analyses it further and passes it to the Control Centre Services. The Control Centre Services is a central department which controls and manages the assignments of the mechanics. The Control Centre Services sends a Field Engineer, a mechanic, to the customer.

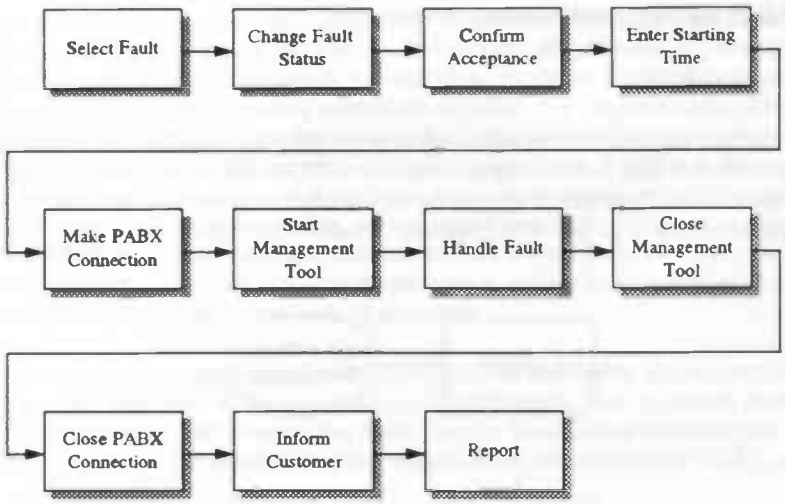


Figure 3-7: Fault handling flow by a Local SOA Engineer

Figure 3-7 presents how a Local SOA Engineer handles a report. The reports can have different states here:

1. The fault is reported by the MAC or the Service Box, but is not yet been handled by an Local SOA Engineer.
2. The Local SOA Engineer is handling the report and tries to solve the fault.
3. The Local SOA Engineer could not solve the fault by remote.
4. The fault has been solved by the Local SOA Engineer or by the PABX.

The Local SOA Engineer receives the reports in SMS, which have initially state 1. All reports are supplied with reporting time, time within the report should be accepted by an Local SOA Engineer, time within the report should be handled by an Local SOA Engineer and the time within a Field Engineer eventual should be at the customer's office. On basis of these data an Local SOA Engineer selects a report and changes the state of the report to 2. After confirming the acceptance and entering the starting time, he can make a connection with the PABX on the other terminal. After making connection he starts the management tool corresponding to that brand and type of PABX. With the management tool the Local SOA Engineer tries to solve the fault on basis of the alarm log which includes all the alarms which occurred after the last fault handling. If the fault is handled or could not be handled the management tool and connection with PABX are closed. The Local SOA Engineer always informs the customer whether the fault is solved or not, and reports the result in SMS. If the fault is solved the state will be changed to 4, else the state will be changed to 3 and a MAC analyst will pass the report to the Control Centre Services as mentioned above.

The whole process of fault handling and describing is described. Now the light is turned on the demonstrator "Virtual SOA" . In Section 3.5 is described who will be the user of "Virtual SOA".

3.5 "Virtual SOA"

In this research we develop a VE demonstrator with the same or a part of the functionality of the SOA-system that offers new possibilities for the users. We do not have the intention to duplicate the current SOA-system in a Virtual Environment.

We hope to prove that a VE-application will offer some advantages to the current SOA system. The user of the VE-application would no longer need the necessary technical information about PABXs and their alarms. He only has to interact with a simple graphical 3D interface, and all the technical details are hidden behind that interface. So the interaction between the user and "Virtual SOA" would be easier than between the user and the current SOA system. Therefore the users of "Virtual SOA" could be less educated than the Local SOA Engineer.

In the next subsections we first make a choice of the SOA-user we take as user of "Virtual SOA". Secondly we make a choice which management category we want to provide with "Virtual SOA".

3.5.1 User of "Virtual SOA"

Currently the faults are reported to the MAC-analyst by telephone and he reports them to the SOA Centre. It would be much easier when the MAC-analyst could solve the fault by himself. The fault will be solved quicker and the Local SOA Engineer will only be needed for hard problems. The delay on simple questions will be decreased considerable.

Because of this latter argument, and because a MAC-analyst is lesser educated than an Local SOA Engineer we have chosen the MAC-analyst as user of our "Virtual SOA" system. A disadvantage would be that a MAC-analyst has to log on to the Virtual Environment system which will cost the MAC-analyst more time than at present, but this extra time will be compensated by the fact that the analyst can handle more by himself than before.

For Local SOA Engineers the added value would be mainly on the area of a clearer and more understandable user-interface, what could improve the speed and accuracy of his work. They have a lot of knowledge about PABXs and have experience with the SOA-system, but these advantages will disappear with the "Virtual SOA" system. At the moment it is still indistinct if VE offers other possibilities to engineers.

In Table 3-1 a profile description is given of a MAC-analyst.

MAC-analyst profile element	Description
Age	19-50 years
Education	secondary school and technical education
Computer skills	M&M007, 4-TEL, KANVAS, MFOS-II, TMOS
Task skills	minimum SOA skills

Table 3-1: Profile of a MAC-analyst

The MAC-analyst is between 19 and 50 years old. He has at least a secondary school education and probably a technical education, because their former job is mostly mechanic. At the moment the MAC-analyst uses computer applications like M&M007, 4-TEL, KANVAS, MFOS-II and TMOS. M&M007 is a fault registration and handling system for telecommunications equipment. 4-TEL is a system that charts the quality of the lines of subscribers of the public net. The other three systems, KANVAS, MFOS-II and TMOS, are management systems for the management of specific switches like 5ESS and AXE. With other applications and systems they will probably have novice experience. MAC-analysts have a minimum of SOA task experience, because the most performance data are gathered by SOA-engineers currently.
The MAC-analyst will have more possibilities with "Virtual SOA" than with the current system. The customer will be more satisfied about the good and fast service, and the Local SOA Engineer will have more time to spend on hard problems, and so it may happen that in less cases a Field Engineer has go to customers.

Now we have chosen a user for "Virtual SOA". In the next subsection the Network Management category is chosen for the implementation of the demonstrator.

3.5.2 Management category of "Virtual SOA"

Currently the management categories provided with SOA are Configuration Management and Fault Management. In the future also Performance Management will be provided. Because within the Devine project already is chosen for Configuration Management
and Fault Management we have chosen for the Performance Management, because this part is not investigated yet. In the next subsection the Performance Management will be explained more extensive and performance elements are selected. In the Subsection Performance Elements and needed data is described which data is needed for the selected performance elements.

3.5.2.1 Performance Management

In this section the Performance Management of a PABX is described, as it could be provided by SOA. The performance of a PABX is divided into the groups *single station*, *group of stations* and *switch*. A single station is a telephone which for example stands on your desk. A group of stations is when a group of stations has the same telephone number, e.g. an information number like the OV-reisinformatie[2] number 06-9292 or the Informatie Beheer Groep[3] number. There are several lines and stations and several operators who can answer your call. If all the operators are busy you will be placed in a waiting queue. Your call will be in the waiting queue until it is your turn and an operator is free. The last group is the switch. All the telephone traffic, internal or external, is arranged by the switch. It is very important that the switch, the stations and the group of stations have a good performance. An internal call is made from a station to a station, and an external call is made from a station to the public network or reversed.

These three performance groups have several performance elements. A choice of which performance elements are used in this research has been made. This choice is made by literature study and by the information gathered by the SOA Centre. Resources used are the "PABX Almanak" of M.R. Oberman [Oberman] and the "Handboek netwerkbeheer" of E. Spaans [Spaans].

Before describing these performance elements the words incoming, outgoing, internal and external calls are described and presented in a table.

call	station	group of stations	switch
incoming	call to a station from another station or from the public network	call to the group number from another station or from the public network	call made from a public number to a station of the switch
outgoing	call from a station to another station or to the public network	call from a group station to another station or to the public network	call from a station to a public network number
internal	call between two stations	call between a group station and another station	call made between two stations of the switch
external	call between a station and a public network number	call between a group station and a public network number	call from a station to a public network number

Table 3-2: Kind of calls

The kind of calls which can be made are presented in Table 3-2. There are not much differences between the definitions of the kind of calls of a station, a group of stations and a switch. Only for the switch an outgoing call is equal to an external call. Below an overview is presented of the performance elements.

[2] The OV-reisinfromatie is the dutch public transport information.
[3] The Informatie Beheer Groep is the dutch scholarship or exhibition authority.

nr.	performance element	station	group	switch
1	number of incoming calls	X	X	X
2	number of outgoing calls	X	X	X
3	number of internal calls made	X	X	X
4	number of external calls made	X	X	X
5	number of times that the station was busy	X		
6	the distribution of the calls over the day	X	X	X
7	the state of a station	X		
8	number of calls in the waiting queue		X	
9	number of waiting callers who hang up before contact		X	
10	maximum waiting time of a call in the wait queue		X	
11	minimum waiting time of a call in the wait queue		X	
12	average waiting time of all calls in the wait queue		X	
13	number of operators who are switched in that group		X	
14	the time that the system has been down			X

Table 3-3: Performance groups and elements

In Table 3-3 an overview is presented of the performance elements for the three groups station, group of stations and switch. The selection is especially aimed at the reachability and capacity of the PABX.

The numbering is in no particular order and is presented in the first column. The second column presents the performance elements. The third, fourth and fifth columns indicate if a performance element belongs to the groups station, group of stations and switch. Below will follow a more extensive description of the performance elements.

1. The "Number of incoming calls" is the number of incoming calls on a station, group or switch. For the switch these are calls made from the public telephone network and for the group and station these can be from another station (internal) or from the public network (external).
2. "Outgoing calls" for the switch are calls to the public network, for the group of stations and station these can be internal or external again, i.e. from a station to a station or from a station to the public network.
3. "Internal calls" are calls from a station to a station where both stations are part of the network.
4. "External calls" are calls made from a station to the outside world.
5. When someone tries to make a call with a station, but can not get connection because that station has already a connection then that station is busy. The "Number of times busy" element gives an indication of the number of times that a station was busy.
6. With the "distribution of calls over the day" can be checked if the capacity is high enough at the busy times. The number of calls will be checked with the capacity of the switch and lines. It is also interesting to see if the calls are incoming, outgoing, internal or external.
7. The "state of a station" indicates if the station is free, busy or down. With free the station can be called, with busy the station is already making a call and with down there can not be made a call, i.e. because of an phone- or line-break.
8. The "number of waiting calls in the waiting queue" are the number of people in the waiting queue who are waiting for a connection with an operator.
9. The "number of waiting callers who hang up before contact" are the number of people who where waiting for a connection with an operator, but did not have enough time or patience to wait longer and hang up.
10. The "maximum waiting time" is the longest waiting time of a call in the waiting queue which has been occurred in a period of time.
11. The "minimum waiting time" is the shortest waiting time of a call in the waiting queue which has been occurred in a period of time.
12. The "average waiting time" is the average of all waiting times of calls in the waiting queue in a period of time.
13. The "number of group participants" are the number of operators in the group of stations.
14. "Total time that the system was down" is the time that the system has been down and no calls could be made or received.

To visualise the performance elements of a PABX data is needed. This data is gathered by the hardware and software of the PABX. In the hardware several counters are integrated for keeping the number of incoming and outgoing calls. The software keeps the so called CDRs, that were

already mentioned earlier in Section 3.3. In the next subsection the needed data for these elements are presented.

3.5.2.2 Performance Elements and needed data

The elements of the performance need data before they can be calculated or derived. These data can in general be derived from the CDRs, the alarm log and the hardware counters. An alarm log contains data about all the alarms which have been occurred in the PABX. In Table 3-4 an overview of performance elements and their data is given.

Performance element	Station	Group of stations	Switch
Number of incoming calls	number of CDRs with B equal to that stations number	number of CDRs with B equal to that groups number	number of CDRs with A an public network number and B a station
Number of outgoing calls	number of CDRs with A equal to that stations number	number of CDRs with A equal to a group member	number of CDRs with A station and B a public network number
Number of internal calls	number of CDRs with A and B both stations numbers	number of CDRs with A group member and B a station	number of CDRs with A and B both stations
Number of external calls	number of CDRs with A or B an public network number	number of CDRs with A or B is a public network number and the other is a group member	number of CDRs with A station and B a public network number
Number of times busy	hardware counters		
Distribution of calls through the day	all data out of the CDR	all CDRs sorted by starting time	all CDRs sorted by starting time
Station state	information out of CDR or alarm log		
Number of calls in the waiting queue		deriving the number of calls with software from the waiting queue	
Number of waiting callers who hang up before contact		deriving the number with software from the waiting queue	
Maximum waiting time		deriving the time with software from the waiting queue	
Minimum waiting time		deriving the time with software from the waiting queue	
Average waiting time		deriving the time with software from the waiting queue	
Number of group members		deriving the members by software from the group-definition	
Total time that the system was down			derive with software from the alarm log

Table 3-4: Needed data for a single station, group of stations and switch.

The number of incoming, outgoing, internal and external calls, can be calculated on the basis of the A- and B-Numbers in the CDRs. The number of times busy, can not be calculated from the CDRs, because a CDR is only kept when there really has been a connection. This information has to be derived from the hardware counters. The distribution over the day uses all the data in the CDRs on that specific day. The state of a station can be derived from the alarm log, if the station is down, or from the CDR, if the station is busy or free.

The first five elements are already described after the previous table. Only the A- and B-Numbers can differ. With a group of stations there is also a waiting queue where callers are queued when there is no operator free. With software the number of waiters in the queue and their waiting times can be registered and if they hang up before getting contact. The number of group members can be derived from the software part where the group is defined with.

The first five elements are already described. The difference here is in the A- and B-Numbers and also that the number of outgoing calls are equal to the number of external calls. The total time that the system was down can be derived on basis of the alarm log. Now all the performance elements and the data needed for their presentation are described. In the design and implementation phases is this information used for the design and implementation of "Virtual SOA".

3.6 Conclusions

In this chapter we gathered information for the demonstrator "Virtual SOA".

First we described Network Management in general and what a PABX is. With this information we could describe the SOA-system. The SOA-system currently provides Fault Management and Configuration Management. In the future also other management categories like Performance Management will be provided.

After describing the SOA-system we decided to choose the MAC-analyst as the user of the demonstrator. The MAC-analyst is lesser educated and has lesser technical knowledge than an Local SOA Engineer. That is why the easy to use 3D interface of "Virtual SOA" can provide more advantage to an MAC-analyst than to an Local SOA Engineer. Also the time delay for customers would decrease, because MAC-analyst does not always have to pass the report to the SOA Centre.

We also made the choice to provide "Virtual SOA" with Performance Management.
This choice is made because within the Devine project already is chosen for Configuration Management and Fault Management and Performance Management is not investigated yet. Further we selected some Performance Elements we want to visualise with "Virtual SOA". This selection is made by literature study and by the information we gathered at the SOA Centre and is especially aimed at the reachability and capacity of the PABX.

4 Guidelines for VE design

4.1 Introduction

In this chapter some guidelines for the design of VEs are presented. This chapter starts with the description of the AIP-Cube of David Zeltzer. He has found a very interesting qualitative tool for describing, categorising, comparing and contrasting VEs. In Section 4.3 the *seven presence elements* of Peter Hofstede are reported. They are the results of a practical evaluation of a range of Shared Virtual Environments from the world of games, tele-collaboration, entertainment and the Internet. In the Sections 4.4 and 4.5 these seven presence elements are checked on completeness and validity. If possible they are supplemented with guidelines out of literature. The result can be used as a starting point in the design of VEs.

4.2 AIP-Cube of Zeltzer

In his article "Autonomy, Interaction and Presence" David Zeltzer presented a taxonomy of graphic simulation systems, based on three salient components: *autonomy*, *interaction* and *presence* [Zeltzer]. The resulting *AIP-Cube* provides a useful qualitative tool for describing, categorising, comparing and contrasting VEs.

To run a VE-application you need a computing platform, graphics engine and associated peripherals. The computing platform can for example be a Personal Computer, a MacIntosh or a UNIX (graphic) workstation. There are developed special graphic workstations, for example by Silicon Graphics (RealityEngine2), which run on parallel architectures. The graphic rendering operations occur on parallel paths. Maintaining an adequate graphic frame rate is so computationally demanding that special-purpose hardware is often necessary. Besides the graphic workstations there are also graphics engines which can provide rapid graphic actions. In combination with associated peripherals like HMD, DataGlove and SpaceMouse the platform and graphics engines are the hardware components of a VE [Durlach].

In addition to the computing platform, graphics engine and associated peripherals, three key components of a VE - or of any graphic simulation or computer animation system - are autonomy, interaction and presence. Let us examine each of these three components in turn.

- ***Autonomy***

Autonomy is the ability of objects to act and react to simulated events and stimuli. At one extreme, graphical VE objects may be passive and with no associated procedures. At the other extreme are virtual actors capable of reactive planning, and, ultimately, more powerful knowledge-based behaviours. Between these extremes, we can augment objects and agents in various ways, for example, with procedures that account for the mechanical properties of rigid and non-rigid objects, yielding what has come to be known as physically based models. *Autonomy* then, is a qualitative measure of the ability of a computational model to act and react to simulated events and stimuli, ranging form 0 for the passive geometric model to 1 for the most sophisticated, physically based virtual agent. An example here could be a bartender in a virtual bar, who pours you out a drink after you ordered one.

- ***Interaction***

In this context, interaction means the ability of access to model parameters at runtime (i.e., the ability to define and modify states of a model with immediate response). The range is from 0 for "batch" processing in which no interaction at runtime is possible , to 1 for comprehensive, real-time access to all model parameters. Most current graphics systems are indeed highly interactive, such that real-time manipulation of rigid objects is often controlled by joysticks, knobs or a mouse. In other application domains, such as computational fluid dynamics, interaction remains quite limited

due to the computational cost of updating the model at each time step. An example here could be a racing game, where the player can change the speed of the car and the car directly slows down or speeds up on the screen.

- ***Presence***

We are immersed in a very high bandwidth stream of sensory input, organised by our perceiving systems, and out of this "bath" of sensation emerges our sense of being in and of the world. This feeling is also engendered by our ability to affect the world through touch, gesture, voice, etc. The presence axis provides a rough, lumped measure of the number and fidelity of available sensory input and output channels. An example here is a motorbike racing game in an Arcade. The player sits on a bike which can slant, and the player feels he is actually participating in a real race.

These three components are taken together in the AIP-Cube (see Figure 4-1).

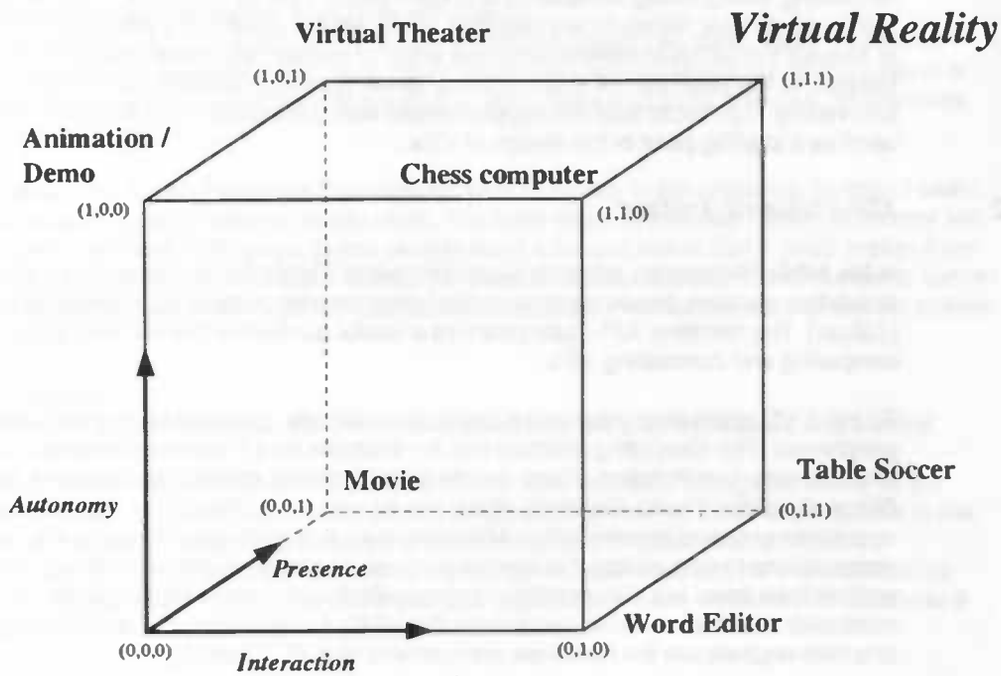


Figure 4-1: AIP-Cube of Zeltzer

The autonomy, interaction and presence degree of a VE can be measured, i.e. by comparing several VEs, and the result is a point within the cube. The measurement of these components is subjective, because there are no plain rules for measurement yet and people give their own, subjective, opinion. It will be hard to compare two VEs on basis of the categorisation by different people. It can give a good impression, but it less precise.

