DEPARTMENT OF COMPUTER SCIENCE
AALBORG UNIVERSITY

TITLE:	GCVRT - Merging the Collaborative Virtual Reality and Round Table Concepts
THEME:	Computer Vision & Virtual Reality
PROJECT TERM:	DAT6, February 2002 - June 2002
PROJECT GROUP:	N4-307a

PARTICIPANTS:
Flemming R. Jønsson
`joensson@cs.auc.dk`
Jacob K. Uhrenholt
`kreel@cs.auc.dk`
Jens Peter Vester
`vester@cs.auc.dk`

SUPERVISORS:
Erik Granum
`eg@cvmt.auc.dk`
Henrik Rojas Nagel
`hrn@cvmt.auc.dk`
Peter Bøgh Andersen
`pba@cs.auc.dk`

Abstract

Collaborative work can be performed in many ways. One way is to have a meeting with the people involved. However, this way of collaborating requires people to be physically present at the same time in the same place. This report proposes a scheme for transcending this constraint by introducing a way of performing and enhancing types of collaborative work, which take place at a meeting table and can be described by manipulating objects in a 3D world. It also enables people who are not present to participate in the collaborative work on the same basic premises as those who are.
The problem is analyzed using CSCW frameworks based on coordination mechanisms and activity theory. The result of the analysis is that a system, which provides a persistent but passive environment describing the field of work and is mainly state oriented, can support collaboration on a co-constructive level.
On the basis of the analysis, a design based on the GCVR system is made, which supports a combined short and long distance approach towards collaboration. Also, the design allows users to interact using common physical objects as input devices by the means of a vision tracking system. The current state of the implementation indicates that such a system can be constructed in practice and that it will be possible to improve the collaborative work in both a local as well as a remote setting.

NUMBER PRINTED:	10
NUMBER OF PAGES:	130
FINISHED:	June 14th 2002.

INSTITUT FOR DATALOGI
AALBORG UNIVERSITET

TITEL:	GCVRT - Merging the Collaborative Virtual Reality and Round Table Concepts
TEMA:	Computer Vision & Virtual Reality
PROJEKTPERIODE:	DAT6, Februar 2002 - Juni 2002
PROJEKTGRUPPE:	N4-307a

DELTAGERE:
Flemming R. Jønsson
`joensson@cs.auc.dk`
Jacob K. Uhrenholt
`kreel@cs.auc.dk`
Jens Peter Vester
`vester@cs.auc.dk`

VEJLEDERE:
Erik Granum
`eg@cvmt.auc.dk`
Henrik Rojas Nagel
`hrn@cvmt.auc.dk`
Peter Bøgh Andersen
`pba@cs.auc.dk`

Synopsis

Samarbejde kan udføres på mange måder. En af disse måder er at arrangere et møde mellem de involverede parter, men denne type samarbejde kræver at parterne er fysisk til stede på samme tid og sted. Denne rapport foreslår en måde at omgå denne begrænsning på ved at introducere en metode, hvorpå møder omkring et bord, som kan beskrives ved at manipulere objekter i en 3D verden, kan forbedres. Denne metode giver også parter, der ikke er til stede omkring bordet, mulighed for at deltage i samarbejdet på de samme basale præmisser, som dem der er.
Problemet analyseres ved at bruge CSCW-metoder baseret på koordineringsmekanismer og aktivitetsteori. Resultatet af analysen er et system, der stiller et blivende, passivt miljø, der beskriver arbejdsområdet og som hovedsageligt er tilstandsorienteret og som er i stand til at understøtte samarbejde på et co-konstruktivt niveau, til rådighed.
På basis af analysen præsenteres et design baseret på GCVR-systemet, som understøtter samarbejde både lokalt og over afstand. Designet tillader desuden brugerne at interagere med systemet ved hjælp af almindelige hverdagsobjekter vha. et computer vision baseret system, der oplyser objekternes position. Den nuværende tilstand af systemimplementationen viser, at systemet kan konstrueres i praksis og at det vil være muligt at forbedre eller understøtte samarbejde i både lokal sammenhæng og over afstand.

ANTAL EKSEMPLARER:	10
SIDEANTAL:	130
AFSLUTTET:	14. juni 2002.

PREFACE

This report serves as documentation of the work done by the project group during the first term of 2002. The project is a continuation of the Generic Collaborative Virtual Reality (GCVR) project (described in [JUV02]), which took its origin in the project proposal "Virtual Environment for Distributed Collaboration" proposed by Erik Granum, Henrik Rojas Nagel and Peter Bøgh Andersen.

The report is divided into three major parts: Analysis, design and finally conclusion. The third part explains the status of the implementation, a specification of how to test it, suggestions for further work on the project and finally the conclusion of the report.

Throughout the report, specific standards are not necessarily kept in the figures. However, when describing class hierarchies, the UML standard will be kept.

When referencing an external source of information, the references will keep the following format: [JUV02]. The exact reference can be seen in the bibliography on page 124. In order to improve the readability of the report, function names are emphasized like this: `someFunction`. Class names are written like this: SomeClass.

During the project, a number of people have contributed. Especially, we would like to thank:

- Moritz Störring at the CVMT research group at Aalborg University for helping the project group to use the vision tracking system, which he is involved in the development of.

- Per Nielsen, president at and co-founder of EMD for evaluating features proposed by the project group and allowing us to analyze the working habits in his company.

Without the contributions of these people, vital parts of the project could not have been completed.

Flemming R. Jønsson   Jens Peter Vester

Jacob K. Uhrenholt

CONTENTS

1 Introduction **1**
1.1 Scenarios of Application 2
1.2 The Report 3

I Analysis **5**

2 Existing Technologies **7**
2.1 Augmented and Virtual Reality 7
2.2 Types of Reality 9
2.2.1 GCVR in the Reality-Virtuality Continuum 11

3 Application Context **13**
3.1 EMD 13
3.1.1 EMD’s work process 13
3.1.2 Collaboration 15

4 Coordination Mechanisms **17**
4.1 Pragmatic Dimensions 18
4.1.1 Persistent or Non-persistent 18
4.1.2 Passive or Active 19
4.2 Semantic Dimensions 19
4.2.1 Process or State 19
4.2.2 Field of Work or Work Arrangement 20
4.3 Discussion of the Virtual Round Table Environment 20
4.4 Discussion of the GCVR system 22
4.5 Discussion of EMD 23
4.5.1 The Artifacts 24
4.6 Comparison of the Approaches to Cooperation 27
4.6.1 Combining the VRT with the GCVR system 27
4.7 Discussion of the Process/State Continuum 29
4.8 Relating the GCVR System to the Working Habits of EMD 30

5 Collaboration and Activity Theory **33**
5.1 Supporting Transitions 36
5.2 The Application 38
5.2.1 Collaborative Work 38
5.2.2 Comparing Theory and Practice 39

5.2.3 Design Implications 42

6 Feature Discussion **45**

6.1 Merging the Paradigms 45

6.1.1 Basic Design Tasks 47

6.2 Advanced Features 48

6.2.1 Movement in the Virtual Environment 48

6.2.2 Interaction in the Virtual Environment 49

6.2.3 Deity and Mortal Mode 50

6.2.4 Environment Features 51

6.2.5 Active Layer 53

6.2.6 Speech and 3D Audio Effects 54

6.3 EMD Feedback 56

7 Requirement Specification **61**

7.1 Supporting the Scenarios of Use 61

7.1.1 Supporting the Round Table Scenario 61

7.1.2 Supporting the Multi-user Scenario 62

7.1.3 Supporting the Single-user Scenario 63

7.1.4 Supporting EMD 63

7.2 Software Requirements 63

7.3 Hardware Requirements 64

7.4 Final Classification 65

II System Design **67**

8 Design Approach **69**

8.1 The Design of the Original GCVR System 69

8.1.1 The Original GCVR Server 71

8.1.2 The Original GCVR Client 71

9 System Structure **73**

9.1 Structural Premises 73

9.1.1 Possible Solutions 75

9.2 System Overview 78

10 Tracking Systems **81**

10.1 Vision Tracking 81

10.1.1 Filtering 83

10.2 Head Tracking 84

11 GCVRT Client Functionality 85
11.1 Mapping PHO States to VO States 86
11.2 Utilizing Head Tracking Information 88
11.3 Receiving Head and Vision Tracking Information 88

12 Auxiliary Features 89
12.1 Moving in Different Coordinate Systems 89
12.2 3D Landscape Generator 90
12.3 The Recording Module 95
12.4 File Distribution 97

13 Assembling the Server and Client Components 99
13.1 Assembling the Server 99
13.2 Assembling the Client 100

III Test, Status and Conclusion 103

14 Testing Specification for the GCVRT System 105
14.1 Functional Testing 106
14.2 Unit, Performance and Stress Testing 108
14.3 Parallel Testing 112
14.4 Final Remarks 112

15 Status of the Implementation 113
15.1 Current Status 114

16 Future Work 117

17 Conclusion 121

Bibliography 124

IV Appendix 125

A Speak Freely Codecs 127

B The Intersense 129

LIST OF FIGURES

2.1 Milgram's Virtuality Continuum. 10
3.1 EMD photo montage 15
4.1 Screen shots from the WindPro application. 26
5.1 A graphical display of the three activity theory concepts. 35
6.1 A possible system structure. 46
6.2 The PHO interaction method. 47
8.1 Overall System Structure 70
8.2 UML diagram of the original GCVR server. 71
8.3 UML diagram of the original GCVR client. 72
9.1 The GCVR/RT system structure using a synchronization approach. 75
9.2 Event flow for a fast and a slow computer performing a mapping calculation. 76
9.3 The GCVR/RT system structure using a dedicated master. 77
9.4 The GCVR/RT system structure. 77
9.5 The GCVRT system structure. 78
10.1 The initializing message of the vision system. 82
10.2 The state message of the vision system. 83
10.3 The head tracker message format. 84
11.1 The format of the records containing the mapping of PHOs to virtual objects. 86
11.2 UML diagram of the communication module of the masterclient. . 88
12.1 The method of displacement maps. 92
12.2 The functionality of the GenerateDEMTerrain. 94
12.3 UML diagram of the GenerateDEMTerrain class. 95
12.4 UML diagram of the Log class. 97
13.1 UML diagram of the assembled server component. 99
13.2 UML diagram of the assembled client component. 101

List of Tables

4.1 The properties of the Virtual Round Table. 22
4.2 The properties of the GCVR system. 23
4.3 The properties of EMD's photo-mockups. 25
4.4 The properties of WindPRO. 27
4.5 The combined properties of the VRT and the GCVR system. 27
4.6 The combined properties of the VRT and the GCVR system extended with a logging or recording system and active application-specific features. 29
6.1 Comparison of the Roger Wilco and Speak Freely codecs 56
7.1 The properties of the GCVRT system. 65

1 INTRODUCTION

Today, systems providing interpersonal communication are mainly focused on long-distance communication. Examples are email, telephone, collaborative virtual reality systems etc. That is, systems which enable users to collaborate despite the fact that they may be located far away from each other.