The origin represents the early graphics systems that were programmed in non-real-time batch mode. These systems were not autonomous, nothing spontaneous happened, were not interactive, after starting nothing could be changed, and they gave no feeling of presence. A chess computer can be seen as an autonomous and interactive system. The chess computer reacts to the moves of the human player by making an unpredictable move. Such a system is placed in point (1,1,0). When a system is completely autonomous and not interactive or providing a feeling of presence, it is placed at point (1,0,0). An example can be an animation or demonstration program which, after start-up, need no input of the user until the programs ends. Point (0,1,0) represents a complete interactive system with no presence feeling and no autonomous processes. The user can control the whole system during execution. An example here is a simple word editor. When you are paying a game like table soccer, you can get so involved in the game that you forget the environment around you. Such a play can be placed at (0,1,1) because the player feels presence and interacts

by kicking the ball with the puppets on the stick. A system is placed at point (0,0,1) when it is non-interactive and passive and extremely realistic and multi sensor. An example here can be an very good movie. The viewer has the feeling of driving the car in the pursuit / chase but can not for example make the car ride faster. It is also not autonomous, because every time you see the movie nothing new happens. When a theatre-play is performed "live" on TV several times, there will be differences between those performance and you could say that such a "Virtual Theatre" is autonomous and give presence feeling. Such a play would be presented by point (1,0,1). The ultimate point is (1,1,1). If a system is fully autonomous, interactive and gives presence feeling it is a real VR-system. It is the holy grail for VE-designers.

The AIP-Cube gives a good indication about the quality of a VE-application. It does not say how to design one. In his research at KPN Research Peter Hofstede derived some guidelines for the design of Shared VEs which can serve as starting points for the design of VEs. A Shared VE is a VE where multiple users at the same time have access on that VE. The users can be present in that VE and can interact and communicate with each other at the same time. Examples of Shared VEs are network games like Doom, Duke Nukem and Descent, simulator platforms like SIMNET and Startours and Multi User Dungeons (MUDs). MUDs are multi-user role playing games and adventure games on the Internet and textual and/or graphical based. The next section presents the Seven Presence Elements and their corresponding guidelines.

4.3 Seven presence elements of Hofstede

Hofstede has derived a general set of rules from the Shared VE examples and the theoretical background mentioned in his report. These elements have been chosen because they can guide the designer to a (shared) virtual environment based on the concept of presence, especially telepresence. Hofstede has chosen these elements they can guide the designer to a (Shared) VE based on the concept of Presence or Telepresence. The first subsection covers the description of the seven elements and the second covers the guidelines corresponding to the seven elements.

4.3.1 Description

In this section the seven presence elements of Hofstede are discussed. They are presented in no particular order. These elements are:

- A spatial representation of the environment
- Freedom of movement
- Real world control
- Communication and interaction with (lots of) users
- Responsiveness
- Autonomous processes
- Sufficient input and output

A spatial representation of the environment
A spatial representation means that users can build an image in their mind of the VE. Users should be allowed to create a spatial mental map of the environment by visual, textual, auditory and other media used for description of the VE. If not directly visible, users must be able to visualise it mentally.

The topological structure of the VE is very important if presence must take place. The style and metaphor of the world, the entry points and exits, the location of important objects and its relation to other scenes largely determines whether a user will perceive the VE as a place or as a collection of things. This element can not stand alone, because it always comes coupled with freedom of motion and a degree of responsiveness (change of viewpoints as reaction to user input).

Freedom of movement
With freedom of movement is meant that users can move freely through the VE in all directions and places. Users must be able to navigate through the VE. By moving around, users must be able to access applications and find other people.

Without freedom of movement, worlds become pictures. Freedom of movement is essential in small spaces when doors, buttons and other objects must be reached. In a larger context, freedom of movement must allow people to explore the environment. Furthermore, if something invites you

to come nearer, it should be possible to actually go there. It will be disappointing if people were invited by something but do not find anything interesting, like an empty room behind a door.

Real world control
With real world control is meant that users can interact with objects which also appear in real life and users know how to interact with these objects. While the environment itself may deviate from real life, direct interaction with objects similar to normal use, (instead of indirect control via commando's) will help to create presence.

A scene that meets the expectations of users induces a sense of presence. The direct common knowledge metaphor is probably the best choice for inexperienced users. Real world control should be available on a functional level: if there's a door and a door handle, click on it to open. Having to pull the door handle downwards might make the world realistic, but not necessary more understandable. This would especially be the case when the procedure must be performed with difficult input devices.

Communication and interaction with (lots of) users
With communication and interaction with users is meant that users can "talk", "fight" or interact /communicate through modalities with each other, which also appear in real life. In the case of a Shared VE, communication and interaction with the other users should be possible, where users are used to in ordinary life.

Hofstede tested the Seven Presence Elements on their influence to the presence feeling of users, with his demonstrator of an Internet music store. The tests seem to indicate that an encounter with another human can work both ways. Some people like it a lot and report that it really makes them feel more present but others said that it breaks the illusion. Most of the aspects of human to human interaction have not been tested yet and it should be researched further before any definite answer can be provided.

Responsiveness
With responsiveness is meant that the environment reacts to the user's presence and actions.

With nothing to do in the VE, it quickly becomes a bore. This will have a negative effect on the presence. To attract the user's attention it should be possible to start all kinds of processes in the environment like performance data requests and connecting to PABXs. Another form of responsiveness is the spontaneous reaction of the VE to the user. This can be implemented by objects that change colour and shape when the user approaches them. In the test of Hofstede it seems that its effect might be a bit overrated.

Autonomous processes
With autonomous processes is meant that things happen spontaneously and not as a result of a explicit request from a user. Autonomous processes increase the liveliness of the VE.

Autonomous processes can be combined with responsiveness. For example, an animation can start playing only after the user requested it. An autonomous character can start talking to the user when he comes within a certain range. Other autonomous processes can be used to introduce atmosphere into an environment, e.g. through background music, clouds flying overhead, vehicles moving in the distance and sounds of nearby machinery. The autonomous processes in the test of Hofstede did not make a big impression on the participants of the test. Nevertheless, it is still one of the subtle touches that can discriminate the VE from others.

Sufficient input and output
With sufficient input and output is meant that the VE can get enough input of and give enough output to the users, and the hardware has sufficient capacity. Instead of striving to maximise technological fidelity, the amount and type of cues must be specifically tailored to the needs of the functions of the VE.

When the hardware is so slow that it obstructs the user from feeling present in the VE, a graphics accelerator can be considered. Extra input and output devices might help to improve immersion. However, other factors such as the operating context of the system (e.g. in a laboratory or a real working environment) influence the choice of hardware.

4.3.2 Guidelines

In this section the guidelines corresponding to the seven presence elements of Hofstede are presented. As already said, these guidelines are a good starting point for the development of a VE. They will be applied in the design of "Virtual SOA".

A spatial representation of the environment

- Design the environment in such a way that the user can construct a mental map of it.
- Use magical "transporters" only if they add extra functionality to the scene. This can be the case if large distances must be crossed to get somewhere.
- Design the scenes in such a way that the user can discriminate between them. When a user is in a room, the wall's should not be exactly the same on each side. Doors, buttons and links can be made the same in each scene to help identify them.
- Create depth in a scene by applying image textures (if the software allows it) or by object textures (grids of lines or small dots, for instance). Objects that shift in front of each other (motion parallax) create an illusion depth.

Freedom of movement

- When possible, use collision detection. Collision detection means that objects can not collide with each other, i.e. that a ball can not go through a wall. When this function is not available, design the scenes in such a way that users do not have to manoeuvre through tight spaces.
- Keeping up speed (frame rate) in the scene is critical.
- Allow people to explore the environment in all necessary dimensions.
- Adjust the travelling speed to the environment: slow when indoor and fast outside.
- Do not make buildings and rooms where nothing is inside, because people will want to enter everything they see and they will be disappointed when there's nothing to be seen.

Real world control

- Real world control on a high level can be achieved by mimicking everyday interaction. Examples are: opening doors, putting stuff in bags, driving a car. These clichés can be used in VE based on standard metaphors. If the scene is abstract, clichés are not available and real world control must be based on certain conventions about the way things work. Such conventions can be based on physics (when you squeeze something it becomes smaller) and on learning (turning a dial to the right results in a linear increase). The science of control ergonomics can provide some insight into these concepts.
- Indirect interaction via pop up menus might be replaced by more natural and direct equivalents in a VE.

Communication and interaction with (lots of) users

- Provide the user with the means to express his feelings and emotions. If the interaction takes place via avatars, build a "gesture" control system (verbs), or make use of real-time audio and/or video communication.
- Enable users to change their representation in the VE. A single user mode or "do not disturb" option should be considered since not everyone is interested in a chat *all* the time.
- Determine the amount of control users have over the environment and over each others world.

Responsiveness

- Provide feedback for every choice the user makes. Feedback can be auditory (clicking of a button) and visual (change of colour of the button). Advance systems can even provide other feedback in the form of touch and force.
- The environment can respond in various ways to the user:
  - by greeting him when he approaches
  - by changing the sounds of footsteps when the surface changes from stones to grass
  - by making a sound or changing the colour of the wall when a user bumps into something
  - by turning on lights when the user approaches
  - by automatically opening doors when the user approaches
- As a general rule, responding objects and interaction make the VE appealing, so make sure that the user can interact with (all) the objects in the scene.

Autonomous processes

- Make some or all objects autonomous in their actions, to create a spontaneous, liveliness VE. In real live a lot of things also happens spontaneous and unpredictable.

Sufficient input and output
Important capabilities of the hardware and software are:

- Number of polygons per second. Ranges from below 100.000 on PCs more than 1.000.000 on high end workstations and special purpose machines.
- Hardware texture mapping. Texture mapping is the process of displaying a single digitised image on a polygon or structure that's made out of polygons. This display technique saves computing power because the image is stored in memory and displayed, as opposed to being recalculated and redrawn. Realistic scenes require texture mapping. To ensure a high frame rate this process must run on separate processors and memory.
- Scripting facilities. A scripting facility is the possibility to provide an object with a script, i.e. a program code with functions and procedures. How difficult is it to make a ball turn blue when you touch it and is it possible to define a parabola as the flight path after you throw it away ? Is it possible to create an autonomous character that starts speaking when you approach it? And very important: can such functions be accessed via a graphical interface, a scripting language or a programming language ?
- Extensibility. Is it possible to extend the software to the "outside world". For example, can it be linked to an external database in real-time? Will drivers for new input devices be available ?
- Development environment. Is it necessary to switch between the graphical modeller, interaction modeller and real-time engine or can all functions be executed in real-time ?

The guidelines presented in this section can be used as starting point for a VE-design. In the next sections the completeness and validity of these elements are discussed. First technical are given, followed by non-functional requirements.

4.4 Technical requirements

We find that the guidelines of the seven presence elements can be supplemented on the issue of user navigation. The following supplementary guidelines are technical guidelines and indicate how you can achieve better user navigation in VEs:

- *Minimising the latency between user movement and change in view is crucial* [Baker]. It even has a maximum of 0.1 seconds. In designing travel mechanisms, the stability of the perceptual-motor loop must be maintained. Disruption or distortion of the natural coupling between body movement and view leads to disorientation and confusion for the user.
- *Maintaining a frame update rate of at least 6 frames per second is necessary for users to stay immersed in the virtual environment* - i.e., to travel using an egocentric view [Airey]. When frame rates fall below this rate , users retreat to using an exocentric view. Currently the frame update rate should be at least 10 or 12 [Esposito].
- *When trade-offs must be made, travel will be more disrupted by degraded update rates than by degraded imagery* [Baker]. If the frame rate gets to low, just make the imagery lesser detailed to increase the frame rates.
- *Make the system responsive* [Baker]. A naturalistic environment depends on a responsive system. A couple of useful strategies for improving responsiveness are:
    - Predictive filtering of the user input stream can have some success in reducing the lag between user movement and display update.
    - Use simple imagery in order that substantial negative impacts are never encountered. Simplified imagery diminishes the sense of visual reality, but from the perspective of user engagement, user "presence" within the environment, and overall system performance, it is well worth the price.
    - Make use of multiple resolutions. The geometry in the centre of the view is rendered with full detail, while parts of the world that are peripheral are rendered more coarsely.
    - Make use of techniques like *adaptive* or *progressive imagery*, i.e. draw simplified views during navigation and progress to more detailed views when the user has come to rest. Care should be taken here to provide sufficient texture information to achieve a realistic sense of forward motion.
- *Allow the direction of motion to be decoupled from the user's direction of gaze* [Baker]. This makes a certain amount of "ecological" sense, since we can easily look to the side while we move forward (Travel-gaze decoupling). This is feasible in those VEs where "real-world" pieces of equipment, such as stationary bicycles and tractors, provide steering controls. The steering

mechanism can control direction of travel, while the head-tracked user gaze can be used to control camera direction.

To complete the spatial representation of the environment we mention some techniques here that contribute to the users construction of a mental map:

- *3D sound cues and speech-synthesis technologies can be used to enhance the user's overall situational awareness of the VE* [Fisher]. 3D sound cues give distance and direction information for proximate objects and events.
- *The utility of making available two views of the data set has been pointed to by a number of researchers* [Baker]. One view is an *egocentric view* of the scene reflecting the user's current viewpoint. The second view is *exocentric* and depicts the overall virtual world, with the user's current location indicated by a **You Are Here marker**. The overall view could be presented as a small inset in the screen, similar to the topographic maps produced by the US Geological Survey, where the location of the quadrant is outlined in a small state map in the upper left corner. Alternatively, a second screen can be used.
- *Users should have control over the orientation of the exocentric map* [Baker]. The orientation of the exocentric map can be significant. The map could be presented with a canonical frame of reference (North up for geographical maps), or in a track up frame, where the orientation corresponds to the user's momentary point of view. A "North up" orientation requires that users perform a mental rotation to align themselves and the map. Since the cost of such mental rotation is well known in the psychology and human factors communities, users should probably have control over the orientation of the map. The importance of this flexibility is borne out by research in the aviation community and in virtual environments.
- *Perception of size and distance diminish with a decreasing field of view* [Baker]. Building a useful cognitive map of a physical space presented as a virtual environment may be hindered by some types of display equipment. Head-mounted displays, while good at immersing the user in the virtual environment and increasing the user's feeling of "presence" in the virtual environment, provide a limited field of view. Henry found that individuals wearing HMDs underestimated the dimensions of a set of rooms in a virtual museum. Individuals who walked through the actual museum made accurate judgements; those who viewed the synthetic museum on a regular computer monitor underestimated, but not as much as those wearing helmets. These results are consistent with earlier work that shows that perception of size and distance diminish with a decreasing field of view.
- *Users employ their cognitive map when searching for target objects* [Baker].
- *Automated travel offers advantages of speed and flexibility* [Baker]. However, automated travel can result in a loss of situation awareness. If I am instantly transported to a new position, I have no way of knowing about the surrounding terrain. On the other hand, I would have first-hand knowledge of the environment if I had found my own path and travelled through it. There are variants of the automated travel metaphor that ease that task of travel but which are less disorienting, since the user does indeed pass through the environment. For example, the user can specify a target destination and the system "moves" the user through the environment to that location. it may be that the user controls the rate of travel, or the system might control the velocity perhaps using a logarithmically diminishing rate as the user nears the target.

4.5 Non-functional requirements

In the previous section the seven presence elements of Hofstede are described and the corresponding guidelines are presented. In this section, we discuss the completeness and validity of these elements. We have divided the discussion in two parts. The first part discusses the seven elements in general and the second part discusses every element separately.

We derived information from literature and articles about VEs and human factors on VEs. Especially from the human factors literature we got information about the health and safety issues and social impact of VEs. We already described these subjects in Chapter 2, Section 2.4.3 and Section 2.4.4. VEs and VR can have bad influence on the *health and safety* of users. Users can get harmful reactions like physical pathologies and Cybersickness. The *social impact* on users can contribute to a complete isolation of the outside world. The seven presence elements do not cover this subject. Because we should not underestimate the health and safety and social impact with the use of VEs we discuss them here.

4.5.1 Health and Safety

When a user is interacting with a VE, it is not in the Intention that this has side effects on the user. Side effects can be that the user gets dizzy or sickening by wearing a HMD too long. In Section 2.4.3 we already discussed the health and safety. Some guidelines to prevent such side effects are [Stanney]:

- Let the user make a minimum number of repeated movements to avoid repetitive stress injury.
- Do not let a user become so involved and engrossed In their virtual world that they become unaware and even possibly disoriented with their immediate surroundings to avoid immersion injury.
- Avoid transmittable diseases by just letting one person use the equipment, clean and disinfect the equipment after each user.
- Do not let a user wear a HMD to long to avoid nausea, eyestrain and the impact of electromagnetic fields upon the eye structures and the Central Nervous System. You could think of a maximum of 10 minutes.
- Warn users about what they can expect in connection with anxieties like acrophobla or claustrophobia to avoid psychological effects.

4.5.2 Social impact

Yet not much is known about social impact. Most VR conferences have yet to even recognise and address that social issues may exist. Currently the potential negative social influences resulting from VE exposure are not well understood. Many open issues need to be proactively explored in order to circumvent negative social consequences.

4.6 Conclusions

First we discussed the AIP Cube of Zeltzer which is a very interesting and useful for the categorising and comparing of VEs. The three components autonomy, interaction and presence describe in a sufficient way the quality of a VE. Comparing and categorising is one thing of VEs, another part is the design of VEs. Peter Hofstede derived the seven presence elements in his research, which are an deepening of the Presence axis of Zeltzer. To make a good design it is recommendable that you use some guidelines, anyway as a starting point, and it will deliver a good basis for a VE. The seven presence elements serve as a good starting point of designing a VE on the basis of presence. The guidelines corresponding to these elements are supplemented in this research on the part of spatial representation and completed with guidelines on the user navigation in the "Technical requirements" section. In the "Non-functional requirements" section guidelines are presented which are for the health and safety of VE users.

5 Design

5.1 Introduction

In this chapter we describe the design of "Virtual SOA". First three basic designs of VEs are made. Based on some criteria a choice is made for one design that is used for "Virtual SOA". In Section 5.3 the user, software and presence requirements are described. The user requirements cover the description of the actions a user has to perform to accomplish a task. There are selected three tasks. Based on the user requirements the software requirements can be derived. The software requirements describe the functionalities "Virtual SOA" has to provide to let the user perform his tasks. The presence requirements are the seven presence elements applied on the chosen design in section 5.2. After the requirements engineering the functional and technical design are made.

5.2 VE-interface design

First we design a user-interface for the presentation of the performance elements mentioned in Chapter 3. We made three designs for a user-interface. These designs have been compared and one is chosen. The first three subsections describe the designs and the last describes the design choice.

5.2.1 "Performance Space"

The first design is called "Performance Space". Several 3D objects are floating in several spaces.

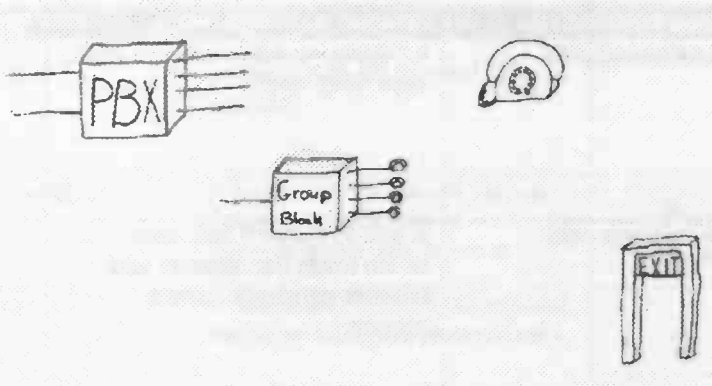


Figure 5-1: "Performance Space" after starting

After starting the application, the user sees the space presented in Figure 5-1. This space consists of four objects. Three of the objects are the performance groups *station*, *group of stations* and *switch*. The fourth object is an exit gate. When the user flies through the exit gate, the application will shut down. The other objects can be selected by clicking on it. After clicking on the "PABX"-object the performance data of the PABX is presented. After clicking on the "station"-object or "group of stations"-object, the user must first select a telephone number corresponding to that station or group. The method of selection is presented in Figure 5-2.