Short-distance communication, however, has not yet received the same attention because human beings are equipped with means for performing this task effectively. Nevertheless, some collaborative tasks involving short-distance interpersonal communication can sometimes be improved.

Imagine a situation where a group of people, say five, need to make an agreement concerning the placement of a number of buildings at a new city plaza. They would sit around a table and see a perspectively correct three dimensional visualization of the plaza in the center of it. The city planners would be able to visualize three-dimensional models of the buildings which should be placed at the plaza and interactively change the position and other attributes of the buildings. In order to easily and intuitively interact with the sub-models of the visualization, the city planners would place a physical object at the position on the table where they want the virtual objects to be placed. The software would then visually track the physical object and map it with the virtual object. In that way, when a physical object is moved, the virtual object follows it.

Imagine a somewhat similar example where two groups of city planners, sitting at two tables located in different buildings, cities or countries. The two groups would be able to see the same visualization and interact with it in the same way as if they were sitting at the same table.

Now imagine that several groups of city planners and some people living near the plaza were going to reach an agreement on this matter. The groups of city planners would be able to sit at tables and see the visualizations. The remaining people would be able to sit at home at their PCs and via their network connections be able to see a representation of the virtual world. Furthermore, they would be able to enlarge the virtual world and move through it just about the same way as they would, if the current plan had been carried out. These people could also very well be invited to take part in the discussion by placing them in a panorama arena or allowing them to experience the plaza from a CAVE.

For these scenarios, two modes of interaction would be interesting (both concepts are described in [LJDV96]):

- Deity mode: The city planners would be very interested in watching the city plan from a schematic point of view and they would need to be able to move and make other changes to the three dimensional models.
- Mortal mode: The users of the city plaza would mainly be interested in experiencing the plaza, as they would if it had already been built. They would, however, not be allowed to move the structures in the scene.

We cannot take credit for all of these ideas. The Virtual Round Table project (VRT) described in [BMS00] has already proposed the short-distance communication part of the described scenarios. What we propose, however, is a combined short- and long-distance interpersonal communication system allowing a variety of arenas to be available. We have already described a design which allows this: The Generic Collaborative Virtual Reality system (GCVR) (described in [JUV02]). The challenge which we will take up in this project is by the means of GCVR to create a system which enables all of the above scenarios to be carried out.

1.1 Scenarios of Application

In this section we will describe some scenarios, which are possible in a system such as the one described in the introduction. We will base this system on the GCVR system, which we have developed and which provides a wide range of basic features which are needed in a collaborative virtual reality system. We will not list the features of this project here but merely refer to the GCVR report (see [JUV02]) for an in-depth analysis, design and discussion. The system, which we will describe in the present report, will therefore include all of the features described in the GCVR report. We will now focus on some basic features which separate this project from the GCVR project.

One way of making a user see a perspectively correct 3D visualization is to equip the users with partially transparent head-mounted stereo displays (HMDs). Another way is to place them in CAVEs or in front of a panorama or a standard PC monitor and equip them with shutter glasses. What we need is a combination of these possibilities in order to enable users to collaborate as independently as possible of their specific arena[1].

At this point we propose three basic scenarios for using the system which could be combined in any way:

- The round table: The users sit around a table and see a visualization on the table between them. This visualization can be made by the means of partially transparent HMDs. That is, when there are no objects in the virtual world or a user does not look at an object, the HMDs are fully transparent. When the user looks at an object in the virtual world, the object appears before him, but all other parts of the HMDs are transparent.

- Single-user: One user per arena interacts with the system. The arena could be a standard PC or a CAVE. The user will see only objects which are loaded into the virtual world.

- Multi-user: Many users are able to watch a session of collaborative work from one common viewpoint in the same way as a single user. In this case the arena will be a panorama. Interaction will be available for one user at a time.

The system must support the interaction possibilities defined in the GCVR project (movement and interaction by the means of mouse, keyboard and wanda). Also, it must be possible to move physical objects and map their positions into the virtual world. A vision tracking system providing the necessary data is already under development at the Computer Vision and Media Technology research group (CVMT)

[1]We will use the term arena for the combination of input and output devices which determines a user's interaction possibilities - for instance a PC with a keyboard, a mouse and a standard PC monitor.

at Aalborg University and we will integrate a development version of this system with our own. A variation of the round table scenario would be to use physical objects to manipulate the geometric patterns of 3D models, but show the result on a computer screen or using a projector. This variation may be useful in situations where HMDs and various tracking devices are not options.

1.2 The Report

In the next part of the report, we will analyze the area of collaboration in the context of the scenarios and problems stated in section 1.1. We will introduce the classification framework of coordination mechanisms and a theory of computer supported cooperative work (CSCW) based on activity theory in order to determine the exact requirements for making a system capable of supporting the scenarios mentioned. Also, a company (EMD), which is involved in the development of wind farms will be consulted in order to ensure consistency between the analysis and actual user requirements. The analysis will be followed by a design, which describes how a system supporting the requirements can be constructed. Finally, parts of the system which prove the viability of the design will be implemented and a test will be specified.

Part I

Problem Domain Analysis and Functional Requirements

In the analysis existing technologies are investigated after which the classification method of coordination mechanisms is introduced. The method is then applied to three scenarios (round table, single- and multi-user) and it is described how the systems differ in some places and how they are similar in others. This results in the conclusion that the three paradigms can be combined and a new system takes form. A discussion of possible features in such a system follows. Then a selection of essential features is made on the basis of meetings with a company, which is involved in designing wind farm layouts. Finally, the requirements for the new combined system will be identified and a classification of this system using the described classification method concludes the analysis.

2 Existing Technologies

Since this project in part builds upon the conceptual framework of the Virtual Round Table (VRT) project (See [BMS00]), a project concerned with the development of an augmented reality (AR) workspace, we will in this chapter introduce the state-of-the-art for related systems. The covered topics fall into the area of virtual reality (VR) and AR in a distributed collaborative setting.

2.1 Augmented and Virtual Reality

- **Virtual Round Table**
  The VRT system was developed as a proof of concept that semi-transparent head mounted displays and place holder objects (PHO) could be utilized for creating a new way of interacting with a virtual world. The concept of PHOs was about assigning a virtual object to a physical object, thus forming what they define as an interaction unit. By moving the PHO the virtual object is affected by a corresponding movement in the virtual environment.

- **ARTHUR**
  ARTHUR is strictly speaking not an existing technology as it is still under development. It is, however mentioned here because it uses a vision tracking system from which input data can be obtained.

  The ARTHUR project has been developed with a local collaborative work setting in mind. It is thought of as a tool for architects, who will use it during the design phases. It is supposed to work as follows: A group of architects are gathered in their normal work setting, with all the normal appliances. Putting on shutter-glasses connected to a PC and a vision tracking system, they will be able to see 3D graphical models of arbitrary objects (house, tree, car, wind turbine etc.). It will make available tools for altering and positioning the models.

  The ARTHUR project is being developed by some of the same people who were involved in the VRT project and makes use of some of the best ideas from the VRT system, but adds further functionality to it. The most important parties in ARTHUR compared to VRT from our perspective are: Saab Avionics, which is developing a high resolution head mounted display and the CVMT research group at Aalborg University which is developing the computer vision tracker system.

  The ARTHUR project use PHOs much in the same way as the VRT system - to influence the data one must influence the physical object. This notion limits collaboration to a local setting, in as much as the virtual objects coupled with physical objects are concerned.

  The new GCVR system will differ fundamentally from ARTHUR because it will make available the same functionality in a remote setting as well. As mentioned in the introduction future inhabitants of a city plaza can make their opinion available to either the architects or the city planners - an opinion derived from having actually experienced the new plaza from their private computer (not necessarily situated in the same room as the architects). In addition, if applicable, remote users will have the opportunity of altering the virtual environment (VE).

- **DARCDA**[1]
  Distributed AR as presented in [AKea95] augments a video stream of the real world with virtual objects. The augmented video is then streamed to each user who sees the result on a high resolution monitor in 3D. Our approach uses similar techniques. We use existing information about the physical world (ARTHUR uses vision tracking of PHOs) to act as information brokers to a remote VR environment. Given knowledge of each physical object our approach will enable a 3D stereo experience at the remote site, instead of a 3D video feed. Additionally, we incorporate the benefits users get from actually sharing the same physical space, DARCDA does not. Users in DARCDA would still look at a monitor even though collaboration takes place locally.

- **Office of the Future**
  The futuristic ideas for the office of the future presented in [RWC$^+$98] offer many ideas on how CSCW can be supported in a local as well as remote setting using the same equipment. They use spatially immersive displays (SID) in a non-intrusive way (also known as ubiquitous computing). Previous SIDs (i.e. CAVE) required highly specialized equipment but gave the user a 3D stereo view of the virtual world - the office of the future does not. In Office of the Future the traditional SID is replaced by existing surfaces (i.e. walls and tables). The imagery displayed on the SIDs are then rendered in 2D. All imagery is comprised of a video feed enhanced with virtual data - that is, all users and their offices are captured on video.