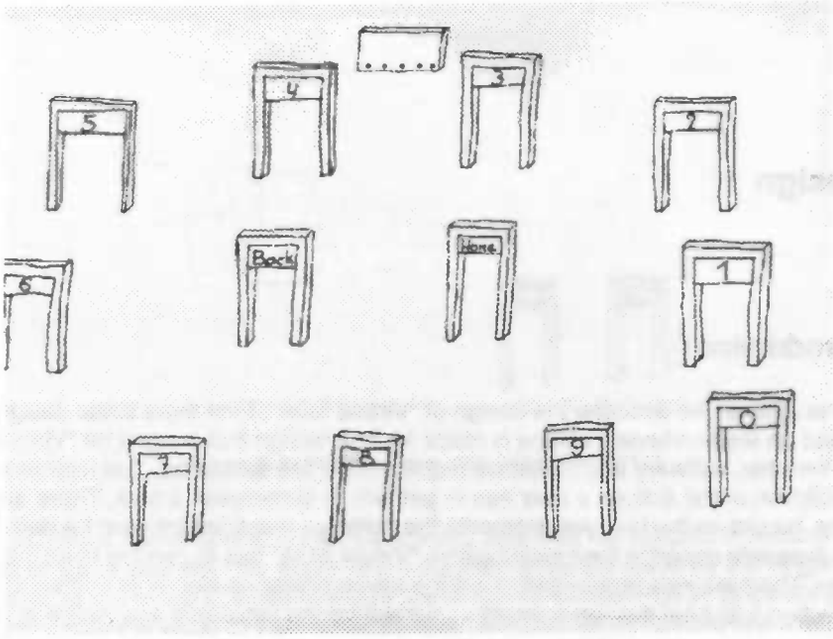
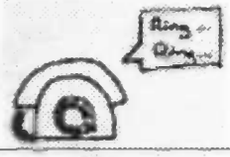
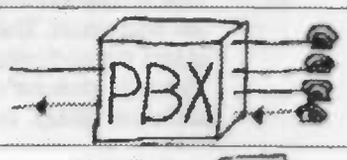
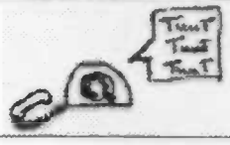


Figure 5-2: Selection of telephone numbers

The telephone numbers consist of five digits. The user sees a space with several gates. The gates marked with "0" till "9" are gates for the selection of a digit. The "Back"-gate takes the user one selection step back, to give the opportunity for rectifying a selection error. With the "Home"-gate the user can travel back to the first space, presented in Figure 5-1. On the sign-object the part of the telephone number is presented that already is selected. After the number selection the performance data is presented.

When the performance data is presented there are floating several performance objects in a space. We designed some 3D-icons for the performance elements, which are presented in the table below.

Performance element	3D-icon description	Image
Number of incoming calls	A telephone with a balloon which says "ring, ring".	
Number of outgoing calls	A telephone with one hand which holds the receiver and another hand who dials a number.	
Number of internal calls	A representation of a PABX with lines and trunks and a connection between two stations connected to lines.	
Number of external calls	A representation of a PABX with lines and trunks and a connection between a station and a trunk.	
Number of times busy	A telephone with a receiver laying next to the phone and a balloon which says "tuut, tuut, tuut".	

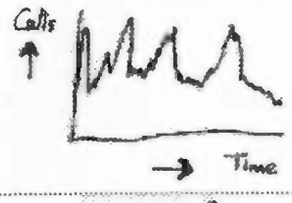
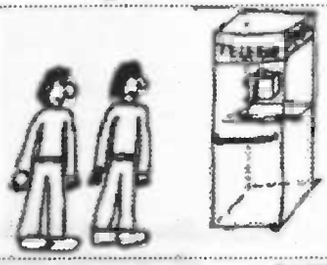
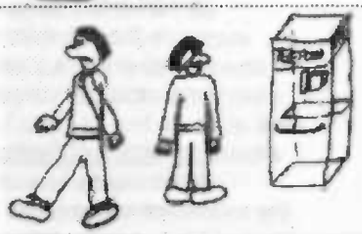
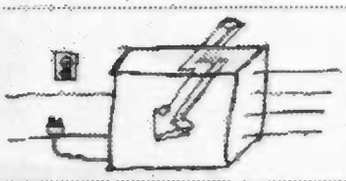
The distribution of calls over the day	A diagram with a number of calls mapped on the time.	
Station state	A telephone with a big "I" of information next to it.	
Number of calls in the waiting queue	A queue of people waiting in front of a public telephone.	
Number of waiting callers who hang up before contact	A queue of people waiting in front of a public telephone, and one person is walking away.	
Maximum waiting time	A clock, with 75 percent of it shaded.	
Minimum waiting time	A clock, with 25 percent of it shaded.	
Average waiting time	A clock, with 50 percent of it shaded.	
Number of group members	A block with the words "Group Block" and a couple of telephones connected to it.	
System down time	A PABX with a red lightning and the plug out of the plug-socket.	

Table 5-1: Table of performance icons

These floating space-elements can be selected by a control device like a mouse. When an element is selected, the space will be filled with 3D diagrams. Figure 5-3 contains a drawing of such a space.

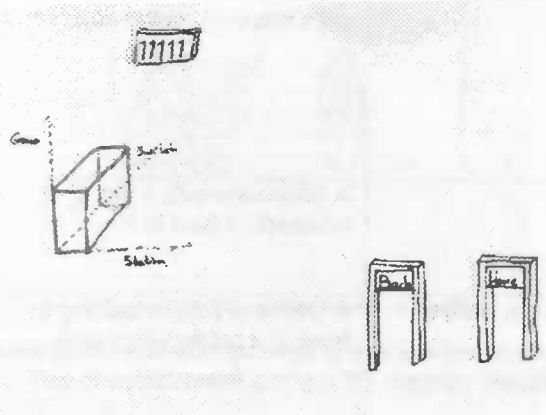


Figure 5-3: Performance data of station 11111

The space contains a "back" and a "home"-gate and a sign with the telephone number of the station or the group of stations. The x-axis of the diagram gives the value for the single station, the y-axis the value for the group of stations where the single station belongs to (if it belongs to a group) and the z-axis the value of the switch. In case the information needed is a time, like waiting time or system down time, the diagram will be replaced by a digital display.

Evaluation of Performance Space:
Advantages: use of 3D
Disadvantages: no direct survey, long search time
Interesting points: -

5.2.2 "Performance Inside"

"Performance Inside" is a world that contains objects like a PABX, blocks, group blocks, stations and lines between them. At the start of "Performance Inside" all these objects are visible, as is presented in Figure 5-4.

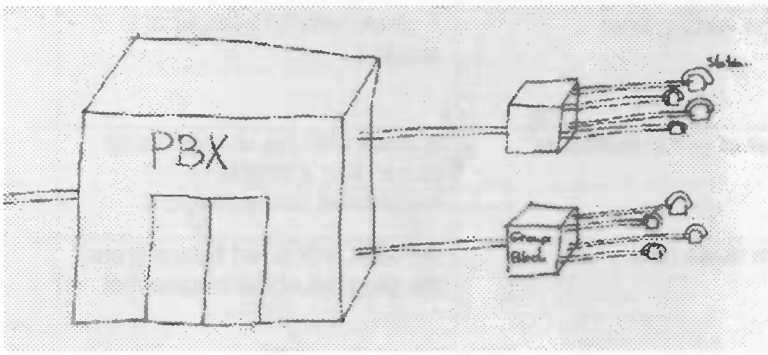


Figure 5-4: Viewpoint at start of Performance Inside

In "Performance Inside" the user can travel inside the objects. The inside world can be reached by the doors in the PABX. After entering the PABX and turning to the right, the user has the view as presented in Figure 5-5.

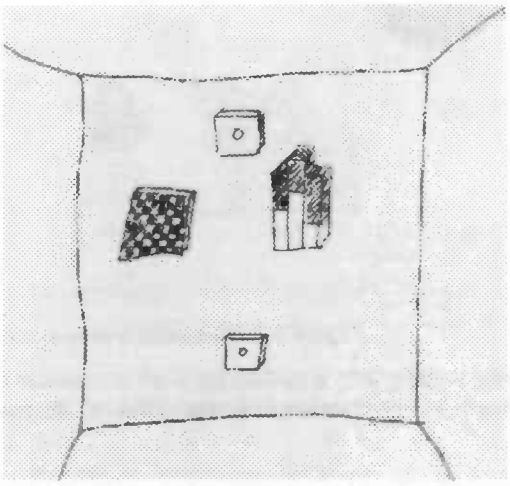


Figure 5-5: Viewpoint inside the PABX

Inside the user sees a checker-board, a 3D bar diagram and two entry-blocks. The checker-board represents the distribution of calls through the day. Every hour is presented as a bar on the checker-board. The number of incoming and outgoing calls and the number of times busy are presented by a block of three 3D bars. The bars are splitted again in internal and external parts. The entry-blocks are the entries of lines, which lead to a block or a group block. A block contains lines to stations or other blocks and a group block represents a group of stations and contains lines to stations belonging to the group. When a PABX has a lot of stations connected to it, the blocks are used to select the telephone numbers. There are presented several entry-blocks for the selection of a digit. After entering the line and arriving in the next block, again ten entry-blocks are presented for the selection of the next digit. This continues until the complete number is selected and the user arrives at a station or a group block.

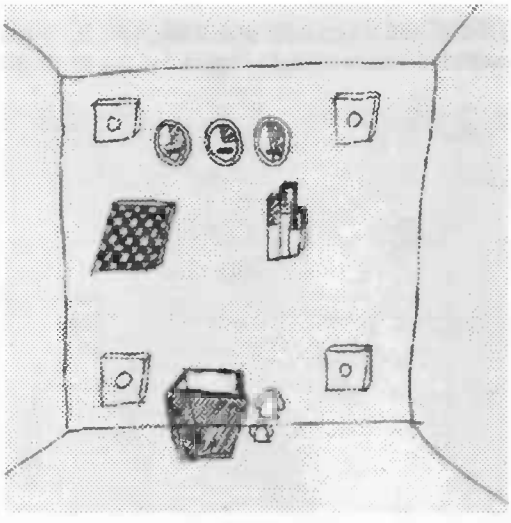


Figure 5-6: Viewpoint inside the group block

In Figure 5-6 the inside of a group-block is presented. A group-block contains entry-blocks, a checker-board, a 3D diagram, three clocks and a wastebasket with telephones floating around it. The checker-board and the 3D diagram present the same kind of data as for the switch, but now data belonging to that group. The three clocks represent the waiting times of calls which are put in the waiting queue. The left clock is for 75 percent shaded and represents the maximum waiting time of the calls. The clock in the centre represents the minimum waiting time and is for 25 percent shaded, and the clock at the right is for 50 percent shaded and represents the average waiting time. The floating stations are calls that are waiting at that moment. If a call is answered the floating station disappears, but if the caller hangs up before getting contact, the floating station is moved to the inside of the wastebasket.

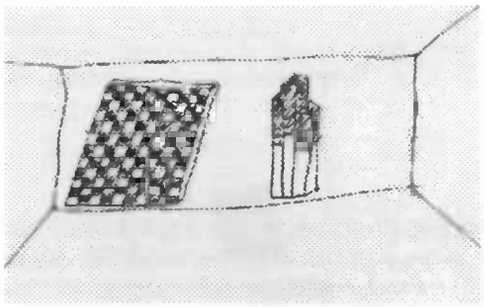


Figure 5-7: Viewpoint inside a station

When the user comes floating into a station he sees a checker-board and a 3D bar diagram. This is presented in Figure 5-7. The checker-board and the 3D diagram visualise the data corresponding to that station.

Users can float through the lines, stations, blocks and PABX in all directions. If the user wants to return from a station to the PABX or wants to undo a telephone number selection, he has to go back through the line he came. The lines have colours to indicate their state, which can be free, busy or down. Free is indicated with green, busy is indicated with yellow, and down is indicated with red.

Evaluation of Performance Inside:
Advantages: use of 3D, close to reality, some information will be seen from all viewpoints
Disadvantages: no direct survey of all performance elements
Interesting points: experience from inside the PABX

5.2.3 "Performance Room"

The "Performance Room" is a room with one desk, two 3D displays which are a kind of hologram and a public phone with waiting queue. In Figure 5-8 the desk environment is drawn.

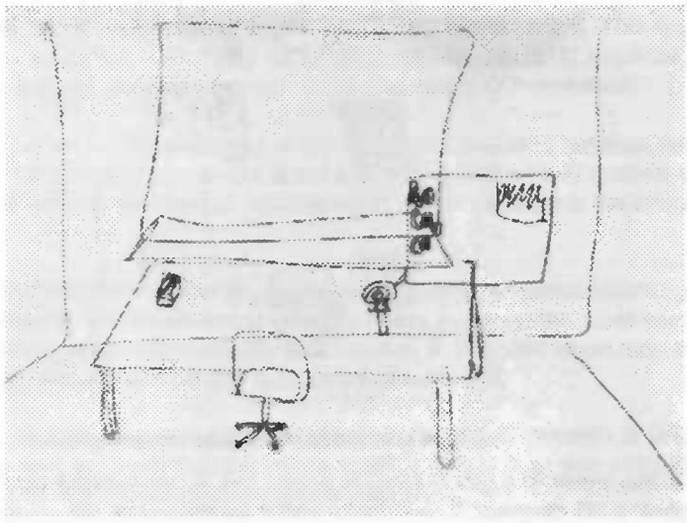


Figure 5-8: Viewpoint to the front side

On the desk there is a card-file and a telephone. With the card-file a telephone number belonging to a station can be selected. The numbers are ordered increasingly. If a station is selected by his number, there appears a balloon with information on the performance elements incoming calls, outgoing calls, internal calls, external calls, number of times busy and the distribution of calls over the day. The telephone on the desk indicates with colours if the station is free (green), busy (yellow) or down (red).

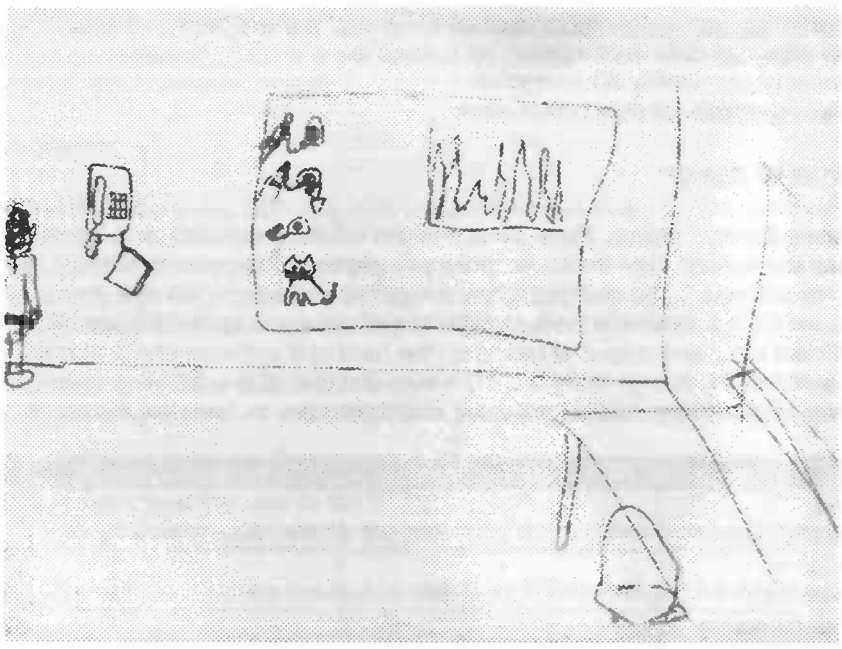


Figure 5-9: Viewpoint to the left side

The left wall of the room presents the performance data of groups of stations. This is drawn in Figure 5-9. The wall contains a public telephone and a display. Below the telephone hangs a telephone-directory. This is used for the selection of the groups by their description and their numbers. When a group is selected the display shows the performance data corresponding to that group. If there are calls in the waiting queue of that group, it is represented by waiting callers in front of the telephone.

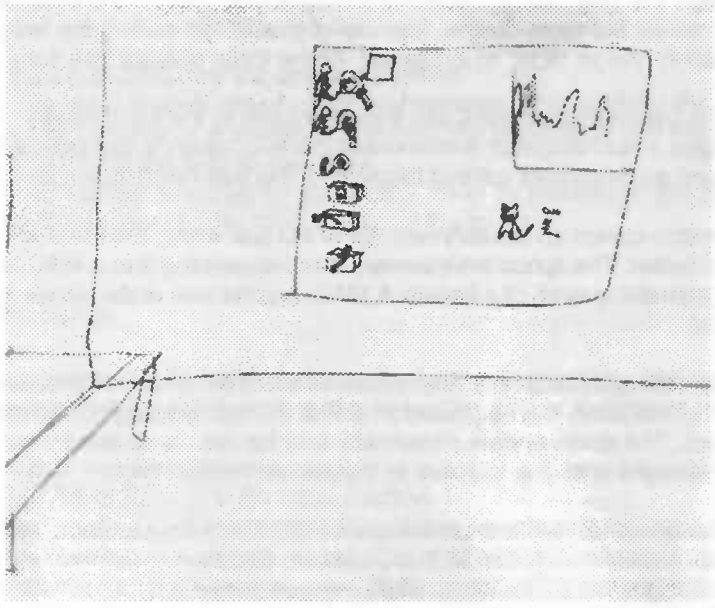


Figure 5-10: Viewpoint to the right side

On the right wall performance information of the switch is represented, as shown in Figure 5-10. The display gives information on the number of incoming calls, outgoing calls, internal calls, external calls, number of times that a station was busy while there was an attempt to make a call, the distribution of calls over the day and the time that the system has been down.

The displays contains 3D-icons like presented in Table 5-1. When selecting those the performance data will pop-up. The user can walk freely through the room.

Evaluation of Performance Room:
Advantages: all data in one room
Disadvantages: mostly 3D navigation
Interesting points: all data in one room

5.2.4 Choice of design

To make the right choice, these three designs will be compared on the basis of some criteria. The criteria are derived from the seven presence elements and completed with the criteria "use of 3D" and "overall view". The element "Communication and interaction with (lots of) users" is left out because the demonstrator "Virtual SOA" is a stand-alone application and is not multi-user. "Sufficient input and output" is based on the hard and software and is also left out. With "use of 3D" is meant that the design really is a 3D space and that all the data is presented in 3D. This criterion indicates if the design has added value on 2D designs. In Table 5-2 the result is presented.

Criteria	"Performance Space"	"Performance Inside"	"Performance Room"
spatial representation	-	+	+
freedom of movement	+	+	+
real world control	-	+	+
responsiveness	+	+	+/-
autonomous processes	+/-	+	+/-
use of 3D	+	+	+
overall view	-	+	+

Table 5-2: Choice of design

A spatial representation covers mostly a mental map, and in a lesser way metaphors. For the space design it is very hard to construct a mental map, but for the other designs it is not difficult, especially not for the room design. The use of metaphors makes the recognition of the data represented in "Virtual SOA" much easier. All the three designs use some metaphors.

For VEs it is important that users can freely navigate in the environment. This is possible in all the three designs. Free navigation works only if the MAC-analyst can build a mental map of the environment, so he can not get lost because of the free navigation possibility.

When a design covers an environment out of the real world, the MAC-analyst understands the application better. The space environment is not something that a MAC-analyst sees everyday, but a room is something everyone knows. A MAC-analyst also understands the inside design with his background.

It is hard to evaluate the criteria responsiveness and autonomous processes on paper. We can predict it, but not prove it. The prediction is that the inside design is the most responsive and autonomous. The doors open automatically and the data is updated when new data is provided. The other designs only give the data at the moment of selection.

All the spaces use 3D, but are not completely 3D. For example, most data is actually presented in 2D. With an overall view is meant that the whole world is visible from one point of view. The inside and room designs have this option, but the space design not.

When you look at the table now, it is clear that the space design is the worst. The other designs come close together, but we prefer the inside design. We find this design more challenging, and the MAC-analyst has enough technical knowledge to recognise all the objects. So, the choice here is "Performance Inside". This design will now be further worked out in the technical design.