  The idea of this office coupled with ubiquitous computing is that it can act as a normal office when no distributed collaborative work is required. The fact that the office is capable of much more is invisible. It is when distribution is required that the office comes to life. Images of remote offices and users are projected onto existing surfaces, so that it appears to users that everybody are located locally.

  This application does not use PHOs. Manipulation of the virtual environment is done directly to the virtual object. We wish to keep PHOs as the interaction unit when collaborating locally, in addition to the ability to collaborate from a remote location. Keeping the third dimension also has the advantage that users can move around, over and into objects in the VE at their leisure. In addition GCVR provides both an AR and a VR setting - GCVR is still applicable when collaborating locally.

- **Studierstube Workspace**
  The Studierstube Workspace as described in [FS] and [FSH00] is a toolkit extension to the Studierstube system. The Studierstube project resulted in an AR system capable of supporting multiple co-located users working on a single model (called data context in [FSH00]). The workspace extension adds two distinct functionalities: Multi-context and multi-application. They try to make a (as much as possible) general toolkit to use with AR applications.

[1]Distributed Augmented Reality for Collaborative Design Applications.

Like our approach they see the benefit of incorporating physical objects into a VE. In the Studierstube project the physical objects act as tactile interfaces to the VE, much like ARTHUR uses PHOs. Each user is equipped with a pen and a pad through which interaction is accomplished. However, PHOs in ARTHUR and GCVR are shared among all local participants, encouraging users to use every day turn-taking techniques.

In many ways the Studierstube Workspace project is like what GCVR will eventually become - multiple users can work on the same graphical 3D model from remote sites. Each user is able to see the model from his own point-of-view and interact with it using the pen and pad (or PHOs in GCVR). The world is seen through shutter glasses worn by each user.

The major differences are that the new GCVR system will use both AR and VR technology, whereas they only use AR.

- **CAVERNSoft**
  CAVERNSoft [LJD97] is a collaborative software system which runs on the CAVE Research Network which uses CAVE-based virtual reality hardware, high-performance computing resources and high-speed networks. The aim is to support collaboration in the areas of design, education, engineering and scientific visualization.

  The CAVERNSoft system is similar to the GCVR system in some respects. However, where the GCVR system is in part aimed at low-end computers residing on low-end networks, the CAVERNSoft system is aimed at state-of-the-art hardware and supports collaboration within most of the areas mentioned above.

  In general, one could say that where the GCVR system is a small and relatively simple framework on which collaborative VR systems can be built, CAVERNSoft is a large, full-featured, customizable VR system aimed at both stand-alone and collaborative VR applications.

As can be read from the above description technologies capable of accomplishing some of (or similar) effects of what GCVR set out to do already exists. However, we believe that the new GCVR system can provide an alternative to all those projects in some way (as described above). The major difference is that the GCVR system combines a pure VR and an AR-like environment, enabling users both in a local and a remote setting to collaborate on arbitrary models. This in addition to the capabilities of the earlier GCVR system as described in [JUV02].

2.2 Types of Reality

The design of the GCVR system was intentionally made as widely applicable as possible, which is why the GCVR system can be applied in both an AR context as well as a VR context. In fact the GCVR system can be applied in the entire mixed reality (MR) spectrum as shown in figure 2.1 as well as VR. In the article [MK94] Milgram and Kishino defines what mixed reality is and how the different types of reality relates to the virtuality continuum. Most people are familiar with what VR. However MR comprising AR, and augmented virtuality (AV) will be introduced according to the definitions in the article. In figure 2.1 Milgram's virtuality continuum can be seen. It shows how the paradigms are distributed in the continuum.

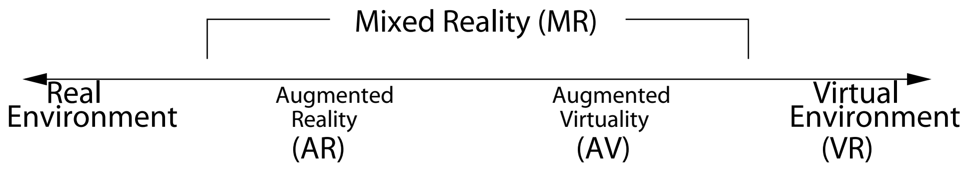

Figure 2.1: *Milgram's Virtuality Continuum.*

MIXED REALITY

Mixed reality covers the entire paradigm of merging the virtual and the physical and is a very broad definition. Mixed reality covers the definitions of augmented virtuality and augmented reality but not the completely artificial reality.

An advantage with regards to how users perceive the reality of the 3D experience is that it is easier to show the user what a given scenario will look like in real life after the proposal have been implemented in MR than it is in pure VR. E.g. when planning wind turbine sites one of the requirements is typically that the turbines are not allowed to be an inconvenience to people living nearby. By using some form of MR it is possible to show users how the turbines will look from their house - how big it will seem, how the turbine will cast shadow at a given time of day of the year etc.

AUGMENTED REALITY

In [MK94] the definition of AR is that:

> *"As an operational definition of Augmented Reality, we take the term to refer to any case in which an otherwise real environment is "augmented" by means of virtual (computer graphic) objects. . . "*

When talking about real environments, we define it to mean all types of existing physical environments that one can see, both directly in the real world or by way of a video feed on a screen.

Typically an AR environment can be obtained by using see through glasses in which only a small part of the glasses are filled with virtual objects.

AUGMENTED VIRTUALITY

AV is closer to a completely virtual environment. In this context, what is being augmented is no longer the real world, but a computer generated world. In AV the virtual objects can be augmented by e.g. overlaying the face of an avatar with a video feed of the human that the avatar represents and thus convey facial expressions in the virtual world. An AV environment is primarily created using conventional immersive or non-immersive graphic displays.

Virtual Reality

In VR, the computer is responsible for creating everything the user can see. In AR the system either receives an image of what the user can see or a semi-transparent display is used. The computer's task is then to place the virtual 3D objects in this image.

In a VR context, much effort would have to be put into creating life-like 3D models to present the user with as much information as in MR and AR in particular. E.g. it requires much computation to render facial expressions and haptic feedback in VR, while these factors can be seen directly in AR, and thus no processing is necessary to accomplish this. What is important to note here is that in [MK94], VR is defined as being a pure computer generated world. It is very hard to create a very realistic computer world without augmenting it with textures or images from the real world, and thus moving the framework towards an AV context as defined by Milgram and Kishino. However it is common today that VR is used as the terminology for such virtual worlds, and we will therefore adopt this definition.

2.2.1 GCVR in the Reality-Virtuality Continuum

In our project we wish to support platforms using MR, as well as platforms using VR. This way we will be able to support collaborative work between e.g. a round table collaborative system and a collaborative VR system in the panorama or CAVE. In round table sessions, the hardware will typically be much slower than the dedicated hardware available in the VR arenas.

Even though the hardware is slower the round table users must still maintain a good level of immersiveness so that users feel they are present at the site. Head tracking for the individual users is necessary so that they can move their head and look at the virtual objects from different angles. As input in the AR system where users are in a real world context it is natural to use real world objects to interact with the world. One could imagine computer vision based interfaces, or even simple interfaces such as keyboard or mice present at the table or maybe a combination.

3 Application Context

An objective of this project is to determine which features are necessary for the system to be usable in practice. For this purpose we have contacted a Danish energy consulting company - EMD[1]. Amongst other things they present suggestions to customers and in this process some iteration is involved. It is our intention to use EMD for supplying the use case and a scenario in which our system can be applied to assist in cooperation. The following section contains an introduction to the areas of work EMD performs in which we can apply our system.

3.1 EMD

EMD's primary task is designing wind farms, small as well as big, and they work with both land-based and sea-based wind farms. Their main tool for assisting them in this task is a software package they have developed in-house, but at the same time it is also sold to clients who wish to do the calculations themselves, instead of hiring EMD consultants.

3.1.1 EMD's work process

The final result of EMD's work is a presentation for the customer and other interested parties, but for EMD to be able to do a presentation the following needs to be carried out:

1. Site inspection
   (a) A site inspection is performed during which pictures of the surroundings are taken from the selected viewpoints.

       This means that the viewpoints are static and therefore must be decided upon in the beginning of the process. Otherwise the consultant will have to do another site inspection which is a costly affair for the customer.

   (b) During the site inspection it is also noted whether or not trees, hedges and structures in the area fit with the map information for that site.

2. Creating the energy map.
   (a) The map of the area is analyzed in the office. In this analysis hedges, houses and other information that has influence on the wind flow is entered into their calculation model in order for it to be able to calculate the wind flow on the site more accurately. Together this information forms what is called roughness data.

   (b) Height contours of the area are entered into the system together with the roughness data. They provide the model with a calculation model of the area which is as accurate as possible.

[1]EMD was formerly known as Energi- og MiljøData, however the name was changed to the abbreviation in 2000.

   (c) Wind statistics from the national weather service is applied to the landscape model and together with the roughness data and the height contours an energy map of the site can be calculated. This energy map is static once it has been calculated, unless information in step 2a or step 2b is changed.

3. Positioning the wind turbines

   (a) The consultant's next job is to place the wind turbines in the area so that noise requirements are met and production is optimized. This step can be iterated over and over without having to perform step 1 and step 2 again.

4. Calculating the production

   (a) After having placed the turbines, the total production for the wind turbines is calculated, based on the energy map and the data of the wind turbines, and a data sheet of the site is printed on a poster for the presentation.

5. Creating the visualization

   (a) A visualization, a so-called photo-mockup, of the area is made. The visualization can only be made from the viewpoints from which pictures were taken in step 1a.

6. Presenting the results

   (a) Finally the project results, the calculations and the visualizations, are presented to the customer, the neighbours and the authorities. Usually it is not until this point that objections to the site layout are made.

As can be seen from the above process, it is a cumbersome job for the consultant to do such a site calculation. If changes are made in step 6, he will have to redo steps 3, 4, 5, and finally the presentation in step 6. However, the amount of presentations are kept to a minimum, since they are quite expensive for the client, and also rather difficult to arrange in that all parties have to attend and approve of a design in order for it to be accepted. Therefore, the final decision on the site layout may not be what is the optimum solution in which all parties are happy with the results. Our contact at EMD has stated that tight budgets in the projects often inhibit the amount of meetings. This is because all parties must travel to the meeting, a presentation must be performed and if alterations have to be made, EMD goes back to the office and creates a new presentation after which another meeting is scheduled. In an attempt to improve this process, they often present two suggestions at each meeting and then the customers and the critics can better describe which type of site they prefer so that the consultant has a better idea of what is required.

It all boils down getting the acceptance of the government, the neighbours and environmental organizations if it is close to an area in which they hold a special interest. Therefore, the visualizations are imperative and must be performed in order for EMD's customer to get their project accepted. One such photo mockup can be seen in figure 3.1.

In such a mockup the image is static and for each time of day and each viewpoint a new mockup must be created and a new photo must be taken. This means that the

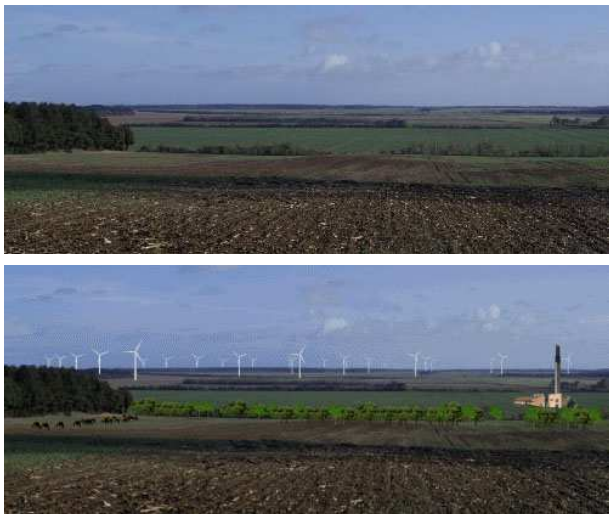

Figure 3.1: *Example of EMD's photo montage tool. The top image is the original picture taken on site, and the bottom picture contains a mockup showing the site in a future representation.*

client can only get a visualization that shows the site at the time of year and day the picture is taken. If they wish for visualizations during both the fall, autumn, summer and winter, or simply just in the morning and in the afternoon - different photos will have to be taken. Apart from the fact that it will have to take place over an entire year or day, it will be quite expensive for the client to have a consultant go take the pictures since an entire day is spent each time. One could argue that the clients could take the pictures themselves. However, this requires intricate knowledge of photography and how EMD's photo visualization software works, since it is crucial for the visualization that the correct focal length is used and that the camera is focusing on the exact same point during all photo sessions. This is not a simple task. Together with the cost, this is probably why it is rarely done this way.

3.1.2 Collaboration

By the means of a system based on this report, EMD will be able to provide the client with a mockup of the wind farm very early in the consulting process, which the client can then show to the affected parties, so that they can better consider the effects of the layout visually as well as geographically. Large wind farms often meet resistance in the community due to the fact that wind turbines are so large and that wind turbines generally disfigure the environment. However, criticism is often based on maps of the area and dots on that map and visualizations from predefined viewpoints. This is typically a time- and money consuming process. If people were able to see what the environment would look like by being in deity mode or mortal mode as described in section 1.1, it would provide both the proposal advocates and opponents with a much better reference frame for their arguments concerning the visual impression. Also, it would be possible to see the site from all the viewpoints one could imagine since it is a virtual model in 3D. Also, one could show different times of year simply by changing the texture of the landscape and the artifacts in the landscape. An inexpensive method for creating 3D landscapes and the use of standardized 3D models could lower the consumption of time and money considerably.

In the realm of collaboration, a GCVR-like system would introduce the possibility to have meetings over the internet in which many intermediate suggestions could be shown over a short period of time and with a relatively small effort. Thereby such a collaborative system could possibly improve on the result since people have had more options and by being able to see the site from all angles it will make it much easier to say what they like and do not like about a project.

A third possibility would be that a consultant is located in his office and some of the meeting participants are located in a meeting room in another county or country while others are located in their homes. In this setting they could then use the network to communicate and show each other solutions and ask questions and they will then be able to alter the suggestions during the meeting and see the results right away since the consultant does not have to do a new photo mockup. There are many possibilities in such a system and more of them will be explored in section 5.2.

4 Coordination Mechanisms

In this chapter we will describe a method for analyzing means of coordination. We will then apply this method to the GCVR system, the VRT project, and EMD's current procedures in order to place them in the context of CSCW. On the basis of this classification, it will be possible to determine whether or not the systems are compatible, and to associate common CSCW concepts with the problems encountered.

The fundamentals of the method were developed by Kjeld Schmidt [SS96] and have been applied to several projects, such as bug reporting within the S4000 project at Foss Electric and others. A more recent study [ACN00] has identified four key areas that must be analyzed in order to classify a system.

It is the method applied in the article [ACN00] that we will adopt throughout the rest of the analysis. In the article, a comparison is done between an artifact based coordination system in the form of the bug reporting system used on the S4000 project, and a verbal coordination system used on the bridge of the container carrier M/S Sally Mærsk. From this comparison they reach the conclusion that coordination in a CSCW system can be classified from a pragmatic point of view: Persistence of the communication versus non-persistence and active versus passive. And from a semantic point of view: Process versus state and field of work versus work arrangement. The definitions are given later on in this section.

As our project applies coordination between different co-located and non-co-located people utilizing verbal as well as artifact based coordination (alterations to the 3D world) the method developed in [ACN00] can be adopted in this project.

For an in-depth description of the method, the following articles must be consulted: [SS96], [SCSD96] and [ACN00]. As an introduction to these definitions we will start out with the definitions as given in [ACN00].

- **Persistence/Non-persistence**

  > *"By a persistent medium we mean a medium that maintains its information over time. In non-persistent media, information is lost and must be recorded elsewhere."*

- **Active/Passive**

  > *"By passive media we mean media that cannot cause actions to happen by themselves, whereas active media can do this without human intervention (i.e. when executed on some machine)."*

- **Process/State**
  [ACN00] describes a process oriented mechanism as also defining the protocol by which collaboration and the future is achieved. From [SS96] we have an abstract description of such a *coordinative protocol*:

  In proposition 5 of this article it says

  > *"A coordinative protocol is a resource for situated action in that it reduces the complexity of articulating cooperative work by providing a precomputation of task interdependencies which actors,*

> *for all practical purposes, can rely on to to reduce the space of possibilities by identifying a valid and yet limited set of options for coordinative action in any given situation."*

Our interpretation is that a protocol is simply a set of procedures and rules defining how collaboration should proceed. Additionally these rules stipulate what must be done to achieve the future and what happens during i.e. break-downs.

- **Common Field of Work/Cooperative Work Arrangement**
  In [SCSD96] they define the common field of work as follows:

  > *"The interdependent tasks and the world of objects and processes,within which they are performed, are referred to as the common field of work in an attempt to underline the difference between isolated work phenomena and work tasks that have a bearing on actors, and beyond the field of work of the individual actor."*

  From all three of the earlier mentioned articles we have the following interpretation that the common field of work is defined as the above definition and that the cooperative work arrangement is *a system of multiple actors who are interdependent in their work.*

The following section contains our interpretation of the above four terms constituting the domain of coordination mechanisms.

4.1 Pragmatic Dimensions

In [ACN00] it is argued that the classification mechanism can be split into a pragmatic and a semantic part. The pragmatic classification mechanisms describe features of the medium through which communication takes place.

The features describing the medium are the persistence and non-persistence, and whether the system is active or passive.

4.1.1 Persistent or Non-persistent

What decides whether coordination is done in a persistent or non-persistent way is the type of communication. If the communication is artifact-based, that is e.g. written down or shown in schematics or otherwise persistent, then the method is also persistent. If, however, the communication is performed verbally then the communication is non-persistent.

The main difference between the two types of communication is that in artifact-based communication the present state of the field of work, its history and possibly its future are made persistent and thus publicly available. This is not the case in verbal communication.

When considering the two methods of communication, it appears that a persistent method is preferable. However, it is not so in all cases, which the following example shows. In [ACN00], an example of this is the analysis of the communication on board the M/S Sally Mærsk. In the analysis it is noted that even though it would improve the work setting for the people to communicate using an artifact based communication method in which information become persistent it is not a flexible enough method for the scenario. On board the ship, decisions can have a serious impact and it must be possible to alter the wrong decisions in a very fast manner. Therefore, communication must be very swift and efficient. Persistent communication is not very flexible since it requires the information to be transferred to the field of work or to the medium before people receive the information.

In the rest of the report, we will denote coordination where all information is somehow transferred to the medium as being persistent. This means that the communication is artifact-based and therefore the medium maintains its state over time. E.g. the communication is written down on paper, or described as alterations to a 3D model. Non-persistence shall be known as the opposite, in which information is lost.

4.1.2 Passive or Active

Decisive for whether the system is passive or active is whether the system can cause actions to happen on its own. In [ACN00] there is an example of an active system used for monitoring ships on the river Elb, Germany. The system is located at the vessel traffic service (VTS Elbe) station in Bremerhaven. This system is an active system, since it updates the position of ships without human intervention. It also warns the operator if two ships are in a one-ship only zone. The system also predicts whether or not two ships will need to pass each other in a one-ship-only zone.

We define an active system as a system, which causes actions to happen by itself, as described in the example above. A passive medium is a medium which cannot cause actions to happen by themselves.