5.3 Requirements engineering

In this section we describe the user requirements, software requirements and presence requirements. The user requirements describe the tasks the user wants to perform with "Virtual SOA". On the basis of these user requirements the software requirements are derived. The

software requirements describe how the user tasks can be performed. In the subsection presence requirements the guidelines of Chapter 4 are applied on "Virtual SOA" to improve the design and as starting-point for the functional design.

5.3.1 User requirements

In this section we describe some user requirements, one task of every performance group. The use-cases which will be described here are:

1. Retrieve the number of times a station was busy.
2. Retrieve the maximum waiting time of a group of stations.
3. Retrieve the distribution of calls over the day of the switch.

Use case 1: Number of times a station was busy
Short description: The MAC-analyst searches for the number of times a station was busy by it's number.
Preconditions: The MAC-analyst is authorised to retrieve the information and the viewpoint of "Virtual SOA" is pointed from the side to the PABX.
Interactions: The MAC-analyst searches for the station by the station's number. Then he focuses on the right data.
Postconditions: The number of times the station was busy is presented to the MAC-analyst.
Exceptions:

Use case 2: Maximum waiting time of a group of stations
Short description: The MAC-analyst searches for the maximum waiting time of a group of stations by the number of the group.
Preconditions: The MAC-analyst is authorised to retrieve the information and the viewpoint of "Virtual SOA" is pointed from the side to the PABX.
Interactions: The MAC-analyst searches for the group by the number of the group. Then he focuses on the right data.
Postconditions: The maximum waiting time of the group is presented to the MAC-analyst.
Exceptions:

Use case 3: Distribution of calls over the day of the PABX
Short description: The MAC-analyst searches for performance information of the switch.
Preconditions: The MAC-analyst is authorised to retrieve the information and the viewpoint of "Virtual SOA" is pointed from the side to the PABX.
Interactions: The MAC-analyst searches for the switch. Then he focuses on the right data.
Postconditions: The distribution of calls over the day of the switch.
Exceptions:

5.3.2 Software requirements

The software requirements can be translated from the user requirements. The software requirements describe what the MAC-analyst should be able to do with "Virtual SOA".

Use case 1: Number of times a station was busy

1. The viewpoint in "Virtual SOA" is in it's initial position.
2. The MAC-analyst should select a station by it's number by selecting the right entries to it.
3. The performance elements are directly visible. The goal here is to directly present the information (when the viewpoint is close enough) if possible. The performance elements are: number of incoming calls, number of outgoing calls, number of internal calls, number of external calls, number of times busy, the distribution of calls over the day and the state of the station. The number of times the station was busy is represented by the busy-bar of the 3D diagram.

Use case 2: Maximum waiting time of a group of stations

1. The viewpoint in "Virtual SOA" is in it's initial position.
2. The MAC-analyst should select a group by it's number by selecting the right entries to it.
3. The performance elements are directly visible if possible. The performance elements are: number of incoming calls, number of outgoing calls, number of internal calls, number of external calls, number of times busy, the distribution of calls over the day, number in the waiting queue, number of waiting callers who hang up, maximum-, minimum- and average waiting time and

the number of group members. The maximum waiting time is represented by the clock with 75 percent of it shaded.

Use case 3: Distribution of calls over the day of the PABX
1. The viewpoint in "Virtual SOA" is in it's initial position.
2. The MAC-analyst should search in the environment for the switch. The switch should be directly visible. The MAC-analyst can "walk" through the space towards the PABX.
3. If the MAC-analyst is near enough the performance elements become directly visible. The performance elements are: number of incoming calls, number of outgoing calls, number of internal calls, number of external calls, the distribution of calls over the day and the system-down time. The distribution of calls through the day is presented by the checker-board.

5.3.3 Presence requirements

In Section 4.3, we described the guidelines of the seven presence elements and supplemented them in Section 4.4 and Section 4.5. In this section, we apply all these guidelines to the SOA system. First we work out the seven presence elements guidelines and then the supplementary guidelines.

- **Seven presence elements guidelines**

The first element we apply is *"a spatial representation of the environment"*. Developing a cognitive map, i.e. a mental model, of the overall virtual space is a fundamental task , underlying successful travel and interaction with the environment. A cognitive map encodes the relative locations as well as the attributes of phenomena in spatial environments. A good map supports all other spatial navigation tasks, such as searching for a target spot. And it aids in maintaining situation awareness - your understanding of where you are in the overall context.

Technique	Description
Landmarks	Mark scenes with landmarks
Flying	Fly like a bird through the VE
Bread-crumb markers	Drop bread-crumbs while travelling through the VE
Map view	Small overview map of the VE

Table 5-1: Techniques to support the development of an adequate cognitive map

There are some techniques that can be applied in the design to let the user develop an adequate cognitive map, represented in Table 5-1. Landmarks are marks or signs or other features within the scene that help the user to keep track of where he is. The user recognises them when he sees them the second time. Flying like a bird through the VE allows users to change the horizontal and vertical position of their viewpoint. The user can explore the environment from all viewpoints. Bread-crumb markers can be dropped to see the route the user has been taken, like in the Hans and Gretel fairy-tale. The user recognises the scene quickly when he has been at the scene before. The map-view technique provides a small overview map of the environment, with a "you are here"- marker. The user has an image of how the complete world is and in which part of the world he is.
Magical "transporters" or other forms of automated travel should only be used when they add extra functionality. If you use such a technique, then try to use it only when the user can see where he goes, for example through PABX lines, and only make big jumps when it is described or viewed clearly. You should make differences between the several rooms or scenes, for example by colour, size or "furniture". If there are more duplicates of a room, try to make them recognisable . If all the rooms are the same the user needs to keep more information in his memory to remember where he is. If a VE provides real depth view the users can place all the objects of the world in the correct relative position. Depth can be created by image textures. This costs a lot of performance power so it decreases the frame rate. Depth can also be created by grid or horizontal lines textures. Make the grids as large possible and the space between lines as large as possible to get the least loss of frame rate.
It depends on the PABX which technique is the best. If the PABX does not have a lot of lines and stations, the bread-crumb technique can be used. If the PABX has to much connections, there will finally be bread-crumbs everywhere and the unvisited parts will be hard to find. Landmarks are always useful. A user sees very quick if he has been there before and can use that information to compose the cognitive map. An easy way of travelling in VEs is flying. There is total freedom and the user can see everything from all sides, like front-view, top-view, etc.. A map-view is a very good technique, especially when the environment is very large. The user can see in what part of

the environment he is and where he has to go. A good supplement is registrating the way a user has travelled, with lines or bread-crumbs. The following is applied on SOA:

- A map-view of the overall environment on the screen.
- Signs with telephone numbers and exit signs to indicate the way the user has to travel.
- The user can fly through the environment.
- Automated travel through the small places like the lines.
- Different kind of rooms/spaces are not equal equipped.

The second element is the *"freedom of movement"*. Users should be able to move freely through the environment and explore the virtual world. All the "six degrees of freedom"-movements should be available. Also, if a part of the environment invites a user to explore that part, then there should be really something visible in that part. So do not make empty rooms.

Collision detection is a powerful tool. Collision detection means that objects can not collide with each other, i.e. that a ball can not go through a wall. If software does not provide it, than do not make small rooms or scenes. It is very hard to navigate precisely through very small spaces without colliding with the walls. If software provides it, but navigation does not go smoothly, choose what is most important: better navigation or collision detection. If you can not use collision detection but also want to make small scenes, it is an idea to use them only for walk-through, no other actions, and it is possible to use automated travel to move through these tight spaces. Automated travel is one of the travel metaphors. Table 5-2 covers an overview of travel metaphors.

Travel metaphor	description
Scene-in-hand	the users hand controls the VE
Flying	flying like a bird through the VE
Walking	walking through the VE
Hovering	combination of flying, walking and looking around
Automated travel	select target and you are automatically moved there

Table 5-2: Travel metaphors

With scene-in-hand the user's hand position and rotations move the entire virtual environment. The user is essentially stationary and moves or turns the environment by hand movements. With flying the user flies like a bird through the virtual world by use of controls and tracking. This is the most general, least restricted form of movement. A metaphor similar to flying is walking, except that the user is constrained to the ground now. Hovering combines intervals of either flying or walking with periods where there is no forward movement and the user's direction of gaze is used to look around from the new position. Automated travel is also referred to as *put-me-there* travel or *point-of-interest* travel. This technique allows users to select their target location and then they are instantly transported to the new location.

If the frame rate gets to low, the screen update is not continuous anymore and people do not have the feeling of immersion anymore, and they have to wait until the results of their movement actions are visible which provides no freedom. It is recommendable to keep the frame rate high enough, with a minimum of 12 frames per second to stay immersed [Esposito]. You can increase the frame rate by simplifying the imagery. If no complete freedom of movement is possible in a part of the VE, try to find a metaphor for it which disables freedom of movement. For example, a user may not enter a specific room for a reason, then remove the handle of the door and put a sign with a message like "out of order" on it.

The following are applied on SOA:

- There are no empty rooms. Only the blocks which are used for the number selection are empty. They only contain signs.
- Use collision detection, especially for small spaces like the lines and stations.
- Use automated travel to travel through lines.
- Simplify imagery of objects when the viewpoint is not close to the object. This gives a higher frame rate.

"Real world control" is a very logical element. Users are used to interact in normal life, and there are several standard ways to interact. Some objects are used only for one function and when people see such an object they directly know what to do with it. Some examples are:

- to open a door you have to pull down the handle.
- to answer a phone call you have to pick up the receiver.

- to get water you have to turn on the water-tap.

So if users of "Virtual SOA" have to make a specific action, then try to find a real world metaphor which tempts the user to that specific action, just like the three examples above. The following are applied on SOA:

- The use of known objects In "Virtual SOA", like PABXs, stations and lines.
- The doors of the PABX which open automatically invite the user to enter the PABX.

"Communication and interaction with (lots of) users" is the fourth element. This element covers the interaction in multiple user systems, like avatars and MUDs. The demonstrator "Virtual SOA" won't be a multi-user system. But if a VE-system will be used in the future for the PABX management, including Performance Management and Fault Management, it can be a multi-user system. There won't be a lot of interaction. There can only be one user at a time within a specific PABX. If multiple users are making changes at the same time in a PABX, you get a communication and synchronisation problem. The changes of one user are overwritten by the changes of an other. There has to be made an indication in the VE that the specific PABX is already "occupied". A form of interaction you could think of is a chat function or room, where colleagues can consult each other.

The *"responsiveness"* of a system stimulates the attention of the user. It can be used to make the environment more attractive and entertaining, but also for serious subjects like alarm reports and functionality of the system. For example, a door could open automatically when a user approaches. Users should be able to ask the system for specific information and the system should respond with the representation of that information. The system also has to provide feedback, auditory, visual or other, on user actions, like creaking of a door which opens. "Virtual SOA" could respond in various ways to the users. There could be a welcome sound with information of the PABX, like company, model, type etc.. When a user enters a virtual PABX the light could turn on. The following are applied on SOA:

- The PABX door opens automatically when the user approaches it.
- The system provides auditory feedback to help the user navigate and inform him.

In real life a lot of *"autonomous processes"* happen, which you mostly can not influence. Some processes can be influenced, like stopping a car by jumping in front of it, but are not common. An autonomous process can start as a response to a user action, like the "opening door" example when users approach it. Real autonomous processes start spontaneous. You could think of rain falling down, or the waving of a tree. In a VE try to find processes which are familiar with the environment. Do not make a cow grazing in the centre of a city. The following are applied on SOA:

- The objects which represents the performance data automatically update after changes.

A VE should provide *"sufficient input and output"*. The hardware can be to slow to handle the input and output of a VE. This will reduce the feeling of presence of the user of the VE. Extra input and output devices might help, but is not always possible in the operating environment. Also the software can be to slow to handle all the actions in time. It may be possible to avoid this problem by designing a simple world with simple graphics. There will be more system performance left for auditory and other forms of Input and output.

- **Supplementary guidelines**

To avoid health and safety problems with the use of VEs and VR-devices, there are some things to hold on to. First it is important to make tasks who are practicable within 10 minutes. If the user has to wear a HMD longer than 10 minutes he can suffer from nausea or eyestrain. Also let the user sit in a chair during his tasks or the room has to be empty, because the user does not notice the system environment. Try to avoid environments with small scenes and high scenes, to avoid anxieties like claustrophobia and acrophobia. Applied on "Virtual SOA" this gives the next adjustments:

- It seems that tasks can be performed within 10 minutes. After testing the application we can confirm this or reject this.

5.4 Functional design

In this section the functional design is developed. On the basis of the software requirements we describe the functionalities that "Virtual SOA" has to provide for the interaction between the user and the application. In the next section this functional design will be used for the technical design.

We describe the functionalities by translating the three use-cases to functionalities of "Virtual SOA".

Use case 1: Number of times a station was busy
1. The viewpoint in "Virtual SOA" is in it's initial position, an overall view of the switch and the stations, lines and trunks.
2. The MAC-analyst clicks on the doors of the PABX. "Virtual SOA" moves the viewpoint in front of the PABX. The doors open automatically when the viewpoint approaches.
3. The MAC-analyst moves inside the PABX. On the right side he sees the starting points of the internal lines.
4. The MAC-analyst selects the upper select-block by clicking on it. The viewpoint moves automated to the inside of and through the line. The user can also fly through the line.
5. In the block the MAC-analyst should select the next line by clicking on the corresponding select-block and the viewpoint travels through the next line.
6. Steps 4 and 5 are repeated until the station is reached.
7. The MAC-analyst has arrived inside a station.
8. The performance elements are directly visible when the viewpoint is close enough, else the MAC analyst should get closer to the element for a detailed view. Look at the busy bar to collect the necessary information.
9. To return to the initial viewpoint the MAC-analyst should take the same way back.

Use case 2: Maximum waiting time of a group of stations
1. The viewpoint in "Virtual SOA" is in it's initial position, an overall view of the switch and the stations, lines and trunks.
2. The MAC-analyst clicks on the doors of the PABX. "Virtual SOA" moves the viewpoint in front of the PABX. The doors open automatically when the viewpoint approaches.
3. The MAC-analyst moves inside the PABX. On the right side he sees the starting points of the internal lines.
4. The MAC-analyst selects the lower select-block by clicking on it. The viewpoint moves automated to the inside of and through the line. The user can also fly through the line.
5. In the block the MAC-analyst should select the next line by clicking on the corresponding select-block and the viewpoint travels through the next line.
6. Steps 4 and 5 are repeated until the block is reached.
7. The MAC-analyst has arrived inside the group block.
8. The performance elements are directly visible when the viewpoint is close enough, else the MAC analyst should get closer to the element for a detalled view. The MAC-analyst has to get close to the maximum waiting time clock, the one with 75 percent of it shaded. The maximum waiting time is represented on the display.
9. To return to the initial viewpoint the MAC-analyst should take the same way back.

Use case 3: Distribution of calls over the day of the PABX
1. The viewpoint in "Virtual SOA" is in it's initial position, an overall view of the switch and the stations, lines and trunks.
2. The MAC-analyst clicks on the doors of the PABX. "Virtual SOA" moves the viewpoint in front of the PABX. The doors open automatically when the viewpoint approaches.
3. The MAC-analyst moves inside the PABX. On the right side he sees the starting points of the internal lines and the performance objects.
4. The performance elements are directly visible when the viewpoint is close enough, else the MAC analyst should get closer to the element for a detailed view.
5. The MAC-analyst should check the checker-board to read the distribution of calls through the day.

5.5 Technical design

On the basis of the functional design and especially the presence requirements we made a technical design. We also describe the technical details of providing the VE with performance data. The performance data is provided by a CDR generator, which is described in Section 5.5.1. In Section 5.5.2 an object oriented design is given for the simulation of the performance data, supplemented with a translation of the functional design.

5.5.1 CDR Generator

We start with the providing of performance data. In section 3.5.2.2 we already described that data is needed from CDRs, alarm logs and derived with software from the hardware. Because the operating environment is not connected to a PABX we can not derive the software from the hardware. That is why we simulate this data. Also the data out of the alarm log is simulated, because this data is not much used. With a CDR generator a CDR is generated.

The CDR generator is especially developed for the "Virtual SOA" application. In this application will be a total of 8 telephones, with the numbers "11111" till "11118". The first four telephones, numbers "11111" till "11114", belong to a group of stations. The number for this group of stations is "11119". For the design of the generator we selected combinations of A- and B-Numbers, because not all combinations are likely. In Table 5-3 the selected combinations are presented.

Anr \ Bnr	11111 - 11114	11115 - 11118	11119	external number
11111 - 11114		✓		✓
11115 - 11118	✓	✓		✓
11119				
external number	✓	✓	✓	

Table 5-3: Selected combinations of A-numbers and B-numbers.

The numbers are divided in the next four categories:

1. 11111 - 11114: The numbers of the single stations belonging to the group of stations.
2. 11115 - 11118: The single stations which do not belong to the group of stations.
3. 11119: The number of the group of stations.
4. external number: A number belonging to the public telephone network.

It is not very likely that two users of a station in the first category call each other, because they work in the same room. These users also will not call the number of the group. Also the users of a station in the second category will not call to the group number, because if they want to call an operator of the group of stations they will call the direct number. There will never be a call made by the number "11119", because this is a group number and it does not belong to a station.

In "Virtual SOA" the calls will be made between 7:00 in the morning and 19:00 in the evening for the stations of category 2. There are always people who start early and people who finish later. For the group of stations the calls will be between 9:00 and 17:00, a standard working day. The distribution of calls over the day is not linear. Between 7:00 and 9:00 and between 17:00 and 19:00 there are very few calls, and only of stations in the category 2. The most busy times are between 11:00 and 12:00 and between 14:00 and 15:00. The generator will generate CDRs with this distribution. In Table 5-4 the distribution is given.

Period	Time	Group station	No group station
1	7:00 - 9:00		4
2	9:00 -11:00	40	6
3	11:00 - 12:00	30	12
4	12:00 - 14:00	40	6
5	14:00 - 15:00	30	12
6	15:00 - 17:00	40	6
7	17:00 - 19:00		8

Table 5-4: Distribution of calls over the day.

The first two columns show the periods defined in the CDR generator source and the corresponding times. The last two columns give the number of calls which are made to or by stations respectively belonging to and not belonging to the group of stations. It is clear that the group of stations will receive and make the most calls.

For the generation of CDRs the CDR generator will use an input file, "cdrgen.ini". The syntax of a line of the input file is: *<TimePeriod> <','> <GrpNoGrp> <','> <Number>*

The TimePeriod is the period from Table 5-4. The GrpNoGrp field indicates if there is a group station involved or not. GrpNoGrp equals 1 if a groupstation is involved and 2 if not. Number indicates the number of calls that are made in that period.

The CDR generator creates CDRs by period and if there is a group station involved or not. These CDR' s are after creation inserted in a binary search tree based on the answering time. After all the CDRs are created the inorder of the tree is written to the output file "cdrgen.dat". This output file will be the input file for "Virtual SOA". The format of lines in this file is:
<A-number> <','> <AnsweringDate> <','> <AnsweringTime> <','> <B-number> <','> <ElapsedTime>.

The source of the CDR generator can be found in Appendix A.

5.5.2 SCL script

The data out of the CDR is read with the script language SCL of Superscape, which has standard I/O-functions like read ("fgetc") and write ("fputs"). To read the input file "cdrgen.dat" we defined a procedure ReadNr. This procedure reads characters from the input file and returns an integer value. User defined procedures have to be placed at the RootObject. In every other SCL script of any object these user defined procedures can be used. The "main program" is connected to PBXHold. The script of PBXHold reads the data from the input file and sends the values with messages to the other objects. The objects can communicate with a standard message system. The objects who are important in the simulation are shown in Figure 5-11.

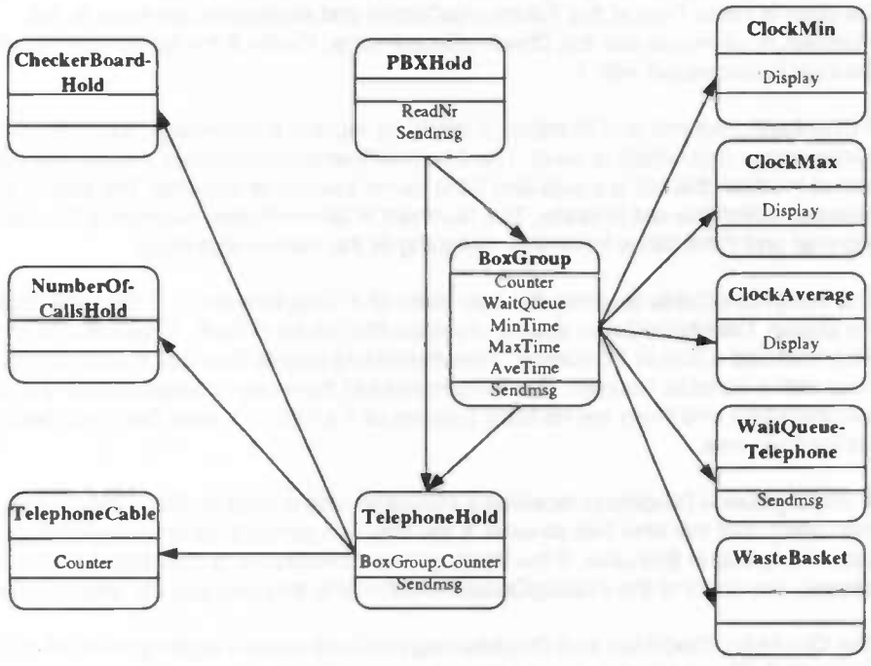


Figure 5-11: Important objects in the simulation.

PBXHold reads the input from the CDR input file "cdrgen.dat". First is waited until the AnsweringTime of the call is reached. Then is checked if the call is internal. If the call is internal then is checked if the station with B-number is free. If this station is free, then messages with data are send directly to both stations, i.e. to TelephoneHolds. If the station is busy then only a message is send to the station with the B-number with the message that the station was busy when someone tried to call it. If the call is external, then first is checked if the A-number is an internal number, i.e. if it is in the range 11111 till 11118. It can not be 11119, because this number does not belong to a station and external calls can not be made with that number. If the A-number is internal, a message is send to the station with the A-number. Otherwise is checked if the B-number is in the range 11111 till 11118, and a message is send to that station in that case. If also the B-number is not internal, it can be equal to the group number 11119 and a message is send to the BoxGroup.

The BoxGroup first checks if there are stations free and if there are waiters in the WaitingQueue. There are stations free if Counter is less than 4. Counter is the number of busy stations. This variable is changed by the TelephoneHolds at the moment that they become busy or free. The WaitingQueue is of the FIFO principle which stands for First In First Out. The element which is first put in the queue, Is removed first too. If a call is removed from the queue, the minimum (MinTime), maximum (MaxTime) and average (AveTime) waiting times are calculated and send to respectively ClockMin, ClockMax and ClockAverage. If there is a free station and a waiting call, the first call in the WaitingQueue is send to a free station. This can be repeated till all stations are busy or there are no calls in the WaitingQueue anymore. Then BoxGroup checks if there are messages waiting. BoxGroup receives two kinds of messages:

1. A message from PBXHold with a new call.
2. A message from a WaitingQueueTelephone, that the caller hang up.

If the message is a new call, the BoxGroup checks if there is a free station. A message is then send to the free station. If there is not a free station, the call Is put in the WaltIngQueue and a WaitingQueueTelephone object is created which floats in the BoxGroup. If the message comes from a WaitingQueueTelephone, the call is removed from the WaitingQueue and the WaitingQueueTelephone object is placed in the Wastebasket.

If a TelephoneHold receives a message it first checks if the state of it is Incoming, Outgoing or Busy. If the station was Busy only messages are send to the NumberOfCallsHolds. If the station was free, messages are send to the TelephoneCables that the state has become Busy and the Counter of the BoxGroup, which contains the number of busy group stations, is increased with 1 if the station belongs to the group. Then there is waited until the ElapsedTime has passed. Following the state is set to Free at the TelephoneCables and messages are send to the NumberOfCallsHolds and the CheckerBoardHolds. Finally if the station belongs to the group the Counter is decreased with 1.

If CheckerBoardHold and NumberOfCallsHold receive an message, they visualise the performance data which is send. The CheckerBoardHold receives a message which includes the period in which the call is made and if the call is Internal or external. The period covers the hours between which the call is made. The NumberOfCallsHold also receives if the call is internal or external and if the call is Incoming, Outgoing or the station was Busy.

The TelephoneCable receives the new state of a TelephoneHold. If the cable belongs directly to the station TelephoneCable directly changes the colour of itself. If the cable is between the PBXHold and a Box or BoxGroup, TelephoneHold counts how much stations connected to it are busy with a variable Counter. The TelephoneHold then only changes colour if a station's state becomes free and there are no busy stations or if a station's state becomes busy and the other stations are free.

A WaitingQueueTelephone receives a message with a random time. The WaitingQueueTelephone then waits until this time has passed. If the time has passed the message Stopped is send. The caller hangs up in this case. If the WaitingQueueTelephone is connected before the time has passed, the script of the WaitingQueueTelephone is stopped and the object is deleted.

The ClockMin, ClockMax and ClockAverage objects receive waiting times when waiting calls are connected. These objects change their displays on base of these messages.

It is not necessary that the inside of objects like the PABX and the stations are always visible. They are only visible when the viewpoint is inside the object. From a distance objects don't have to be very detailed, because you can't see the details from large distances. Detailed objects can be replaced by lesser detailed objects when the distance is to big to see any details.

The SCL code of "Virtual SOA" can be found in Appendix B.

5.6 Conclusions

In this chapter we described the design of "Virtual SOA". We made three designs and chose one of them. The "Performance Space" is a very abstract world with floating objects, "Performance Inside" covers looking in the inside world of a PABX and in "Performance Room" we tried to collect a lot of data in one room. The worlds on their own are all 3D worlds, where the user has to travel In 3D. The hard factor in all of the designs is: "How do you visualise (a lot of) data in 3D?". In all

designs the 3D visualisation is not really achieved. The data uses only two dimensions and are mapped on 3D objects. We chose "Performance Inside" as VE interface of "Virtual SOA".

In the section "Requirements engineering" the requirements on the functionality of "Virtual SOA" are described, together with the presence requirements. The presence requirements are the guidelines from Chapter 4 applied on SOA. In the section "Functional" the functionality of "Virtual SOA" is described, on base of the use cases of the requirements engineering. In the technical design the data simulation is described. CDRs, generated by the designed CDR generator, will be read and handled by SCL programs in Superscape.

In "Virtual SOA" we chose for a standard configuration of in total 8 telephones. These are connected in groups of 4 telephones to a box, which is connected to the PABX. One of the boxes is a group of stations. The CDR generator and the SCL script are all adjusted to this configuration.

6 Implementation of "Virtual SOA"

6.1 Introduction

The step following on the design of "Virtual SOA" is the Implementation of the design. The software and hardware use for the implementation are described in Section 6.2. Then the 3D world "Virtual SOA" is described, using some screen captures. In Section 6.4 we evaluate "Virtual SOA". This evaluation is based on the comments of Human Factors and VE specialists at KPN Research. In the last section the changes are describe which are made on "Virtual SOA" after the evaluation.

6.2 Software and hardware

6.2.1 Software

The "Virtual SOA" application is implemented with Superscape, which is chosen within the project Devine. This authoring tool consists of three parts:

- Superscape VRT (Virtual Reality Toolkit) - an integrated design environment for the design of virtual environment applications.
- Superscape Developers Kit (SDK) - a toolkit for programmers to create extra functionality for the Superscape system.
- Superscape Networks (Net) - an extension of the VRT for the design of multi-user VEs.

Superscape is the best known 3D authoring tool for the PC platform. Using this platform and authoring tool can indicate where they stand in the world of VE design. The choice is explained in the report "Devine: selectie van een Virtual Environment authoring tool" [Bergh]. In section 6.2.3 there will be an extensive description of Superscape. The operating system used is MS-DOS Version 6.20.

6.2.2 Hardware

The platform used for the design is a Compaq Pentium 120 MHz PC with 16 MB RAM and a Diamond Stealth 64 video card. Besides a PS/2 Style mouse there are special VR-devices like a SpaceMouse and I-Glasses of Virtual I/O (HMD-set).

6.2.3 Experiences with Superscape

Superscape VRT is a Virtual Reality studio for desktop computers. It consists of a suite of editors which let you create 3D Virtual Worlds, and Visualiser which you use to display and interact with your worlds. The Visualiser can also be freely downloaded from the Internet at "http://www.superscape.com" as a stand alone program. To help you create realistic worlds you can add textures and sounds to objects in the world, and different lighting setups. You can use the Superscape Control Language (SCL), a programming language based on the popular C language, to control objects in the world and perform complex actions.

Visualiser is designed so that all basic movement and interaction with worlds can be made from the mouse and keyboard, although a peripheral proportional device (such as a SpaceMouse or joystick) is recommended. VRT contains a data converter which lets you import data from different formats into VRT - a module for importing the popular DXF data format used in many CAD packages is included in VRT. You can connect VRT to external programs using DDE links to provide real-time visual representation of computer data in the standalone Windows Visualiser application.

VRT contains seven editors which you use to create virtual worlds, and control how the end-user interacts with them. The two most important editors are the Shape Editor and the World Editor, which you use to create the shapes and objects that make up each world. The Image Editor and Sound Editor allow you to import and create pictures, textures and sounds to enhance the appearance of the world. The Layout Editor, Resource Editor and Keyboard Editor allow you to customise the computer screen and keyboard to provide additional information and interaction with the user.

In the next subsections a description of used editors and parts of VRT are described. The experiences with these editors are described too.

6.2.3.1 Shape Editor

You use the Shape Editor to create points in space which are connected together to define two dimensional facets. You use these facets to assemble 3D shapes. A shape can be simple as a single facet or as complex as you wish. The Shape Editor has several editing features - such as copying, duplication, extrusion and transformation - which speed up the creation process. Once you have created a shape, you can edit its colour attribute and create an animation sequence producing a more realistic and usable shape.

A point is the smallest component of every item in the virtual world, simply defined as a point in 3D space. There are two kinds of points in VRT, the relative and the geometric point. A relative point is a point which is given an X, Y, Z position, referenced to the origin of the facet's bounding cube. The bounding cube is an invisible cube used by VRT to sort objects within the virtual world. Bounding cubes are displayed in the Shape Editor as a cube outline with X, Y and Z axes. A geometric point is a point which is defined in terms of where it is proportionately on a line between two other point, know as the anchor points. The advantage of geometric points is that they are processed faster, help to maintain proportions during animation and provide a minimum of adjustments in complex shape changes. Moving the starting point moves all of the attached geometric points.

If you want to make round facets, you have to perform a lot of work. There is no utility of defining several points on a circle. You have to calculate the position of every point on the circle and create a relative point for each.

Facets are surfaces created by connecting points together. A single point is displayed as a single pixel facet, two points create a line facet one pixel thick and three or more points create a flat surface facet. Single point and line facets are always visible. Flat surface facets, however, are directional. They have a front but no back. Each facet that you create is given a facet number that VRT uses to reference it during processing and rendering. The order of facets within a shape is one of the most important factors in producing a convincing shape. To render a frame correctly and display a convincing virtual shape, VRT must know the correct order to draw the facets. Facets at the back of a shape must be rendered first so that they do not obscure facets at the front of the shape. Each shape has a facet list which controls the order in which the facets are drawn. You define this order using the Facet Editor. This editor also provides functions like reordering facets, moving facets, swapping facets, flip facets, colour facets, delete facets and copy facets to another shape. This latter function can be very useful when you are creating several shapes with the same complex facets. Also textures can be placed on facets and the attributes of facets can be changed with the Facet Editor. With the Shape Editor also animations and animated colours can be created. Normally a shape consists of 1 cell. With animations a shape consist of more, user defined cells. Within each cell the shape can be modified on colours or appearance. There are a couple of animation modes like different loops and a mode for controlling animations by SCL.

6.2.3.2 World Editor

The World Editor lets you build virtual worlds from the shapes that you create in the Shape Editor. You can construct individual objects and then use them to create more complex objects called groups. You can position objects, resize them, edit their colours, and apply textures. To create realistic movement in a world objects can be bent around a point, rotated and animated. Using SCL, complex patterns of movement, interaction, and conditional control can be build into objects.

The World Editor is to complex to describe completely here, so we only describe the parts we used. If you want to create objects, you can select different shapes for it. Standard are the group shape and the cube shape. The group shape defines a group which can hold other objects, known as children. This is very useful to create an object which is build out of several objects. This gives and ordered result and if changes are made on the group, they can also affect the children. The cube is a standard defined cube in VRT. Besides the group and the cube there can be selected user defined shapes, created with the Shape Editor. Also Virtual Clip Arts can be imported. If you need to create several identical objects there is a duplicate function. This duplicates the objects with all attributes and SCL script. The new object gets the same name, completed with a new object number. For example, a duplicate of Cube can be Cube[10]. After creating an objects all attributes can be changed if needed. The object can be renamed, resized, rotated or repositioned and the shape can be changed. An object can also be made moveable or animated. A moveable object can be controlled by the user, as viewpoint, or by a defined controller. For an animated object the objects shape has to have several cells and the animation can be controlled by SCL or by a predefined animation mode.

Objects in VRT can be ordered in an object tree. The object tree is very important in virtual worlds, allowing you to manipulate and control groups of objects together. Efficient grouping of relations can greatly improve the speed of virtual worlds, as it simplifies many of the calculations that VRT has to do every frame. Within the object tree, all objects are descendants of the root object, except movable objects are treated as siblings of the root object. With VRT you can assign objects as child of another object. In this way you create a object tree with the RootObject at the top of the tree.

In the virtual world you create, you can define several viewpoints. All viewpoints are attached to an object. You can attach a viewpoint to a moving object so that it moves with it, or you can move a viewpoint relative to its attached object using the input device. Finally, you can define a viewpoint which moves along a predefined path that could take you on a flying tour of your virtual world or that can be used for automated travel. A disadvantage of the viewpoint path is that it is always a closed loop. If you want to use it for automated travel, for example through small spaces, you have to create an object at the end of the path that can detect if the current viewpoint, the viewpoint which you see on the screen now, is inside it and stop the loop with SCL. It is very hard to place such an object and it is very time consuming.

As objects recede into the distance, the eye can determine less detail. A point is reached at which you can replace a detailed shape with a less complex shape without noticing the difference. Since simpler objects take less time to process there can be generated substantial gains in processing speed with this technique called distancing. With the Distance Editor you can insert replacements for the object and define the distance at which it is replaced by the next object in the list. The last object has a distance of 0.

Each object in VRT has flags. These flags that you can set for each object are very important, as they instruct the VRT system on how to sort and render each item. The flags are divided in the three categories general, initial and current. The general flags are valid through the whole simulation. The initial flags are only valid at the start and can change during the simulation and the current flags give the flags on that moment in the simulation, for example when you switch from Visualiser to World Editor and do not reset the world. Some flags often used are:

- General
  - * Movable: has to be set if you want to make a movable object.
  - * Replacement: has to be set if you want to use replacements on an object.
  - * Enterable: allows moving objects to enter the object's bounding cube.
- Initial
  - * Visible: makes an object visible when the world is first loaded or reset.
  - * Animate: turns on an object's animation when the world is first loaded or reset.
  - * Debug: debugging of SCL on an object is turned on by default.
- Current
  - * Visible: the object is currently visible.
  - * Animate: an object's animation is currently on.
  - * Movement: enables an object's movement.
  - * SCL: activates the objects SCL program.

An object's bounding cube should not overlap with the bounding cube of a sibling as VRT may not be able to sort them correctly from some viewpoints. If it is unavoidable, you can attach an object to a facet on a sibling object. The sibling object dictates the sorting order depending on whether the facet is visible or not.

6.2.3.3 Image Editor

The Image Editor lets you import or create images that can be applied to objects as textures, or used in the Layout Editor and Resource Editor as instruments or controls. You can use the Image Editor painting tools to create new images or edit pictures that you have imported. VRT supports a variety of common file formats. We only used the Image Editor to import images. This is necessary to use an image in VRT. The Image Editor can be used for simple images, but if you need realistic images you can better use another application for creating images and then import them with the Image Editor.

6.2.3.4 Sound Editor

The Sound Editor, in conjunction with your internal sound card, lets you record, play import and modify sounds. These sounds can be played using SCL from within the virtual world. We tried to record sounds with the Sound Editor, but were not really satisfied. The microphone had to be hold a great distance, a couple of meters, to avoid an irritating beeping noise because of the interference between the microphone and speakers. After avoiding this there is still a lot of noise in the sound. Therefore we recorded the sound in windows with a windows sound editor and imported the WAV file in the Sound Editor. It is preferable to turn the microphone of in the device settings or to unplug it from your sound card. If you have imported sounds, you can edit them with the Sound Editor. There are functions like cropping, cutting, copying and pasting sounds, filtering, fading and reversing. With cropping you can select a part that you want to keep. The rest of the sound is removed. Filtering removes noise in the sound. You can make sound fade in (volume goes from zero to normal) and fade out (volume goes from normal to zero).

6.2.3.5 Layout Editor

You can make the basic screen in the Visualiser more interesting and useful using the Layout Editor. It enables you to change the size of the window on to the virtual world, embed the window in a surrounding picture, add icons that trigger functions, and display instruments that show data or allow interaction with the user.

With the Layout Editor consoles, windows, icons and several kind of instruments can be created. A console can be provided with several windows, icons and instruments. An icon is an invisible active area, which performs a function if you activate it. Such an area can be used for creating buttons. An image instrument can be used as the visible button. Other instruments you can use are for

example different kinds of value instruments(decimal, binary, etc.), text instruments and user defined.

With the Layout Editor also a layout can be created for 3D Glasses or an HMD. We tried this but we did not succeed. We followed the instructions in the manuals, but it didn't work. Obviously it must be the VRT application, because the HMD did work with the provided games.

6.2.3.6 Resource Editor

The Resource Editor lets you create your own dialogue boxes and menus which can be used from within SCL. This gives an easy way of getting information from the user within Visualiser. We did not use it.

6.2.3.7 Keyboard Editor

All the keys on the keyboard have been assigned default functions that allow the user to control the editors and Visualiser. You can use the Keyboard Editor to remap the functions if these do not suite your needs. We also did not use this editor.

6.2.3.8 Visualiser

Visualiser is an application that lets you display and interact with the virtual worlds that you create using VRT editors. It is supplied as part of the VRT, or as standalone system which only has input device configuration. Up to thirty-two Visuallsers can be networked together with each user interacting in the same world in real time. In Visualiser you still can change the settings for viewing the world.

6.2.3.9 SCL

The Superscape Control Language (SCL) is similar in any respects to the popular programming language C. It is used to control objects within the virtual world, and performing actions which could not be performed by the automatic rotations or movement functions. SCL is very powerful and simple to use.

Each object in the world can have its own attached SCL program, which is executed once per frame. A simple method is provided to allow SCL programs to continue execution over several frames. SCL can perform some of the actions that are available through the World and Shape Editors, but it offers a much wider range of functions in the areas of position, angles and rotations, movement, animation, bending, visibility, colour and viewpoints. Each function generates or reports actions occurring in the virtual world. They are supported by a variety of mathematical and programming features.

A very positive point of the SCL Editor is, that it checks the syntax of the SCL code before closing the editing on an object. Also the structure of the code is ordered, which gives a very arranged code. Only if you use a lot of comments the structure is not always arranged as you would like it to be.

We already described that if you duplicate objects, also the SCL code is duplicated. It is big disadvantage that if you change the SCL program after duplicating, you have to change it by hand for every duplicate. This is logically because maybe you want the programs to differ. But if they have to be equal, it is very inconvenient. Especially if there are a lot of object numbers included in the code, of children of the object. Changes can be very laborious then. Another disadvantage is the print function in the SCL Editor. With this function you only can print to the printer port LPT1. If you want to save the code in a file, because you want to import it in a report or have to print it elsewhere it is not possible with VRT. Therefore we had to download a memory resident DOS program from the Internet, that capture the printer output and saves it into a file. This method worked very well.

You can define procedures yourself. You have to put these in the SCL program of the root object. You have to watch out when you make changes to this program. The SCL Interpreter obviously does not link calls to procedures to the procedures name, but to the procedures position in a list of defined procedures. If a procedure is removed in the middle of the program, it also is removed

from the table. In the other SCL programs where you use these defined procedures, the names of the procedures, which stood after the removed procedure in the root program, are changed to another procedure's name because another procedure corresponds to the number in the table after the removal. This is also true for variable names. Variables can be used in other object's programs. If a variable is removed the name of the variable in another object's program is changed. These changes can give a fault report because of a different number of procedure parameter, but if the number and types correspond, no errors are detected. Only a lot of thinks can mess up in your simulation.