4.2 Semantic Dimensions

The semantic dimensions relate to the meaning of the medium: Process and/or state and field of work and/or work arrangement. The semantic dimensions can be seen as a continuum. In several cases both of the opposing dimensions will be present in some more or less dominant way. In these cases the dominant dimension will be selected.

4.2.1 Process or State

The concepts of process and state are a bit more loosely defined than the pragmatic dimensions. Therefore, we use examples of classifications of process and state systems and explain why they belong to one or the other.

The VTS system is a system for indicating possible problematic situations on the river Elb. The system does not explain how to solve the problem, it only gives information about the state. Therefore, the VTS system is a state oriented system. That is, a system wherein the current state and possibly future states are described, but it must not have any descriptions of other parts of the process whether future or past.

In the S4000 project, the computerized bug reporting scheme contains, not only the current situation, but also the history and what the next step is according to the protocol. Therefore, the S4000 project is a process oriented system, since it defines what actions should be taken from here, but it is also a state oriented system, since it describes the current state.

We define a process oriented system as a system in which a description of the process and what to do next is explained.

4.2.2 Field of Work or Work Arrangement

We will adopt the definitions of [ACN00] for field of work and work arrangement. A work arrangement is classified as a group of people performing a set of tasks. A cooperative work arrangement is a work arrangement in which the tasks are interdependent. This means that a cooperative work arrangement is defined as a set of tasks and who those tasks should be performed by.

Tasks oriented towards the field of work are the interdependent tasks, the processes and objects within which the tasks are performed. Field of work is thus a classification indicating a state of the real world.

We will adopt the definition made in the introduction to the current chapter.

In the following section we will apply the described framework to existing technologies such as GCVR, VRT and EMD's own software. This way we will be able to classify these systems in a way that allow us to determine whether the systems and the way work is done with them are compatible with the other systems.

4.3 Discussion of the Virtual Round Table Environment

The VRT is - in the words of its inventors - a collaborative augmented multi-user environment. It is based on:

- Collaboration between multiple users.
- Augmentation of the working environment of the users (AR).
- Interaction with 3D objects.

The basic idea is to produce a perspectively correct 3D stereo visualization of a virtual world within the physical working environment of the users. The users are intended to be able to see each other and use normal human behavior (e.g. speech, gestures, facial expressions) for communicating with each other despite the fact that they are wired up for using the VRT. The VRT developers classify their idea as being within the scope of augmented reality.

According to the definition in section 2.2, the VRT system can be classified as an AR system. However, in this report we will classify the VRT paradigm as being within the scope of virtual reality. This is due to the fact that the augmentation of the physical world takes place without associating any significant real-world objects with virtual ones. The only physical objects associated with objects in the virtual world are place holder objects, which in turn represent the actual, real-world objects. If the system was an augmented reality system, it should be able to utilize real objects in a more general way than just as input devices, which is the case in the VRT paradigm. That is, the objects in the virtual world should be associated directly with the objects, which are represented by place holders in the current system. In section 2.2 the AR-VR continuum is described and the VRT does not fit perfectly into any of the categories.

In order to be able to determine the usefulness of VRT in relation to this project, we will classify it using the powerful framework described in the previous sections.

- **Persistent/Non-Persistent**
  The VRT-article (see [BMS00]) does not make clear which of these properties that characterizes the VRT. However, the nature of a system like the VRT indicates that a virtual world is persistent as long as it is not being restarted. That is, if and only if it is possible for users to leave and join the virtual world at pleasure. Since the VRT seems to be aimed at small groups of concurrent users who use the system for explaining something or for reaching an agreement, it does not seem likely that some users will leave the system, turn off their terminals, but come back later and join the session again. The nature of the VRT with regards to persistence has not been made explicit in the article describing it in its purest form. Also, there does not seem to be an immediate need for total persistence[1] since the scenarios for its use take place in a local setting. Therefore we will characterize the VRT as being non-persistent.

- **Active/Passive**
  The VRT is intended to support collaborative work, which makes the interaction of people its primary task. Therefore, the system is basically passive in nature. One might be able to find active elements which could enhance the collaborative work. However, since the article only describes the very basic elements of the VRT concept such specialized features have not been mentioned. We will classify the VRT as being passive.

- **Work Arrangement/Field of Work**
  Systems such as the VRT are basically maps with tangible objects on them. The maps describe the field of work. The responsibilities of the users, participating in the decision of how a map-state is configured is not described in any way - or taken the least bit of interest in. Therefore, the VRT is oriented towards the field of work.

- **State/Process**
  The article describing the VRT system does not mention any methods for recording the actions of the users in virtual worlds, nor does it describe what they should do in the future. Therefore, one must assume that a basic implementation of the VRT must be state oriented.

The results of the discussion of the items above are summarized in table 4.1.

The VRT collaboration method is non-persistent, passive and oriented towards the field of work in a state oriented fashion. These characteristics do not necessarily

[1]The term "total persistence" will be used for describing the kind of persistence which allows the user to shut down the system and restart later in the same state and also allows users to join and leave the virtual world at will.

	VRT
Pragmatics	+non-persistent, +passive
Semantics	+field of work, +state

Table 4.1: The properties of the Virtual Round Table.

mean that the idea cannot be developed into a powerful tool for improving traditional collaborative work.

4.4 Discussion of the GCVR system

The GCVR system provides a basic API for establishing distributed collaborative virtual reality systems. Its main features are:

- Distribution of the Abstract World Model (AWM) [JUV02].
- Visualization of the AWM.
- An API for performing changes on the AWM.

The basic idea was to make available a fundamental system on top of which specialized collaborative VR systems could be constructed. The main parts of the system are a networking protocol which enables communication on consumer-level connections, a visualization part which is able to perform well on low-performance equipment but also to increase its performance on high-performance equipment and a server part which is able to keep the visualizations of the virtual world consistent across the network.

As with the VRT, the GCVR system will be classified in terms of the framework described in sections 4.1 and 4.2:

- **Persistent/Non-Persistent**
  The GCVR system can be classified as being persistent because it enables users to leave a running virtual world and rejoin it later. The server part of the GCVR client/server structure ensures this. Meanwhile, other users are able to make changes to the virtual world. In addition, plans for making available a scheme for saving the virtual world to a file and to record each change made to the virtual world exist. Therefore, the GCVR system clearly provides a persistent environment.

- **Active/Passive**
  As is the case with VRT, the GCVR system is a passive environment. Since the system has not been aimed towards any specific application, no need for active functionality has been encountered.

- **Work Arrangement/Field of Work**
  The GCVR system is like the VRT because it also basically enables the users to manipulate objects which are located on a map, which in turn represents real-world locations and constructions. That is, it is oriented towards the field of work. However, GCVR supports a scheme for locking objects. This

scheme is a very small and simple step towards the introduction of turn taking techniques which would make the GCVR system more oriented towards the work arrangement. Nevertheless, this step is too insignificant for the system to be classified as being oriented towards the work arrangement.

- **State/Process**
  The fundamental data structure of the GCVR system is the abstract world model, which describes the state of the virtual world created on the basis of it. As such, the AWM does not record the actions of the users involved. Therefore, at the current state of the GCVR system, it must be classified as being state oriented. Because of the close coupling between the network protocol of the GCVR system and its AWM, it would however be relatively easy to support a partially process oriented approach by logging the network communication.

The results of the discussion of the items above are summarized in table 4.2:

	The GCVR system
Pragmatics	+persistent, +passive
Semantics	+field of work, +state

Table 4.2: The properties of the GCVR system.

The GCVR system is persistent, but passive and oriented towards the field of work and the state of it. Collaborative VR systems are difficult to turn into coordination mechanisms - and that is not necessarily the point of making them. The main reason for this is that VR systems like the GCVR system and the VRT are aimed at making it easy to manipulate virtual representations of the real world. Their focus is to introduce the way people think about things into the way people discuss things - to visualize their thoughts. The goal is not to put constraints onto the thoughts of the users, but to set the users free of the physical constraints of manipulating the real world to try out an idea. That being said, coordination of work is still necessary and the introduction of turn-taking techniques can support work arrangements.

4.5 Discussion of EMD

In the following section, we will discuss and analyze the mechanisms utilized by EMD in order to communicate with customers. This analysis will end up with a classification of the work form of EMD in relation to the scheme described in sections 4.1 and 4.2. We will bring into focus the artifacts used by EMD while cooperating, as these are instrumental in obtaining fruitful cooperation and communication.

Because we are only interested in the collaborative aspects of EMD's work, we will focus the analysis on identifying by which means EMD collaborates with customers. These are the areas where the new GCVR system can improve their work process.

4.5.1 The Artifacts

EMD currently uses two methods by which they communicate and collaborate with external sources. The first one is pictures of a future wind turbine site overlaid with computer generated images of wind turbines. The second is a newly developed application capable of run-time alteration of a group of variables (energy output, visualization etc.). Currently they are not using this application during collaborative sessions with customers. The artifacts will be described in the following sections.

The Pictures

EMD currently visualizes wind turbines on a future site using still-pictures. It is their way of showing, from a few points of view, how the aesthetics of wind turbines placed in a particular way on a particular site will be. Examples of these pictures can be seen in figure 3.1 on page 15.

So far as to show aesthetics these pictures have proven adequate in most cases. However, they have two inherent problems, as we see it, both of which stem from the fact that they are static in nature.

1. **Aesthetics**
   As they are static they cannot be manipulated on-the-fly. If a customer wants to see the turbines from a slightly different angle or wants to see how they will look in the evening, a new set of pictures must be made and distributed to customers. The more visible the wind turbines become the higher is the likelihood that many angles need to be observed. As such they are only able to visualize a glimpse of what might be.

2. **Information**
   EMD can put as much or as little additional information into these pictures as they deem necessary to communicate whatever they want to communicate. Let us assume that a customer in addition to aesthetics also wanted noise and energy output visualized. EMD could handle this in two ways: Put all data into the same photo or into three distinct photos. If one is used and the customer at some time during discussions only wanted to observe the aesthetic aspect, EMD would have to create an additional photo. If multiple photos are used, there is the possibility that the photos will be useless due to information under- or overload.