6.2.3.10 Conclusions

Superscape is a very powerful but also complex system. It has a lot of functionalities to design 3D worlds. The Superscape VRT consist of several editors like a world editor, shape editor, image editor, sound editor, layout editor and resource editor.

Superscape is provided with many manuals. The information provided by the manuals is not complete. Sometimes the information is very general and not to the point, so the details you need are not provided. It happens also that you follow the procedure provided by the manual, and you have to perform it a couple of times before it works.

On the functionality part there can also be made improvements. For example on the part of the path definitions in the viewpoint. These are always loop-oriented. But mostly you need paths that are just one-way and not in a loop. This to be resolved with an object and SCL and is not very smoothly. It could be better an option to the paths to be a loop.

Do not underrate the learning time of working with Superscape. Because of the complexity and all supplied functionalities, a lot of time is needed to get to know the system. It took more time than expected in this research. Also if you have to use the SCL script, it is preferable that you have enough experience with programming and the programming language C.

The overall idea of Superscape is that it is a very powerful and complete system, although there can be made improvements to it. It can be that it is depending on the application area if Superscape has his shortcomings or not. The manuals could be a more extended with detailed information. You also should take enough time to learn working with Superscape.

6.3 "Virtual SOA"

In this section we describe the demonstrator "Virtual SOA" on base of some screen captures. In Figure 6-1 the starting viewpoint of "Virtual SOA" is presented.

Figure 6-1: Starting viewpoint in "Virtual SOA".

The console is divided in one big screen, two small screens and two lines of text. The big screen contains the current viewpoint of the user. The two small screens are map views, one from above and one from the front. The map views are to provide the user with good opportunities to create a mental map. The two lines of text contain extra information, like actions to do, position information and the result of a number request. Users can fly freely through the environment and go inside the PABX to enter the inside world. To prevent that users can fly through walls, we wanted to use collision detection. We did not use it, because of two reasons. First we needed to create collision cuboids, because of the holes in the objects we could not use the bounding cubes of the objects. Second, the navigation was very hard. It could be that the system did not had the good parameters, but it did not work very well. Because we do not use collision detection now, and the user has to navigate through small lines, we use automated travel. The first automated path is the path to the inside of the PABX. If the user clicks on the PABX-doors with the mouse, the viewpoint moves towards the PABX, enters the doorway with automatic opened doors and turns to the right. The viewpoint is now in the position of Figure 6-2.

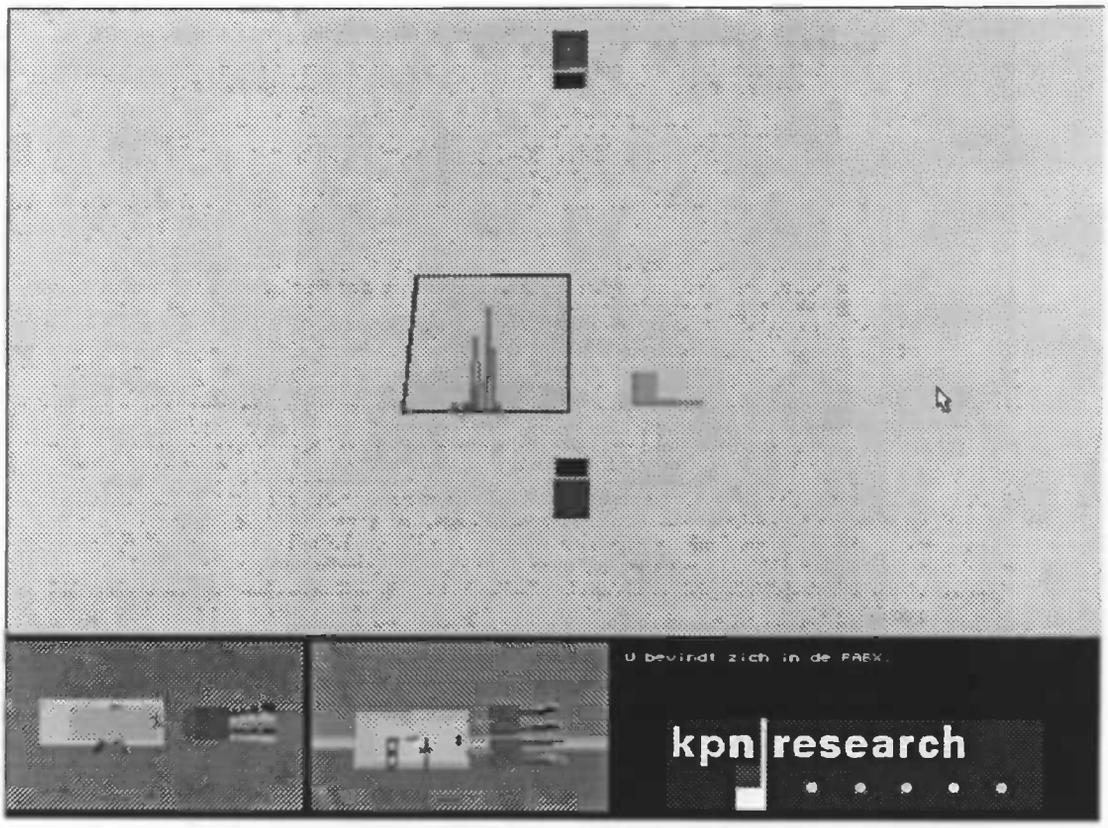


Figure 6-2: Viewpoint inside the PABX.

The users sees now a checkerboard, 3D bars and two blue blocks with signs. Below the map views indicate where the user is. The inside of the PABX is visible in both maps. The text line also indicates that the user is inside the PABX. The checkerboard indicates the distribution of calls over the day. When the user comes closer to the checkerboard, the grid of the board gets visible. The technique used for this, object replacement, increases the performance of the virtual world. The grid is divided in 24 by 24 squares. On the horizontal side a square is equal to one hour. In every square two bars rises. The green bar gives the number of internal calls in that hour, the yellow the external calls. The 3D bars give the number of incoming and outgoing calls and the number of times a station was busy. When the viewpoint is near the blocks, the signs become visible. These signs hold the numbers of stations connected to the line behind the block. If the user clicks on the block, the viewpoint moves to the block and disappears through in the line. The colour of the line can change, because the colour indicates the state of the line. Also during the travel through the line, the colour can change. At the end of the line the viewpoint stops in the box. In Figure 6-3 the viewpoint is presented inside the group box. The lowest block is selected then.

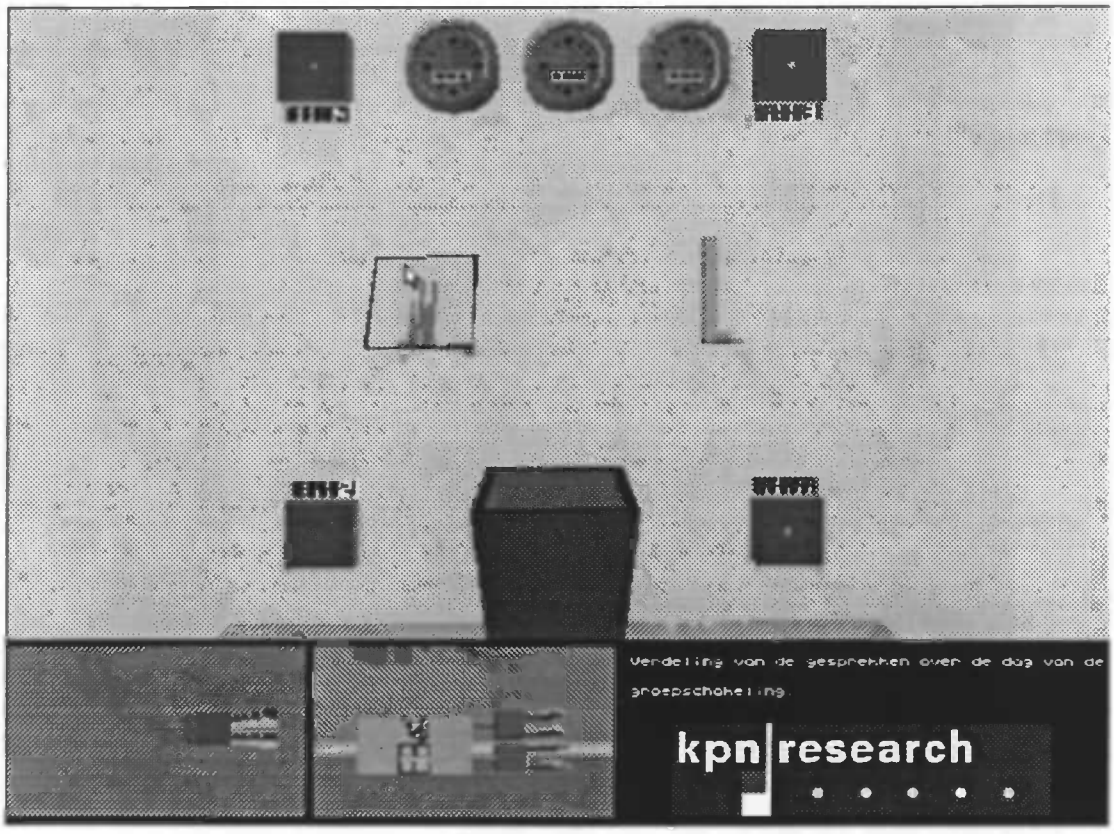


Figure 6-3: Viewpoint inside the GroupBox.

Inside the group box there are more objects visible than in the PABX. The checkerboard and the 3D bars are here too, but also a wastebasket and three clocks. On the map views is visible that the viewpoint is inside the box. When calls are in the WaitingQueue, there float little telephones around the wastebasket. Unfortunately there are no waiting calls at this moment. When the caller hangs up before getting contact, the telephone will be placed inside the wastebasket. After clicking on the wastebasket, the number of callers who hang up will be visible in the text lines. When the caller does not hang up, he is connected with an operator and his waiting time is kept. Out of all the waiting times, a minimum, maximum and average waiting time can be calculated. These times are visible on the displays of the three clocks. The left clock, for 25 percent shaded, presents the minimum waiting time. The clock in the middle, for 75 percent shaded, displays the maximum waiting time. The 50 percent shaded clock at the right has the average waiting time. Here the blocks also are starting points of lines. The signs indicate which station is connected. After selecting the block below on the right, the viewpoint enters the line to station 11111 and stops inside the station. The viewpoint is presented in Figure 6-4.

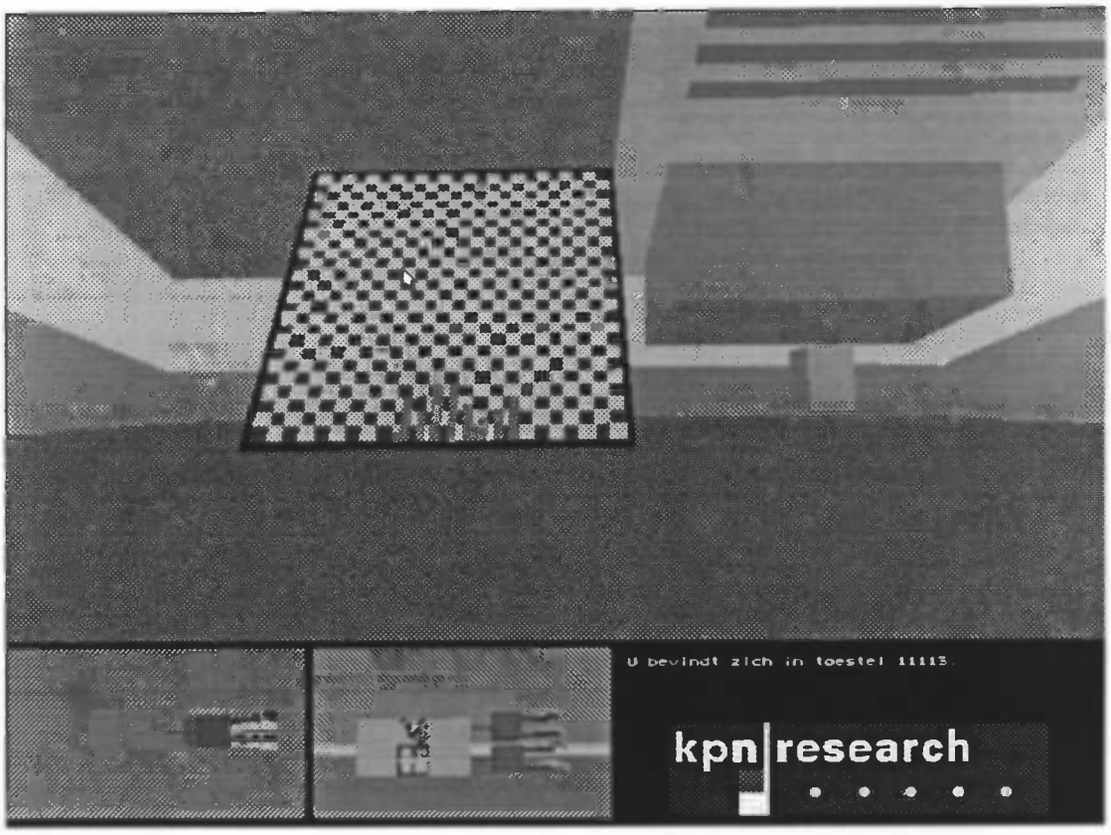


Figure 6-4: Viewpoint inside a station.

Inside the station the checkerboard and 3D bars indicate the performance of that single station. The inside of the station is visible on the map view to indicate that the user is inside the station. this is also visible in the line of text. With the SpaceMouse the user can move through the station and turn around. After turning around the exit is visible. After clicking on the block below the exit sign the viewpoint flies back through the line to the group box. Here also is an exit block to return to the PABX. In the PABX the user can navigate to the outside with the SpaceMouse.

6.4 Evaluation

After the design we let specialist of Human Factors and VEs of KPN Research look at the design to evaluate our environment. The results are presented in a list of remarks. At some remarks we made a remark, for example how it can be done with Superscape or why not. Next follows the list of remarks:

- It is preferable that the viewpoint zooms in on performance elements after clicking on them. The appliance of this remark has a negative side. Superscape has a limited number of 99 defined viewpoints. This number is not sufficient with all automated paths defined yet. Especially when the viewpoint has to move back after clicking a second time on the element, what would be a logically result.
- Change the outside colour of the PABX. On the map view from above, the PABX does not contrast with the underground.
- Change the colours representing internal and external calls in the performance objects. These colours conflict with the colour of the state of the stations and lines. Of course there is a limited number of colours in Superscape, and not all usable.
- It would be preferable if the performance of a day could be mapped on the performance of other days to see the difference or correspondence.
- The map-views are not very clear at indicating the position. There are two possible changes, also a combination of them is possible. First, the colour of the outside of the objects can change to the same colour, when the viewpoint is inside. Secondly, a 'You are here marker' could indicate where the viewpoint is. Such an object should be made moveable and connected to the viewpoint. It is not clear yet, if it is possible to make it controllable by the standard viewpoint of Superscape.

- The environment of the PABX, the sky and the grass, do not match the PABX. It is better to change the environment in a big grid plane. This also gives a better indication of depth.
- Do not make the speed of sound to fast. The user has to hear all details of it.
- The starting viewpoint can be closer to the PABX entry. The map view already indicates the construction of the whole environment. But more important, it indicates better that the user has to get inside the PABX.
- The lines of text may not contain important information that is not provide in the environment. Users do not read these lines all the time and they then miss important information.
- The checkerboard needs more explanation. This can be done with sound, with an index provided on the side and with a name on a sign. We tried to give the board an index. The index could not be placed on the border of the board. The texture was not readable at the necessary distance. A solution than can be to increase the size of the border, but then the proportions of the parts of the checkerboard do not match.
- The groups of three 3D bars should be changed. At least there should be created small distances between the bars, to get a clearer view. Better would be to create transparent tubes, where marbles can be put in. This is also clearer at a distance.
- On the wastebasket also should be placed a sign with a name. Also on the clocks. For all objects counts, if the viewpoint is close enough then the sign has to be visible. It also has to be big enough to make it readable.
- The wastebasket should also hold a sign indicating the number of telephones inside.
- There has to be better difference between spaces. The colours and furniture are not distinguishable enough for users to recognise the spaces.
- When people return to the block, after being inside a station, they do not know where they come. They do not recognise the room (see previous point).
- At some viewpoints, after travelling through a line, there is still visible a little part of the end of the line. This round entry can confuse people. They think that the room is round.
- The lines between the stations, blocks and PABX are so small, that the size of a person is to big to crawl through it. Therefore you should make clear that the user is a small object in the virtual world, like a fly. This can be said in the introduction sound, and the navigation viewpoint can be a controllable fly object. We tried to create a controllable object, but it was very hard to control. The opening doors can be confusing now, because a fly does not need such a big opening.
- Use the 3D space more. All performance objects and cable entries are put together on one wall. Also the rooms may be better recognisable then, when returning from a station. A disadvantage can be that users nee to use the SpaceMouse. The user needs a training period to get used to the SpaceMouse.

6.5 Design changes

In this section we describe the changes that are made after the design was finished. There are two kinds of changes made. First the changes made during the implementation, which can be the result of a shortcoming of the software or a shortcoming of the knowledge of the designer on the software. This latter is not impossible, because the software is very extensive and we had to learn working with it from the tutorials and manuals. The second kind of changes are the changes made after the evaluation that was described in the previous section. We first describe the changes made during the implementation, and after that the changes made on base of the remarks.

Changes during Implementation

In the design we described that we wanted to use collision detection. During the simulation it came clear that this was hard to achieve. Collision detection can only be achieved between objects. In Superscape controllable objects can be created. After creating a movable object, there can be add a control mode to the object and a viewpoint. In that viewpoint you make the object visible and with the control device the object can be controlled. It was very hard to control the object. We changed a lot of parameters, but that did not help. The speed still was increasing, and it was hard to stop the object. The object also did not react correctly to changes of directions. It could be a lack of our knowledge on Superscape, but the control did not work well. Another negative point is the collision detection. Every object has a bounding cube in which it is placed. This object has a flag to make it enterable or not. Because of the holes in objects, like cable entries and door entries, we could not use this flag. Another option is to create collision cuboids. These can be placed on the position of walls. If there is a whole in a wall, you have to create several collision cuboids to create a non-enterable wall with a hole in it. Because there are a lot of objects with holes in "Virtual SOA", this is very task intensive. Because of this intensive task and the bad controllable object we

decided not to use these techniques. Instead we increased the number of automated paths, so that users do not have to use the SpaceMouse often. But if they do not have to use the SpaceMouse often, the training period of the SpaceMouse will increase.

We also wanted the use a HMD with "Virtual SOA.". The design tool Superscape should support the HMD that was purchased for the project "Devine". Unfortunately, it did not work with our application. We followed the instruction from the HMD and Superscape, like correct resolution and console division, but without success. We could get the VE on the HMD display, but the resolution then is to small, there is no depth and the head tracking does not work. Another big disappointment.

Changes on base of remarks

On base of the remarks we made also some changes. Because of a lack of time (read deadlines), we could not make all changes. The changes we could make are:

- Changing the outside colour of the PABX to make a contrast with the green grass.
- Changing the colours for representing internal and external into light blue and dark blue. Internal will be light blue, external will be dark blue.
- In the map-view changing the outside colour of the objects, when the viewpoint is inside the object.
- Decrease the speed of the sound.
- The starting viewpoint is now closer to the PABX entry.
- No important information in the lines of text. if there is information, it is also visible in the VE.
- There is space between the 3D bar objects now.

We are aware that these changes are not sufficient to create the best possible VE, but that is something that can be taken in further research.

6.6 Conclusions

In this chapter we described the implementation of the demonstrator "Virtual SOA". The demonstrator is implemented with Superscape VRT. This tool is currently the best on Personal Computer platforms. Superscape is a very powerful and complex system that needs a long training time. But when you use other design programs to create realistic objects and import these in Superscape, you can create realistic worlds. It is clear that designing such worlds takes a lot of time.

In the evaluation it came clear that "Virtual SOA" has its shortcomings. This is particularly on the areas of visualisation of data in 3D and better support on letting the user create a mental map by making rooms recognisable. For further research it is recommendable to also examine the visualisation of (a lot of) data in 3D.

7 Conclusions & Recommendations

7.1 Introduction

In the last chapter of this report we present the overall conclusions and give recommendations. In the conclusions part we want to answer the research question. In the recommendations part we indicate what research we think still has to be done on this subject.