   What it all boils down to is that what is shown in those pictures is there to stay. If additional data is needed or other points-of-view are required new photo sets must be created.

In relation to the coordination mechanism method we will now classify the picture artifacts within the four areas:

- **Persistent/Non-persistent**
  The picture sets are persistent in as much as they maintain the data they are meant to convey at the point in time where they were created.

- **Active/Passive**
  As also mentioned above they are static - the pictures in themselves cannot do anything. This means that the picture sets are passive.

- **Process/State**
  The pictures do not tell us how to achieve the future nor what has passed. They show us the state of the project in a possible future.

- **Work Arrangement/Field of Work**
  They show state information on different interdependent objects, hence they represent what is known as a map, see [ACN00] for details.

  The pictures contain no information about a work arrangement as there are no mutually interdependent actors involved. Relating to how we defined field of work in 4.2.2, the pictures do describe objects, namely the aesthetics of the whole but they do not define any processes by themselves. Looking at their intent however we find the processes. They are meant as information brokers during meetings. When discussions turn to pros and cons about what they tell us, we start an interactive process of refining those pictures.

The results of the discussion of the items above are summarized in table 4.3:

	Pictures
Pragmatics	+persistent, +passive
Semantics	+field of work, +state

Table 4.3: The properties of EMD's photo-mockups.

The picture sets represent a method that is persistent, passive and oriented towards the field of work in a state oriented fashion.

THE APPLICATION

The application (WindPro[2]) has been developed in-house at EMD over the last few years. It is meant as a combined calculation and visualization tool capable of drawing upon different data sources (wind flow charts, turbine models, noise areas etc.). A screen shot can be seen in figure 4.1:

The upper half of the picture shows a visualization of the area in which the wind turbines are scheduled to be placed. The lower half shows the aesthetics of the placement from a given point-of-view which the user can change at any time. Users are able to move turbines around in the bottom half and, by clicking an appropriate button update energy gain and noise areas. The application will on-the-fly update the visualization above based on commands and attribute changes made by the user (wind turbine attributes, roughness data, additional wind turbine, point of view etc.).

One of the features relevant to our system is the history feature. It displays a history of state information of all wind turbines. Whenever the user wishes he can recall an earlier state.

WindPRO has no support for collaboration, local or distributed, other than additional viewers can see what happens on the screen over the shoulder of the one controlling the mouse, or the display from the screen can be projected onto a wall. It is not meant to replace the picture sets, but more as an additional mode of visualization. It is also not meant to be used during every session of all projects, but in the beginning only in high-profile projects.

[2]WindPRO and documentation can be obtained from www.emd.dk

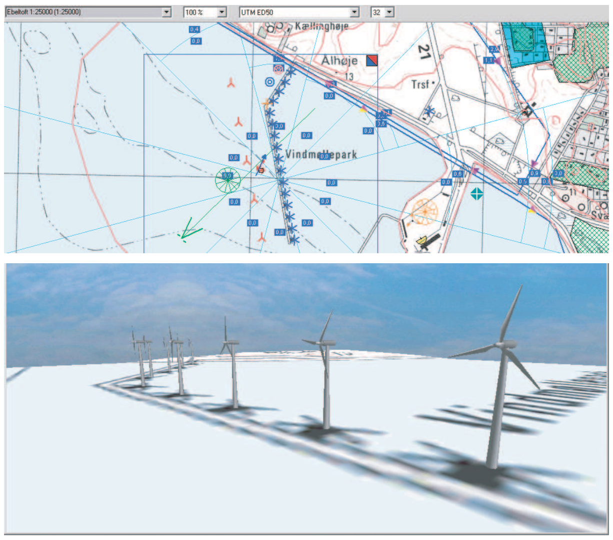

Figure 4.1: *Screen shots from the WindPro application. Both windows are shown simultaneously. The top one shows the wind turbine site in 2D, with various objects indicated (wind turbines are blue crosses.). The bottom picture shows the same site rendered in 3D*

In relation to the coordination mechanism method we will now classify the application within the four areas:

- **Persistent/Non-persistent**
  In so far as users remember to save their session the application is persistent.

- **Active/Passive**
  As with the picture sets it is also primarily passive. However, it also has features that turn it towards an active application. Depending on how the user issues select-move-select commands on wind turbines, it can on-the-fly update energy output and noise production. This feature can be dis- or enabled as it requires a lot of computational power.

- **Process/State**
  Together the top and bottom displays in addition to the underlying data structures describe the current state of the application. The history feature mentioned in the description contributes to also defining the application as being process oriented, as it is possible to recall any previous state.
  The history feature mentioned in the description could contribute to also defining the application as providing a process. However the history feature is only a collection of earlier states, it does not stipulate procedures or future actions nor how that state was obtained. As such we define the application as primarily state based, slightly orientated towards process.

- **Work Arrangement/Field of Work**
  Basically the application defines a set of objects and attributes. It is a map of the part of the world that users cooperate around. These are the defining qualities of field of work.

The results of the discussion of the items above are summarized in table 4.5.1:

	WindPRO
Pragmatics	+persistent, +passive
Semantics	+field of work, +state, (+process)

Table 4.4: The properties of WindPRO.

WindPRO represents a method that is persistent, passive and oriented towards the field of work in a state oriented fashion. It has features which orient it towards both active and process, though.

The picture sets and the application have a very similar classification. Because they are both oriented towards field of work, they will be able to fit very well into a collaborative VR environment like GCVR. Additionally we can see that EMD has also expanded their own field of work to encompass process oriented features. This tells us that GCVR should also support these (process) features in the future.

4.6 Comparison of the Approaches to Cooperation

In this section we will compare the approaches to cooperation and collaborative work reflected in the three cases discussed in the previous sections. Two results will appear from this comparison:

- A description of the gains of incorporating the Virtual Round Table approach into the GCVR system.
- A description of how EMD's approach can be related to the combined GCVR /VRT system and be valuable both regarding EMD's work and the further development of GCVR.

In order to reach conclusions on these matters, we will discuss them in the following two sections.

4.6.1 Combining the VRT with the GCVR system

When looking at the property tables for table 4.1 on page 22 and table 4.2 on page 23 it appears that with respect to these properties, the only difference between the systems is that the GCVR system supports persistence, whereas VRT does not. The combined table is shown in table 4.5.

	GCVR and VRT combined
Pragmatics	+persistent, +passive, (+active)
Semantics	+field of work, +state, (+process)

Table 4.5: The combined properties of the VRT and the GCVR system.

The immediate conclusion which can be drawn from this fact is that the two systems can be combined into one powerful framework for collaboration. A representation of the virtual world which is easily understandable and manipulatable can be extracted. From the GCVR system, a framework for enabling users to collaborate in the same virtual world can be extracted. Another conclusion which can be drawn from the comparison is that the conceptual framework used does not cover all of the features of the two concepts. There are, in fact, great differences between the two systems which do not appear from the comparison of coordination mechanism properties. However, the similar properties mean that the two sets of ideas are compatible.

Nevertheless, when combining the two sets of ideas, a framework appears in which it is possible for people located near each other as well as far away from each other to participate in what appears to be a real world meeting. Some advantages and disadvantages appear for the people located near each other:

- Advantages:
  - They are able to cooperate in a virtual world, enabling them to manipulate virtual objects.
  - They are able to cooperate with people located far away in pretty much the same way as they would if they were located at the same table as themselves (assuming that the system is used in that situation).
  - They are able to see facial expressions and gestures on people located at the table.
  - They are able to participate in discussions using normal speech.
- Disadvantages:
  - They must adapt themselves to performing certain kinds of collaborative work in a way which is limited to the way it is supported in the virtual world.
  - They must wear HMDs and head tracking devices.

The advantages and disadvantages for people located far away from the meeting:

- Advantages:
  - They are able to participate in a meeting located far away from them on almost the same premises as those located near each other.
  - They are able to participate in discussions using an external speech application.
  - Depending on their way of participating, HMDs, head tracking devices etc., may not be necessary or even interesting.
- Disadvantages:
  - No facial expressions and only few simplified gestures will be available for supporting speech.
  - They may only appear as a form of "ghosts" in the meeting, not taking any primary roles in a session of collaborative work because of their limited presence.

The combined system can be implemented on the basis of the GCVR system, which provides some of the fundamental features necessary for implementing the round table concept in addition to some features which enable functionalities outside the round table concept. The union of the features of both concepts can be extended with a recording system, which makes the system more process oriented (if we disregard the fact that the definition of process orientation requires that future events must be part of the system - as described in section 4.7). Also, when aiming the system towards a specific area of application, active features will be interesting in the combined system. In case these ideas are implemented, the properties of the combined system will be as indicated in table 4.6.

	GCVR and VRT combined and extended
Pragmatics	+persistent, +active
Semantics	+field of work, +state, (+process)

Table 4.6: The combined properties of the VRT and the GCVR system extended with a logging or recording system and active application-specific features.

4.7 Discussion of the Process/State Continuum

During our analysis of coordination mechanisms in the GCVR system, the round table and EMD, we have encountered an unfortunate effect of the restrictions of the terms process and state. We have been faced with the fact that a mechanism must either be classified as process or state oriented. This has led to classifications of the systems in which for instance they were said to be state oriented, where in fact they were state- and part process oriented.

Let us briefly sum up what these terms cover. A mechanism is process oriented if it states facts about the past and stipulates how to obtain the future. This is often accomplished by rules and protocols. The state represents a snapshot of the field of work at a given point in time. Neither the past nor the future is represented in a state. Something worth noting about the two is that process describes the field of work in an almost abstract manner (i.e. protocol), while state describes an instance of the field of work.