7.2 Conclusions

In this research we asked ourselves the question: "Does a Virtual Environment add value to remote management of PABXs?". We wanted to prove the added value of Virtual Environments to Network Management on the basis of a demonstrator that visualises a number of performance data of a PABX. Therefore this research is divided in the phases *definition*, *requirements* and *design and implementation*.

The goal of the definition phase was getting information on the subjects of Virtual Environments and Network Management. In Chapter 2 we first gave an introduction on Virtual Reality, followed by a description of VEs. Then we putted the light on the Human Factors side, the use of VEs. VEs as interface have the advantage that we are used to interact with 3D worlds. If the interaction with devices and VEs reflect the way we interact in the real world, we can perform tasks in VEs in a intuitive way. This will also be stimulated if users of VEs get immersed in VEs or at least feel presence. Immersion and presence also provide a better task performance, because users are more concentrated on their tasks. On the Human Factors side it is a goal to maximise the human performance efficiency and to minimise the negative health and safety consequences and social impact of VR. For the performance efficiency it is important that the VEs are appropriate for the kind of tasks you want to perform. Also still work has to be done to gain more knowledge about the human processes, especially the visual and auditory perception. On the subject of health and safety already work has been done, but this is only a start. A lot of research still has to be done to minimise the known side effects of VR applications. The social impact of VR applications is a subject that should be monitored. The result of the introduction of VR in the households can be that people stay at home and alienate from the outside world.

In Chapter 3 we gave an introduction on Network Management. We described the Private Automatic Branch eXchange and the Network Management system SOA. We also described the Performance Management of PABXs. The MAC analyst is chosen as user of "Virtual SOA". The MAC analyst has little technical knowledge and should be able to perform the tasks provided with "Virtual SOA". Another possible user of "Virtual SOA" is the buyer of a PABX. If the buyer of a PABX is able to manage his own PABX, without needing the technical knowledge, it would be a great completion on the product PABX. "Virtual SOA" demonstrates the Performance Management of PABXs. This functionality is not yet provided in SOA, but will be in the future. So Performance Management is an added functionality to the functionality of SOA.

The goal of the requirements engineering was to compose a list of guidelines as starting point for the design of Virtual Environments. We first presented the AIP cube of Zeltzer, which is very useful for the categorisation of and comparing of Virtual Environments. Virtual Environments can be mapped on the three axes autonomy, interaction and presence, all scaled between 0 and 1. The ultimate Virtual Environment is situated in the point (1,1,1). The presence axis is deepened by Peter Hofstede. He derived the Seven Presence Elements by studying and testing several Virtual Environment applications and provided guidelines to optimise these elements in Virtual Environments. These guidelines were supplemented in this research to get a more complete list of guidelines.

We presented the design and the implementation of "Virtual SOA" in respectively Chapter 5 and Chapter 6. First we designed the Virtual Environment, and completed the design with a technical and functional design, on the basis of the information and guidelines gathered in the phases definition and requirements. Following, the design is implemented with the design tool Superscape VRT and evaluated by specialist on Human Factors and Virtual Environments. In the evaluation it came clear that "Virtual SOA" has his shortcomings, particularly on the areas of visualisation of data in 3D and better support on letting the user create a mental map by making rooms recognisable.

At the start of this research we planned a usability test. In this test, MAC analysts perform tasks with both "Virtual SOA" and the current SOA application. The objective was to compare both cases. There are two reasons why these tests were not performed. First because the research took more time than expected, especially on the requirements engineering and on the implementation with Superscape. Secondly, the Performance Management is not provided in SOA, so the systems could not be compared on tasks like retrieving performance data. What could be tested is the usability of "Virtual SOA" with novice users. Unfortunately, there was no time for that.

To prove the added value of Virtual Environments to Network Management, on the basis of the developed demonstrator "Virtual SOA", it is necessary to perform a usability test. In this test, the old SOA application has to be compared with "Virtual SOA" on the task performance of users. The users in the test should be members of the target group of "Virtual SOA". In such a test, we measure the task perform time and retrieve information from the users with a questionnaire. In the previous paragraph, we described why we could not perform a test.

At the start of this research we had the expectation that Virtual Environments do have added value to Network Management. This expectation only has grown during this research. In "Virtual SOA" users can search very intuitively, as they are used to in real world situation. They can create faster a mental map than in the current SOA application and therefore faster find the necessary information. What should be taken in mind at applying Virtual Environments is that you first should check which tasks are suitable and which are not, to be implemented in 3D. Sometimes a 2D representation can be more clear than a 3D, or there is only added a third dimension to the same representation. In the case of "Virtual SOA" the task is suitable but not implemented good enough. The data representation should be more 3D.

7.3 Recommendations

At the end of this research, we like to make the following recommendations:

- Further research has to be done to the human processes of perception, especially the visual and auditory perception. These processes are not completely understood yet. If these processes are better understood, the interaction between user and Virtual Environment can be improved and a higher degree of presence and immersion can be achieved.
- Further research on devices is needed. The devices have to provide intuitive interaction between user and Virtual Environment. Not all devices provide this or can be improved to do so.
- Serious attention to the health and safety consequences and the social impact of the use of Virtual Environments should be given, in order to prevent injuries and social alienation of users.
- Research has to be done at the Network Management systems, that are currently provided with PABXs. Virtual Environment Management systems can be an added value to a PABX. Buyers can manage their PABXs without needing the technical knowledge.
- Still research has to be done to guidelines for the design of VEs. The generated list here can age / become obsolete because of improvement in technical or theoretical science.
- When you want to visualise (a lot of) 3D data in a Virtual Environment, also sufficient time must be spend on how to visualise these data.

8 References

[Abrahams] Abrahams, John R., *"Integrated PBX Systems: An NCC State-of-the-Art Report"*, Blackwell, Oxford, 1990.

[Airey] Airey, John M., Rohl, John H. and Brooks, Frederick P. Jr., "Towards image realism with interactive update rates in complex virtual building environments", *Interactive 3D* 1990, pages 41-50, 1990.

[Baker] Baker, M. Pauline & Wickens, Christopher D., *"Human factors in virtual environments for the visual analysis of scientific data"*.

[Balaguer] Francis Balaguer, et. al., *"Virtual Environments"*, ftp://ftp.u.washington.edu/public/virtual-worlds/papers/ .

[Bergh] Bergh, C.A.A. van den & Luijken, M.J.E, *"Devine: selectie van Virtual Environment authoring tool"*, KPN Research, R&D-RA-95-1103, november 1995.

[Black] Black, Uyless, *"Network Management Standards"*, McGraw-Hill, New York, 1992.

[Brooks] Brooks, Frederick P. Jr., "Grasping reality through illusion: Interactive graphics serving science", in *Proceedings of SIGCHI 1988*, pages 1-11, 1988.

[Choe] Norman Choe, *"No crash and burn here"*, http://www.xensei.com/users/normanc/www/paper.html, 1995.

[Computer Gr.] Computer Graphics, V26#3

[Cons] Consens, M., Hasan, M., "Supporting network management through declaratively specified data visualisations", *Proceedings of the third {1FIP/IEEE} International Symposium on Integrated Network Management*, 1993, 725-738.

[Darken] Darken, Rudy P. & Sibert, L., "A toolset for navigation in virtual environments", *proceedings of ACM User Interface Software & Technology*, 1993, pp. 157-165.

[Dorland] Dorland, W.A., *"Dorland's Illustrated Medical Dictionary"*, 28th ed., Philadelphia, Saunders, 1994.

[Durlach] Durlach, I. & Mavor, Anne S., *"Virtual Reality: scientific and technological challenges"*, National Academy Press, Washington, 1995.

[Esposito] Esposito, Chris, "Boeing Information Systems: User Interfaces for Virtual Reality Systems", *tutorial notes CHI'96*, 14 april 1996.

[Fisher] Fisher, Scott S., "Virtual environments: Personal simulations & telepresence", *Virtual Reality, theory, practice and promise*, p. 101 - 110, Meckler Publishing, Westport, 1991.

[Hofstede] Hofstede, Peter, *"Telepresence in shared virtual environments"*, October 1995, KPN Confidential, R&D-RA-95-899.

[InterCHI '93] *InterCHI '93 Conference Proceedings*, ACM Press/Addison Wesley , ISBN 0-201-58884-6.

[Isdale] Jerry Isdale, *"What is Virtual Reality? A homebrew introduction and information resource list"*, Version 2.1, 8 October 1993, ftp://ftp.u.washington.edu/public/virtual-worlds/papers/whatisvr.txt.

[Jacobson] Jacobson, Robert, "Beyond 'Hip', 'Hype' and 'Hope': The bigger picture of virtual worlds", *SIGGRAPH '90 Show Daily*, pages 6-8, August 1990.

[Laurel] Brenda Laurel, "VR will transform computers into extensions of our whole bodies", *Scientific American*, September 1995.

[Oberman] Oberman, M.R., *"PABX Almanak"*, Oberman Telecom Management Consultants, Blaricum, 1993.

[Pimentel] Pimentel, Ken & Teixeira, kevin, *"Virtual Reality. Through the new looking glass"*, Windcrest, New York, 1993.

[Psotka] Joseph Psotka, Ph.D. and Sharon Davison, *"Cognitive Factors Associated with Immersion in Virtual Environments"*

[Sala] Sala, Luc H.D.J & Barlow, John P., *"Virtual Reality: de metafysische kermisattractie"*, Verwey Mijdrecht, 1995.

[Sheridan] Sheridan, T.B., "Musings on telepresence and Virtual Presence", *PRESENCE, teleoperations and Virtual Environments*, Vol. 1 nr.1 pp 6, 120-126.

[Spaans] Spaans, E., *"Handboek netwerkbeheer"*, PTT Research Netwerktechnologie, Leidschendam, 1992, KPN internal report.

[Stanney] Stanney, Kay M., Ph.D., "Realizing the full potential of virtual reality: human factors issues that could stand in the way", in *Proceedings of the IEEE Virtual Reality Annual International Symposium*, IEEE Computer Society Press, NJ, 28-34.

[Viirre] Viirre, E., "A survey of medical issues and virtual reality technology", *Virtual Reality World*, August 1994, p. 16-24.

[Wantland] Cyril A.Wantland et al., *"Safety Considerations for Current and Future VR Applications"*

[Zeltzer] Zeltzer, David, "Autonomy, Interaction and Presence", *Presence: Teleoperators and Virtual Environments*, 1(1), January 1992, pp. 127-132.

Appendix A

```
Program CDRgen (input, output);

Const IntCall = 1;                (* constant InternalCall has value 1
*)
      ExtCall = 2;                (* constant ExternalCall has value 2
*)
      Incoming = 1;               (* constant IncomingCall has value 1
*)
      Outgoing = 2;               (* constant OutgoingCall has value 2
*)
      Grp = 1;                    (* constant Group has value 1
*)
      NoGrp = 2;                  (* constant NoGroup has value 2
*)

Type CDRType= Record
                Anr,                (* Caller        *)
                Bnr,                (* Receiver      *)
                ADate,              (* AnsweringDate *)
                ATime,              (* AnsweringTime *)
                ETime: LongInt;     (* ElapsedTime   *)
              End;
     CDRTree=^CDRTreeRec;
(* CDRTree is a binary tree of CDR's. CDR's will be sorted by
   AnsweringTime *)
     CDRTreeRec=Record
                 Left, Right,
                 Parent: CDRTree;
                 CDR: CDRType;
                End;

Var Infile, OutFile: Text;
    Tree: CDRTree;
    Period, GrpNoGrp,
    Number: Integer;

Function GetInt(Var i: Integer;S: String): Integer;
(* GetInt reads an integer from a string starting at position i. *)
Var x: Integer;
Begin
  x:=0;
  While (S[i] In ['0'..'9']) And (i<=Length(S))
  Do Begin
       x := 10*x+Ord(S[i])-Ord('0');
       i := i+1;
     End;
  GetInt := x;
  i:=i+1;
End; (* GetInt *)
```

```
Procedure ReadParams(Var Period, GrpNoGrp, Number: Integer);
(* ReadParams reads the parameters for the MakeRec procedure from the
*)
(* INI file 'cdrgen.ini'.
*)
Var S: String;
    i: Integer;
Begin
  Repeat
    Readln(InFile, S)
  Until (S[1] In ['0'..'9']) OR EOF(InFile);
  If Not (EOF(InFile))
  Then Begin
         i:=1;
         Period := GetInt(i, S);
         GrpNoGrp := GetInt(i, S);
         Number := GetInt(i, S);
         If Not (Period In [1..7])
         Then Writeln('Illegal period.')
         Else Begin
                If Not (GrpNoGrp In [1..2])
                Then Writeln('Illegal group.')
              End
       End
End; (* ReadParams *)

Procedure InitTree(Var Tree: CDRTree);
(* InitTree creates e new, empty tree of the CDRTree type. *)
Begin
  New(Tree);
  Tree:= Nil;
End; (* InitTree *)

Procedure InsertCDR(Var Tree, Parent: CDRTree; CDR: CDRType);
(* CDR's are recursively inserted. CDR's are ordered on AnsweringTime
*)
(* Tree is the tree on which the new node must be linked.
*)
(* Parent is the parent node of the node Tree.
*)
(* If Tree is Nil, then Parent is also Nil.
*)
Var Temp: CDRTree;
Begin
  If Tree=Nil
      (* Tree is a leaf on which the new node has to be linked.    *)
  Then Begin
         New(Temp);
         Temp^.CDR := CDR;
         Temp^.Left := Nil;
         Temp^.Right := Nil;
         If Parent <> Nil
         Then If Parent^.CDR.ATime < CDR.ATime
(* If AnsweringTime of Parent is smaller than AnsweringTime *)
              Then Parent^.Right := Temp
(* of new CDR, then insert CDR on the right side of Parent, *)
              Else Parent^.Left := Temp
(* else on the left side of Parent.                         *)
         Else Tree := Temp;
(* Parent is Nil, so total tree is empty and Tree := Parent *)
       End
  Else If Tree^.CDR.ATime < CDR.ATime
(* There is searched for the leaf on which the CDR has to be *)
       Then InsertCDR(Tree^.Right, Tree, CDR)
(* linked. Left and right are chosen on base of the          *)
       Else InsertCDR(Tree^.Left, Tree, CDR);
(* AnsweringTimes of the CDR and of the Tree node.           *)
End; (* InsertCDR *)
```

```
Procedure WriteOutFile(CDR: CDRType);
(* WriteOutFile writes the values of the records of a CDR to the output
file, separated by commas *)
Begin
  WriteLn(OutFile,
CDR.Anr,',',CDR.ADate,',',CDR.ATime,',',CDR.Bnr,',',CDR.ETime)
End; (* WriteOutFile *)

Procedure Inorder(Tree: CDRTree);
(* Inorder writes the inorder of the binary Tree to the output file.
The
   CDR's are ordered then in ascending order on AnsweringTime. *)
Begin
  If Tree <> Nil
  Then Begin
         Inorder(Tree^.Left);
         WriteOutFile(Tree^.CDR);
         Inorder(Tree^.Right);
       End
End; (* Inorder *)

Procedure InitCDR(Var CDR: CDRType);
(* Initialises the fields of CDR with values zero *)
Begin
  CDR.Anr := 0;
  CDR.ADate := 0;
  CDR.ATime := 0;
  CDR.Bnr := 0;
  CDR.ETime := 0;
End; (* InitCDR*)

Function IsGroupStation(CDR: CDRType): Boolean;
(* A value true is returned if the A-number or the B-number belongs to
   the group of stations. *)
Begin
  If (((CDR.Anr >= 11111) AND (CDR.Anr <= 11114)) OR (CDR.Anr = 11119))
OR
  (((CDR.Bnr >= 11111) AND (CDR.Bnr <= 11114)) OR (CDR.Bnr = 11119))
  Then IsGroupStation := True
  Else IsGroupStation := False;
End; (* IsGroupStation *)

Procedure SetANumber(Var Number:LongInt; GrpNoGrp, IntExt, IncOutg:
Integer);
(* SetANumber sets the A-number of the CDR based on the data if the
call is internal or external, *)
(* incoming or outgoing and if one of the numbers belongs to the group
of stations.                  *)
(* This is to filter out the combinations which are not selected. (see
the report, Section 5.5.1) *)
Var i: Integer;
Begin
  If (IntExt=IntCall)
  Then Begin
         If GrpNoGrp=Grp
         Then Number := 11110+random(4)+1
         Else Number := 11110+random(4)+5;
       End
  Else If IncOutg=Incoming
       Then For i := 1 To 8 Do Number := (Number*10)+random(9)+1
       Else Begin
              If GrpNoGrp=Grp
              Then Number := 11110+random(4)+1
              Else Number := 11110+random(4)+5;
            End
End; (* SetANumber *)
```

```
Procedure SetBNumber(Var Number:LongInt; GrpNoGrp, IntExt, IncOutg,
Except: Integer);
(* SetBNumber sets the B-number of the CDR based on the data if the
call is internal or external, *)
(* incoming or outgoing and if one of the numbers belongs to the group
of stations.                *)
(* This is to filter out the combinations which are not selected. (see
the report, Section 5.5.1) *)
(* The except parameter is the value of the A-Number in case of an
internal call. This is to      *)
(* avoid the possiblity that the A- and B-Number are equal.
*)
Var i: Integer;
Begin
  If (IntExt=IntCall)
  Then Begin
         If GrpNoGrp=Grp
         Then Number := 11110+random(4)+5
         Else Repeat
                Number := 11110+random(8)+1;
              Until Number <> Except
       End
  Else If IncOutg=Outgoing
       Then For i := 1 To 8 Do Number := (Number*10)+random(9)+1
       Else Begin
              If GrpNoGrp=Grp
              Then If random(40)=10
                   Then Number := 11110+random(4)+1
                   Else Number := 11119
              Else Number := 11110+random(4)+5;
            End
End; (* SetBNumber *)

Procedure SetDate(Var CDR: CDRType);
(* The date will always be set to 01-01-97. The simulation will cover
one day. *)
Begin
  CDR.ADate := 10197;
End; (* SetDate *)

Procedure SetATime(Var CDR: CDRType; Period: Integer);
(* The AnsweringTime will be set on base of the given period (in
seconds). *)
(* Period 1:  7:00 -  9:00                                          *)
(* Period 2:  9:00 - 11:00                                          *)
(* Period 3: 11:00 - 12:00                                          *)
(* Period 4: 12:00 - 14:00                                          *)
(* Period 5: 14:00 - 15:00                                          *)
(* Period 6: 15:00 - 17:00                                          *)
(* Period 7: 17:00 - 19:00                                          *)
Var h, m, s: LongInt;
Begin
  Case Period Of
    1:h := random(2)+7;
    2:h := random(2)+9;
    3:h := 11;
    4:h := random(2)+12;
    5:h := 14;
    6:h := random(2)+15;
    7:h := random(2)+17;
  End;
  m := random(60);
  s := random(60);
  CDR.ATime := (h*3600)+(m*60)+s;
End; (* SetATime *)
```

```
Procedure SetETime(Var CDR: CDRType);
(* Elapsed time depends on the A- and B-number. The calls made with or
to a station of the group of stations *)
(* is in general shorter than other calls.
*)
Var s: LongInt;
Begin
  If IsGroupStation(CDR)
  Then s := random(120)+30
  Else s := random(600)+30;
  CDR.ETime := s;
End; (* SetETime *)

Procedure MakeRec(Period, GrpNoGrp, aantal: Integer);
(* MakeRec creates a number of "aantal" CDR's based on the period in
which the call is made and if one of *)
(* the numbers belongs to a station of the group of stations.
*)
Var i, IntExt, Except, IncOutg: Integer;
    CDR: CDRType;
Begin
  For i := 0 To aantal-1
  Do Begin
       If (GrpNoGrp=Grp)
       Then If random(20)=10
            Then Begin
                   IntExt := IntCall;       (* The call is internal.*)
                   IncOutg := Outgoing;     (* The call is outgoing.*)
                 End
            Else If random(40)=10
                 Then Begin
                        IntExt := ExtCall;  (* The call is external.*)
                        IncOutg := Outgoing;(* The call is outgoing.*)
                      End
                 Else Begin
                        IntExt := ExtCall;  (* The call is external.*)
                        IncOutg := Incoming;(* The call is incoming.*)
                      End
       Else Begin
              If random(5)=3
              Then IntExt := IntCall        (* The call is internal.*)
              Else IntExt := ExtCall;       (* The call is external.*)
              IncOutg := random(2)+1;
                              (* The call is incoming/outgoing. *)
            End;
       InitCDR(CDR);
       SetANumber(CDR.Anr, GrpNoGrp, IntExt, IncOutg);
       SetBNumber(CDR.Bnr, GrpNoGrp, IntExt, IncOutg, CDR.Anr);
       SetDate(CDR);
       SetATime(CDR, Period);
       SetETime(CDR);
       InsertCDR(Tree, Tree, CDR);
     End
End; (*MakeRecs *)
```

```
(* Main *)
Begin
  Writeln('CDRGenerator Version 2.2');
  Writeln;
  Writeln('Initialising ...');
  Assign(InFile, 'cdrgen.ini');
  Reset(InFile);
  Assign(OutFile, 'cdrgen.dat');
                  (* Output will be written to "cdrgen.out".  *)
  Rewrite(OutFile);
  Randomize;
  InitTree(Tree);
  Writeln('Creating records ...');
  Repeat
    ReadParams(Period, GrpNoGrp, Number);
    MakeRec(Period, GrpNoGrp, Number);
  Until EOF(Infile);
  Inorder(Tree);
  Close(OutFile);
  Writeln;
  Writeln('Done');
End.
```

Appendix B

```
/*##########################################################################
                    SCL script of object RootObject
##########################################################################

==========================================================================
Procedure Max returns the maximum of two long variables x and y.
Parameters x (=p1) and y (=p2).
*/

proc (Max, 1, 2);
if (p1<p2)
  return (p2);
else
  return (p1);


/*==========================================================================
Procedure Min returns the minimum of two long variables x and y.
Parameters x (=p1) and y (=p2).
*/

proc (Min, 1, 2);
if (p1<p2)
  return (p1);
else
  return (p2);


/*==========================================================================
Procedure OpenCDR opens the input file 'cdrgen.dat' for reading.
Output is file-handle.
*/

proc (OpenCDR, 1, 0);
short  handle;

handle=fopen ("cdrgen.dat", "r");
if (handle<0)
  alert ("Input file 'cdrgen.dat' not found.", 0);
return (handle);


/*==========================================================================
Procedure ClsCDR closes the input file 'cdrgen.dat'.
Parameter p1 (=file-handle).
*/

proc (ClsCDR, 0, 1);
fclose (p1);
return;


/*==========================================================================
Procedure ReadNr reads a number out of the input file 'cdrgen.dat'.
Parameters p1 (=file-handle).
Output is EOF (=-1) or a number.
Ascii-numbers: 48 equals digit 0, 57 equals 9.
*/
```

```
proc (ReadNr, 1, 1);
long   a, ch;

ch=fgetc (p1);
a=0;
if (ch==-1)
{
  fputs ("End of file - cdrgen.dat.\n", 'PBXHold'.out);
  return (ch);
}
else
{
  do
  {
    a=10*a+ch-48;
    ch=fgetc (p1);
  }
  until (ch<48 || ch>57);
  return (a);
}


/*=============================================================================
Procedure SetIOB returns the IOB value (stands for Incoming/Outgoing/
IOBusy).
The IOB value can have the values 0 (=Incoming), 1 (=Outgoing) and
2 (=IOBusy).
Parameters p1 (=Anr), p2 (=Bnr) and p3 (=Station number).
*/

proc (SetIOB, 1, 3);
long   IOB, Busy=1;

if (p3==p1)
      /* Outgoing call */
  IOB=1;
else
{
  if (p3==p2)
        /* Incoming call */
    IOB=0;
  else
/* Should not be possible */
    IOB=-1;
}
switch (p3-11111);    /* Select Station */
case 0:
{
  if ('TelephoneHold'.Free==Busy)
    IOB=2;
}
case 1:
{
  if ('TelephoneHold[340]'.Free==Busy)
    IOB=2;
}
case 2:
{
  if ('TelephoneHold[462]'.Free==Busy)
    IOB=2;
}
case 3:
{
  if ('TelephoneHold[586]'.Free==Busy)
    IOB=2;
}
case 4:
{
```

```
  if ('TelephoneHold[898]'.Free==Busy)
    IOB=2;
}
case 5:
{
  if ('TelephoneHold[1056]'.Free==Busy)
    IOB=2;
}
case 6:
{
  if ('TelephoneHold[1176]'.Free==Busy)
    IOB=2;
}
case 7:
{
  if ('TelephoneHold[1296]'.Free==Busy)
    IOB=2;
}
else
/* Should not be possible */
  IOB=-1;
return (IOB);


/*=====================================================================
Procedure IntStat returns true (=-1) if the number is an internal
number, and false (=0) otherwise.
Parameter p1 (=Telephone number).
*/

proc (IntStat, 1, 1);
if (p1>=11111 && p1<=11118)
  return (true);
else
  return (false);


/*=====================================================================
Procedure IntCall returns true (=-1) if both numbers are internal
numbers,and false (=0) otherwise.
Parameters p1 and p2 (Telephone numbers).
*/

proc (IntCall, 1, 2);
if (IntStat (p1) && IntStat (p2))
  return (true);
else
  return (false);


/*=====================================================================
Procedure IsFree checks if the state of the station is free and returns
true (=-1) if free and false (=0) otherwise.
Parameter p1 (=Station number).
*/

proc (IsFree, 1, 1);
long   Free=0;

switch (p1%10);    /* Select station */
case 1:
{
  if ('TelephoneHold'.Free==Free)
    return (true);
  else
    return (false);
}
case 2:
{
```

```
  if ('TelephoneHold[340]'.Free==Free)
    return (true);
  else
    return (false);
}
case 3:
{
  if ('TelephoneHold[462]'.Free==Free)
    return (true);
  else
    return (false);
}
case 4:
{
  if ('TelephoneHold[586]'.Free==Free)
    return (true);
  else
    return (false);
}
case 5:
{
  if ('TelephoneHold[898]'.Free==Free)
    return (true);
  else
    return (false);
}
case 6:
{
  if ('TelephoneHold[1056]'.Free==Free)
    return (true);
  else
    return (false);
}
case 7:
{
  if ('TelephoneHold[1176]'.Free==Free)
    return (true);
  else
    return (false);
}
case 8:
{
  if ('TelephoneHold[1296]'.Free==Free)
    return (true);
  else
    return (false);
}
else
/* Should not be possible */
  return (false);

/*=====================================================================
Procedure GetFree returns the next free group station starting at last
station.
Parameter p1 (=Last).
*/

proc (GetFree, 1, 1);
long   i, Free=0;

i=p1;
do
  i= (i+1)%4;
until (IsFree (11111+i) || i==p1);
if (IsFree (11111+i))
  return (i);
else
  return (-1);
```

```
/*=============================================================================
Procedure Valid returns if ATime is valid.
This because of errors during sending messages which could not be
detected. Probably because the messages stack is full at the time of
sending. ATime is valid between 7am (25200 seconds) and 7pm
(68400 seconds).
*/

proc (Valid, 1, 1);
if (p1>=25200 && p1<=68400)
  return (true);
else
  return (false);


/*############################################################################
                    End of SCL script of object RootObject
############################################################################*/



/*############################################################################
                    SCL script of object PBXHold
############################################################################*/

short  out, CDR, Internal=1, External=0, Incoming=0, Outgoing=1,
    Busy=2, res;
long   Anr, Date, ATime, Bnr, ETime, CTime=-1, WTime, TmSpeed=1,
    OldTime, geluid;
objnum StatObj[8];

/* Set resume */
resume (1, 2);

/* Initialise StatObj */
StatObj[0]='TelephoneHold';
StatObj[1]='TelephoneHold[340]';
StatObj[2]='TelephoneHold[462]';
StatObj[3]='TelephoneHold[586]';
StatObj[4]='TelephoneHold[898]';
StatObj[5]='TelephoneHold[1056]';
StatObj[6]='TelephoneHold[1176]';
StatObj[7]='TelephoneHold[1296]';

/* Assign output file */
out=fopen ("virsoa.log", "w");
if (out<0)
  alert ("Log file not created.", 0);
fputs ("Log file from Virtual SOA\n\n", out);

/* Write messages to the screen. */
instr (1)="Welkom bij Virtual SOA.";
instr (3)="Klik op de deuren om naar de PABX te vliegen.";

/*Start intro sound. */
OldTime=vrtime;
geluid=sound (1, 74, 127, 0);
while (vrtime-OldTime<26000)
{
  if (sound? (geluid)==0)
    geluid=sound (1, 54, 127, 0);
  waitf;
}

/* Read data of input file */
CDR=OpenCDR;
if (CDR>=0)
{
```

```
  do
  {
/* ATime and ETime are in seconds. */
    Anr=ReadNr (CDR);
    Date=ReadNr (CDR);
    ATime=ReadNr (CDR);
    Bnr=ReadNr (CDR);
    ETime=ReadNr (CDR);
    if (Anr>=0)
    {
/* Anr is less than zero when EOF(cdrgen.dat) *//* Wait until CTime
(current time) is equal to ATime.*/
      if (ATime>=CTime)
      {
        if (CTime==-1)
          WTime=0;
        else
          WTime=ATime-CTime;
      }
      else
        WTime=24*3600+ATime-CTime;
      waitfs (WTime/TmSpeed);

/* Set CTime. CTime is a variable containing the 'current time' of the
   simulation.
*/
      CTime=ATime;

/* Send message to station. */
      if (IntCall (Anr, Bnr))
      {
/* The call is internal. If the receiver (Bnr) is Free, the call can be
   made, else a message to the receiver is send that he was Busy.
*/
        if (IsFree (Bnr))
        {
          do
          {
            res=sendmsg (ATime, StatObj[Anr-11111]);
            waitf;
          }
          until (res==0);
          do
          {
            res=sendmsg (ETime*100+Internal*10+Outgoing,
                StatObj[Anr-11111]);
            waitf;
          }
          until (res==0);
          do
          {
            res=sendmsg (ATime, StatObj[Bnr-11111]);
            waitf;
          }
          until (res==0);
          do
          {
            res=sendmsg (ETime*100+Internal*10+Incoming,
                StatObj[Bnr-11111]);
            waitf;
          }
          until (res==0);
        }
        else
        {
          do
          {
            res=sendmsg (Busy, StatObj[Bnr-11111]);
            waitf;
```

```
          }
          until (res==0);
        }
      }
      else
      {
/* The call is external. One of the numbers, either Anr or Bnr, is
   internal.
*/
        if (IntStat (Anr))
        {
          do
          {
            res=sendmsg (ATime, StatObj[Anr-11111]);
            waitf;
          }
          until (res==0);
          do
          {
            res=sendmsg (ETime*100+External*10+SetIOB (Anr, Bnr,
                Anr), StatObj[Anr-11111]);
            waitf;
          }
          until (res==0);
        }
        else
        {
          if (IntStat (Bnr))
          {
            do
            {
              res=sendmsg (ATime, StatObj[Bnr-11111]);
              waitf;
            }
            until (res==0);
            do
            {
              res=sendmsg (ETime*100+External*10+SetIOB (Anr,
                  Bnr, Bnr), StatObj[Bnr-11111]);
              waitf;
            }
            until (res==0);
          }
          else
          {
            if (Bnr==11119)
            {
              do
              {
                res=sendmsg (ATime, 'BoxGroup');
                waitf;
              }
              until (res==0);
              do
              {
                res=sendmsg (ETime*100+External*10+Incoming,
                    'BoxGroup');
                waitf;
              }
              until (res==0);
            }
          }
        }
      }
    }
  }
  until (Anr<0);
}
ClsCDR (CDR);
```

```
fclose (out);
stopc (me);

/*##############################################################################
                    End of SCL script of object PBXHold
##############################################################################*/



/*##############################################################################
                    SCL script of object BoxGroup
##############################################################################*/

long   Free=0, Counter, Sender, Stopped=4, NrWaits,
    MinTime=1000000000, MaxTime, AveTime, FirstPos, LastPos,
    Anr, ATime, ETime, IOB, NrFrames=0, CNrWaits, ATimeA[1000],
    ETimeA[1000], WTimeA[1000], HangUp, Last, External,
    Incoming, h=-500, i, t, y, z;
objnum Waiter, WaitQ[1000], StatObj[4];
short  x, res;

/* Set resume */
resume (2, 1);

/* Initialise variables */
StatObj[0]='TelephoneHold';
StatObj[1]='TelephoneHold[340]';
StatObj[2]='TelephoneHold[462]';
StatObj[3]='TelephoneHold[586]';

/* NrFrames counts the number of times the script is called */
++NrFrames;

/* Is there a Free station and a waiting call? */
if (Counter!=4 && FirstPos!=LastPos)
{
/* WaitQueue not empty.
   One call from the WaitQueue is send to Free station.
   Stop SCL script of WaitQueueTelephone and delete WaitQueueTelephone.
*/
  stopc (WaitQ[FirstPos]);
  delete (WaitQ[FirstPos]);

/* NrWaits equals total nr. of waiters that have been passed to Free
   stations.
   CNrWaits equals the current number of waiters in the WaitingQueue.
*/
  ++NrWaits;
  --CNrWaits;

/* Send data of call to Free station */
  z=GetFree (Last);
  if (z!=-1)
  {
    Last=z;
    do
    {
      res=sendmsg (ATimeA[FirstPos], StatObj[z]);
      waitf;
    }
    until (res==0);
    do
    {
      res=sendmsg (ETimeA[FirstPos]*100+External*10+Incoming,
          StatObj[z]);
      waitf;
    }
    until (res==0);
```

```
  }
  else
    alert ("No Free station available.", 0);
/* WaitingTime t is nr. of frames the call waited * TmSpeed
   (= 1 frame equals TmSpeed seconds).
*/
  t= (NrFrames-WTimeA[FirstPos])*'PBXHold'.TmSpeed;
  MinTime=Min (MinTime, t);
  do
  {
    res=sendmsg (MinTime, 'ClockMin');
    waitf;
  }
  until (res==0);
  MaxTime=Max (MaxTime, t);
  do
  {
    res=sendmsg (MaxTime, 'ClockMax');
    waitf;
  }
  until (res==0);
  AveTime= (AveTime* (NrWaits-1)+t)/NrWaits;
  do
  {
    res=sendmsg (AveTime, 'ClockAverage');
    waitf;
  }
  until (res==0);
  FirstPos= (FirstPos+1)%1000;
  ++Counter;

/* Waiting for a message */
}
else
{
  if (msgrdy?)
  {
    y=waitmsg (&Sender);
    if (y==Stopped)
    {

/* Waiting call hanged up.
   Put WaitQueueTelephone in WasteBasket.
   Stop SCL script of WaitQueueTelephone
*/
      stopc (object (Sender));

/* Delete object from WaitQ.
   First seek Sender in WaitQ, then shuffle the other waiters.
   Also think of ATimeA, ETimeA, WTimeA.
*/
      i=FirstPos-1;
      do
        i= (i+1)%1000;
      until (WaitQ[i]==object (Sender));
      if (i==FirstPos)
        FirstPos= (FirstPos+1)%1000;
      else
      {
        do
        {
          WaitQ[i]=WaitQ[ (i+1)%1000];
          ATimeA[i]=ATimeA[ (i+1)%1000];
          ETimeA[i]=ETimeA[ (i+1)%1000];
          WTimeA[i]=WTimeA[ (i+1)%1000];
          i= (i+1)%1000;
        }
        until (i==LastPos);
        LastPos= (LastPos-1+1000)%1000;
```

```
      }

/* Move object to WasteBasket */
      xpos (object (Sender))=16000+random (3000);
      h+=550;
      ypos (object (Sender))=h;
      zpos (object (Sender))=9500+random (1000);
      --CNrWaits;
      ++HangUp;
      do
      {
        res=sendmsg (HangUp, 'WasteBasketGroup');
        waitf;
      }
      until (res==0);
    }
    else
    {
/* New call received from the PABX.*/
      ATime=y;
      y=waitmsg (&Sender);
      ETime=y/100;
      IOB=y%10;

/* Handle the message.
   If Counter equals four then all four stations are busy and the call
   is put in the WaitingQueue.
*/
      if (Counter==4)
      {
        Waiter=dup ('WaitQueueTelephone');
        if (Waiter=='RootObject')
          alert ("Duplication not established.", 0);
        else
        {
          x=adopt ('BoxInsideHold', Waiter);
          if (x!=0)
          {
            alert ("Adoption not established.", 0);
            stopc (Waiter);
            delete (Waiter);
          }
          else
          {
            xpos (Waiter)=15000+random (4000);
            ypos (Waiter)=5000+random (10000);
            zpos (Waiter)=7500+random (5000);
            vis (Waiter);

/* Insert Waiter in the WaitQ.
   If FirstPos==LastPos, the WaitQ is empty.
   If FirstPos==LastPos+1 mod 1000, the WaitQ is full.
*/
            if (FirstPos!= (LastPos+1)%1000)
            {
              WaitQ[LastPos]=Waiter;
              ATimeA[LastPos]=ATime;
              ETimeA[LastPos]=ETime;
              WTimeA[LastPos]=NrFrames;
              LastPos= (LastPos+1)%1000;
              do
              {
                res=sendmsg (random (420)+180, Waiter);
                waitf;
              }
              until (res==0);
              ++CNrWaits;
            }
            else
```

```
            {
              delete (Waiter);
              alert ("Maximum number of Waiters achieved.", 0);
            }
          }
        }
      }
      else
      {
/* Send message to a Free station. */
        z=GetFree (Last);
        if (z!=-1)
        {
          Last=z;
          do
          {
            res=sendmsg (ATime, StatObj[z]);
            waitf;
          }
          until (res==0);
          do
          {
            res=sendmsg (ETime*100+External*10+Incoming,
                StatObj[z]);
            waitf;
          }
          until (res==0);
          ++Counter;
        }
      }
    }

/*##############################################################################
                    End of SCL script of object BoxGroup
##############################################################################*/
  }
}



/*##############################################################################
                    SCL script of object TelephoneHold[2]
##############################################################################*/

long   Free=0, Busy=1, ATime, ETime, Internal, IOB, x, Sender;
short  res;

/* Set resume */
resume (2, 1);

/* Wait for a message. Message contains <ETime><ATime><Internal><IOB>.
*/
ATime=waitmsg (&Sender);
x=waitmsg (&Sender);
if (Valid (ATime))
{
  ETime=x/100;
  Internal=x%100/10;
  IOB=x%10;

/* Send NumberOfCallsHold message. Message contains <Internal><IOB>. */
  do
  {
    res=sendmsg (Internal*10+IOB, 'NumberOfCallsHold');
    waitf;
  }
  until (res==0);
  do
```

```
  {
    res=sendmsg (Internal*10+IOB, 'NumberOfCallsHold[24]');
    waitf;
  }
  until (res==0);
  do
  {
    res=sendmsg (Internal*10+IOB, 'NumberOfCallsHold[277]');
    waitf;
  }
  until (res==0);
  if (IOB!=2)
  {
/* Send CheckerBoardHold message. Message contains <hours><Internal>.
   hours is equal to ATime div 3600. */
    do
    {
      res=sendmsg (ATime/3600*10+Internal, 'CheckerBoardHold');
      waitf;
    }
    until (res==0);
    do
    {
      res=sendmsg (ATime/3600*10+Internal,
          'CheckerBoardHold[169]');
      waitf;
    }
    until (res==0);
    do
    {
      res=sendmsg (ATime/3600*10+Internal,
          'CheckerBoardHold[289]');
      waitf;
    }
    until (res==0);

/* State becomes Busy. Add 1 to Counter's of TelephoneCable's. */
    if (object (Sender)=='PBXHold')
      ++'BoxGroup'.Counter;
    ++'TelCableHold[22]'.Counter;
    ++'TelCableHold[1035]'.Counter;

/* Waiting until call is finished. (ETime has passed) */
    waitfs (ETime/'PBXHold'.TmSpeed);

/* State becomes Free. Substract 1 of Counter's of TelephoneCable's and
   BoxGroup. */
    --'BoxGroup'.Counter;
    --'TelCableHold[22]'.Counter;
    --'TelCableHold[1035]'.Counter;
/*##########################################################################
                    End of SCL script of object TelephoneHold[2]
##########################################################################*/
  }
}



/*##########################################################################
                  SCL script of object WaitQueueTelephone
##########################################################################*/

long   ETime, Sender, Stopped=4;
short  res;

/* Set resume
*/
resume (2, 1);
```

```
/* Waiting for a message
*/
ETime=waitmsg (&Sender);

/* Handle message
*/
waitfs (ETime/'PBXHold'.TmSpeed);
do
{
  res=sendmsg (Stopped, 'BoxGroup');
  waitf;
}
until (res==0);

/*##############################################################################
                    , End of SCL script of object WaitQueueTelephone
##############################################################################*/
```