It is our belief that we can (and must) redefine the two terms to some extend. This redefinition should not be seen as our way of removing an obstacle that is in the way - rather, what we are doing is to refine the terms, because we believe that the current definition is not clear and specific enough to describe two terms that in some instances fall so close to each other. Following is the redefinition that we wish to make:

- Process need not stipulate the future. Imagine an application that need not describe the future (GCVR at EMD for instance). The whole idea with this setup is that users are not aware of the future, the future is undefined to them so to speak. Labeling the new GCVR system as state oriented will be wrong, as it can describe the process by which users have reached the current state (using a recording feature).

When removing the future from the term process we broaden the definition - we weaken it in some sense. We can however still describe the original process term by making use of the terms *map* and *script* as they can help describe to which end of the continuum a mechanism belongs to. [ACN00] describes the two as:

> *”Map is a representation in which the state is the structuring factor, and the process must be inferred, whereas a script is the opposite.”*

The GCVR system describes a map, for instance. The new GCVR describes both a map and a script (when using the recording feature) in addition to being process oriented.

4.8 Relating the GCVR System to the Working Habits of EMD

In this section we will divide the way EMD works as described in section 3.1 into the three scenarios introduced in section 1.1: The round table, single- and multiuser. Following that we will describe the three scenarios again, but with the addition of the GCVR system. This will provide us with a framework from which we can see in which instances and by what means the GCVR system will be able to assist EMD in the way they collaborate with customers.

Let us start with a short description of the three concepts. The round table refers to instances during collaboration where all users are gathered in the same room, and all users are able to interact with the world. Single user refers to users situated in separate arenas, collaborating over distance. Multi-user is when more than one user is gathered around the same arena, but only one is able to edit the world.

EMD’s working habits can be subdivided in the following way:

- **The Round Table**
  During presentation of the ideas to the customers, they are all situated in the same room. In addition to the picture sets, EMD explains pros and cons for the selected layout - this also involves showing expected energy production and aesthetics. Artifacts in use could be the picture sets for visualization, paper notes and black boards for calculations, new ideas and thoughts. Anyone can comment the result, posing new ideas, wishes or objections.

- **Single User**
  EMD currently has no collaborative tool capable of distributed communication. If communication over distance is required it is accomplished using standard existing technologies like e-mail, standard mail, telephone or screenshots for instance. An additional way of visual communication could be to send project files from their own simulations application WindPRO to other users. This way distant users are able to see both a visualization of the wind turbine site and the data basis on which the calculations are based. The issue of how to communicate multiple changes to EMD in a useful manner remains though.

- **Multi User**
  Over the last few years EMD has created what could constitute a multi-user scenario - WindPRO. It is a complex editing and visualization tool in which

all variables and objects needed to create a virtual wind turbine park can be represented. One could imagine a session in which the display from the computer on which the application executes is projected onto a wall. In this way a master placed by the computer could show-and-tell the creation of a wind turbine site. Another angle could be to project the picture sets onto a wall - again the master could then show-and-tell ensuring that all participants have focused their attention to what he is explaining.

As can be seen from the above description, EMD can be represented in all three scenarios, which is fortunate as it means that EMD and its customers are, if not already, then at least on their way towards a way of thinking that will help make the addition of the GCVR system more easy.

In reference to chapter 1, concerning deity and mortal mode, the fact that EMD is represented in all scenarios has the direct consequence that both modes of interaction are utilized. For instance when EMD employees are in the act of creating the wind turbine site they would adopt the deity perspective - in these instances an overview of the entire site is the most useful. People living close to this site does not need to see the site from above - their concerns are how the site will look from their backyard. This will make them use the mortal mode.

When introducing EMD to the GCVR system we see the following subdivision. In all scenarios EMD is able to create a perspectively correct 3D representations of a wind turbine site.

- **The Round Table**
  During sessions where participants are gathered around the same table, users are equipped with HMDs which will enable them to see a perspectively correct 3D image of the wind turbine site. Anyone can change attributes of the environment (i.e. position of a wind turbine). A master having access to advanced controls has access to a computer on which the server executes. Three modes of interaction can be chosen:

  – PHOs are associated with graphical objects. Users have a relatively large freedom of movement, so they are able to see the site from different perspectives. They are able to move individual objects by moving their physical counterpart, the PHO.
  – PHOs are not associated with graphical objects. Users can be said to have total freedom of movement as they are able to rotate the world, for instance to see the site ’from the other side of the table’. In this mode individual objects cannot be altered.
  – PHOs are associated with functionality. This means that moving PHOs results in changing i.e. size, complexity, position, zoom etc. of virtual objects. Users can still physically move relative to the world however they all see the world ’through the same eyes’ - that is all users share the same point-of-view at all times. Only the person designated as administrator can change the point-of-view (of the entire group).

  Additionally EMD will now be able to show additional data during presentations. Customers may want see how far away the noise from the turbines can be heard, or see the site during sunset. They may also want to see directly how the impact on energy production will be when moving the site 200 meters east.

- **Single User**
  At any time EMD, the government, customers and citizens can meet in the virtual world, though they may be separated by great distances. Instead of having to explain an alteration through the phone or over mail, EMD can show the impact of an alteration on the spot while communicating verbally through an external speech application. Meetings can even be held without the interaction of EMD as mediator, in which those present can come up with a visual representation of their ideas and wishes where after they can present their arguments to EMD in an intuitive and informative manner.

- **Multi User**
  EMD can take participants on a round-trip tour of the site showing all relevant point-of-views, surroundings and any additional data coupled with those sites. They are able to fly over, through and around the site and show to all present how the aesthetics of the site will actually present itself with the added insurance that everybody sees things from the same point-of-view. This enables EMD to describe aspects of the plan a layman may not understand at first. Users in other countries can even participate on almost the same level as all others present. These remote users could be running the GCVR client using shutter glasses on their own computer. Verbal communication can be accomplished through an external speech application.

- **Additional**
  We will only touch the subject here, as a full description of the additional features that GCVR enables EMD to utilize will be made later in the report.

  One of the obvious benefits enabled by the use of a computer visualization system is the ability to add dynamic real-time behavior to objects. Animation for instance opens up a whole set of functionalities enabling EMD to immerse customers deeper into the illusion of reality and thereby obtaining better results and better feedback.

The GCVR/EMD scenarios have been extrapolated from the way EMD works presently and as can be read from the scenarios we see a definite possibility for the new GCVR system to be applicable in the context of EMD. The next step is to analyze these scenarios in a more abstract manner which will enable us to state facts about the functionalities and design perspectives we need to consider further.

5 Collaboration and Activity Theory

Using the framework described in chapter 4 we have been able to classify the different systems which we are working with. However, the classification says nothing about how an organization will perceive computer support, how it will utilize it nor exactly how individuals in the organization behave when working in groups. All these areas are crucial in order to be able to introduce more complex computer support in an orderly manner. We need an understanding of the dynamics of cooperative work and how groups handle break-downs and unforeseen events. We will base this analysis on activity theory (AT) which we believe is capable of describing these dynamics in such a way as to be beneficial for the development and introduction of the GCVR system to EMD.

Let us begin our discussion of cooperative work by defining exactly what it is. We will use two different sources for this definition: Schmidt [SS96] and Bardram [JB].

- **Schmidt**
  In his first proposition he writes:

  > *"Cooperative work is constituted by the interdependence of multiple actors who interact through changing the state of the common field of work."*

- **Bardram**
  When describing AT he writes:

  > *"AT describes cooperation as a collaborative activity, with one objective, but distributed into several actors."*

These theoretical definitions coincide very much with our own interpretation of cooperative work. Our interpretation is that collaboration is a joint venture (of multiple actors) towards some common goal. Each individual has a set of tasks that must be accomplished to enable all individuals to achieve this common goal. These tasks are in some sense shared in that on some level they are interconnected.

We can say that the goal can be diffuse because it might not precisely define that which all parties strive for - but on some level they all share this common goal. This can be seen from i.e. the point of view of a project group: All members (should) want to finish the project in a good and timely manner (the common goal). Individuals have differing roles in the greater scheme or even differing ideas on how this should be accomplished (the diffuse aspect of the goal). Also, individuals may have a private idea as to what should be accomplished.

When designing collaborative systems, which depend upon computers to achieve this collaboration, we need to consider, and preferably anticipate, how users will perceive such a system, in addition to how they can and will collaborate using it. Theory from the area of CSCW research based on activity theory can help us do just that.

We need to gain knowledge about not only the intentions of users, but also about how collaboration between a number of people is accomplished. So we will continue by identifying what collaboration is from an abstract point of view.

AT focuses on the dynamics in the interaction between a group of individuals and their activities with human activity being of central importance. Another important concept in AT is the notion of objects around which individuals shape their work and groups shape their interaction and collaboration. In a way the object defines the activity.

From AT we have a way of dividing the concept of collaboration into three distinct groups: Co-ordination, co-operation and co-construction.

- **Co-ordination**
  The interface between users, the way they collaborate, is static. They collaborate according to a set of rules or predefined roles in the organization. Objects are private in that they are only of concern to the individual using them.

  An example could be two teams working shifts at the same company. Team one relays information about the day's work to team two, by way of a standardized solution - a status report for instance. Another example is a project team in the final stages developing a system: The analysis and design phases are over and the project can be split up into distinct modules that need implementation. Given a module the role of the individual is well defined - adhere to the analysis and design. The means by which this is accomplished are based solely on the abilities of that individual.

- **Co-operation**
  Collaboration is directed towards a common goal. Individuals still have a distinct role in the whole and a set of rules to abide by. The task(s) the individuals are supposed to manage are not necessarily shared but they do overlap. This way collaboration is dynamic because users have to continuously adapt to the way other users act. Objects are shared in that all individuals are able to influence or change all objects involved.

  The way politicians negotiate the details of the tax legislation could be an example of co-operation. The goal is to enable a country to financially survive for another year - one politician wants to finance the new hospital by raising taxes, while another wants to reduce social support. Again the example of the project team can be applied: All the way through analysis, design and implementation individuals are given smaller tasks collectively supposed to make up the whole project. Both the means by which these tasks are assigned (discussions and meetings) and the means by which they are evaluated (seminars, midterm evaluation etc.) are co-operative in nature.

- **Co-construction**
  Co-construction is characterized by shifts in focus. At this level of cooperation the object around which the individuals are gathered might not exist. Words like reflection and iteration comes to mind in an effort to bring a common object to life or alter an existing one. If objects do exist they evolve continuously, changing shape and behavior.

  The project team can again be said to exhibit co-constructive behavior. Iteration and reflection during collaborative group work is one example of co-constructive behavior.

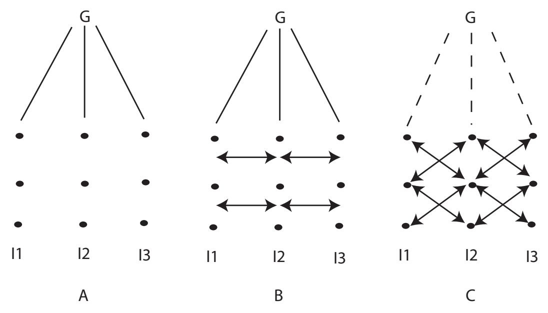

Figure 5.1: *A graphical display of the three activity theory concepts. From left to right: co-ordination, co-operation, co-construction.*

In figure 5.1 three individuals are shown: I1 is individual one and so on. The small black dots represent objects each individual utilize in order to reach the goal (the top-most letter 'G') The vertical row of black dots represent an amount of objects that each individual uses to reach his or her objective. The arrows in the middle figure (co-operation) show that the collection of objects are shared between users. In the right figure (co-construction) they show that existing objects are shared and that they change continuously. The dashed lines in the rightmost figure indicates that the path to the goal is not well defined (it changes continuously).

These concepts where also used in [Sch94] a few years earlier in a more detailed manner. Basically the author divided collaborative work into five areas, some of which are i.e. part co-operation, part co-ordination. We will proceed with the assumption that the above grouping is adequate.

However, this division is very coarse, so we need a way to move between the three as most work areas cover more than one of the three. It is important to note that this subdivision is purely analytical - they describe different aspects of *the same* activity. Imagine a case in which a group of people are gathered around a monitor, discussing and arguing about the objects visualized. At this point they exhibit co-operative behavior. Finding a fundamental error in those objects they turn away from the monitor to look on each other to start a dialog on ideas of how to proceed - how should or could the objects be changed to eradicate this error. Now they exhibit co-constructive behavior. This transition from co-operation to co-construction happens seemingly effortlessly and continuously. So for our application to be used in a cooperative setting it needs to be able to support these transitions.

When trying to introduce our application to an organization we must face the fact that what we might be doing will radically change the way they perceive their work. Not only that, but we may introduce a totally different way of accomplishing their goal. We must not be blind to this fact as it may result in the eventual rejection of our application. We face either of two problems should the organization not want to change their work arrangement:

- Either we must make sure that the application stays within the boundaries (as mentioned above) and abides by the same set of rules as the organization has previously used,
- or we must make a transition between work arrangements - for instance by

standardizing the way users communicate changes to a design we move towards co-operation.

In general we will want a transition:

- **Towards co-construction**
  This is to enable a more dynamic and flexible work arrangement. Either the reason is in anticipation of how the future will evolve or the wish to broaden the application context of a work arrangement. Schmidt in [Sch94] uses a similar argument when stating the reason for the need of advanced information systems to facilitate distributed decision making in modern work arrangement organizations.

  If the introduction of a CSCW system drives the collaboration towards a more open forum in which the human-to-human interface becomes more dynamic and complex, the collaboration is driven towards co-construction.

- **Towards co-ordination**
  We can also move towards co-ordination. The area of work might not necessarily be highly dynamic. In cases where the work is aimed at a more or less well defined common goal, the application can benefit from specialization. This is where a move towards co-ordination will help us gain more control, reduce the work to a more simple representation. Our application could introduce a dynamic work environment but still be co-ordinated in that it gives all parties a common ground with a set of shared rules.
  If the human-to-human interface is relatively static and based on some form of formal agreement, as in the case of an employer - employee relationship [BN99] for instance, the design of the system should reflect this - for instance with the help of standardized solutions.

In this section we have introduced one model for structuring a collaborative work environment. We have stated that any work arrangement is dynamic, continuously changing focus - in the course of a collaborative session, individuals inter-operate on differing levels reflecting the requirements of tasks and subtasks. In the next few sections we will focus on identifying whether the new GCVR system will support cooperation, coordination and/or co-construction when introduced into an actual application context.

5.1 SUPPORTING TRANSITIONS

In this section we will describe how transitions are or can be supported by the new GCVR system. We will begin by describing the set of possible transitions and continue with a discussion on how each transition fits into GCVR from the point-of-view of a possible scenario.

Three sets of transitions can be readily identified:

- Between co-ordination and co-operation
- Between co-operation and co-construction
- From co-ordination to co-construction (or the other way around)

CO-ORDINATION AND CO-OPERATION

A scenario could be as follows: Two people are sitting in two different buildings working on the same general project - this is co-ordination. They have agreed that coordination meetings should be held at certain intervals during a week. The purpose is to keep one another up to date on the status of each subtask. Not having the time to meet in person they meet in a virtual environment like the one offered by the new GCVR system. They can show graphically how much work they have done and talk about it verbally. This only covers the co-operative level, not the transition though. We can support the transition by using a scheme where 3D models (which the above two persons are working on) are automatically distributed. As such a server could be running all the time with both persons running a client. Models are updated all the time and these changes will take effect in the virtual environment automatically. Additionally changes made to the whole scenario could be saved - when the two users return from their meeting they could have the ability to visualize the scenario in their own office.

CO-OPERATION AND CO-CONSTRUCTION

Using the same scenario, the GCVR system in its current state supports co-construction to a limited degree. One can argue that for the GCVR system to be able to support true co-construction three features are needed:

- The system must be able to reflect the changing nature and behavior of the artifacts to be co-constructive. This could for instance be accomplished by being able to edit 3D models or dynamically create them on-the-fly.
- Being able to attach additional information not explicitly affecting the aesthetics of 3D models. This is needed in order to reiterate in an orderly fashion for instance. A need for being able to trace the paths taken arises, in order for the implicated people to be able to chose the best one at some point.
- Being able to bring up previous states - for instance being able to have three different proposals for how a certain city plaza should look could be necessary. This could be accomplished by being able to save and load models or entire scenarios.

The GCVR system supports co-construction in a limited way: When multiple users are located in the same virtual environment, they are able to talk to one another (they can present ideas and different view-points concerning a model) and they can move models relative to each other - having all the building blocks of a house different users can place the windows in differing ways.

CO-ORDINATION AND CO-CONSTRUCTION

This transition is only present in our discussion in order to get the whole spectrum of transitions. It contributes nothing new to what has already been mentioned above. In the case of the GCVR system it is indifferent as from what level users start out on.

SUMMING UP

We will use this opportunity to mention the differences in moving between the three levels of collaboration. In fact in the GCVR system there is little difference in the way people should behave in the virtual environment. There are no buttons to press or phrases to speak in order to move to another level. It all boils down to what features and functionalities they decide to utilize. As such we can say that if the above mentioned additional features are accommodated, transitions are seamless and transparent to users.

5.2 THE APPLICATION

The GCVR system as it presents itself now has the potential to take many different directions - it has been tied to no specific application area. As described in chapter 4 we will move towards an area of interest that will challenge the GCVR system both technically and in a CSCW aspect. To enable us to make that move we start out by talking about how collaborative work is accomplished independent of application area[1]. Following that we turn towards the GCVR system in reference to theory and practice.

5.2.1 COLLABORATIVE WORK

In order to understand collaboration which takes place over distance, we first turn our attention towards co-located collaboration. When working with persons face-to-face we as humans utilize a number of skills, consciously and subconsciously, as means of coordination. These are speech, gestures, and facial expressions. These are the defineables, however there is also an undefinable aspect to collaboration. It can be covered by the term ’tacit knowledge’, defined in [Bar] as

> *”...knowledge that enters into the production of behaviors and/or the constitution of mental states but is not ordinarily accessible to the consciousness.”*

This definition covers what is ’between the ears’ of individuals that they cannot or do not access consciously. Everyday words like experience, implied, skill, perspective or shared understanding [SR96] have been applied in an effort to describe it. Whatever it is it cannot simply be put to paper, but it is part of what enables two persons to solve a task better than two individuals.

Within the scope of this project we will look more closely at speech and gestures. As for the undefineables we will turn a critical eye towards our chosen application context. We will apply activity theory to EMD in order to help us understand not only tacit knowledge in the context of EMD, but also the dynamics of their work.

- **Speech**
  As also mentioned in [JUV02] we recognized the need for audio communication. We still have no intention of implementing any form of audio - that is as for the design we don’t want to add an entire submodule integrated into our protocol. If time allows we will integrate an already existing speech transmission technology.

  Speech is imperative during co-constructive collaborative work, more than in any other area. As this level of work is highly dynamic participants must

[1]It is not totally independent as we require a human element present in the collaborative work.

be able communicate - this is how individuals express experience, desires and wishes. As Bardram described in [JB] this form of speech is covered by the term 'articulation work', describing *the work actors do, caused by their interdependency, in order to coordinate, schedule, mesh, integrate, etc, their cooperative work*. This should not be mistaken for the cooperative work arrangement which describes the actors who are interdependent in their work.

This does not exclude the fact that we need some form of basic graphical communication - along the lines of what we described in [JUV02]. These were basically just acceptance and rejection. What we need to consider later in the design is how to separate a system reject from a user reject, for instance.

- **Gestures**
  Gestures could possibly be introduced into the system by an external source as gesture recognition is a possible future extension to the ARTHUR project, so far as the new GCVR system is concerned the vision-tracking part of ARTHUR project will handle that for us. Gesture recognition as a functionality in GCVR is beyond the scope of this system though.

  Object tracking could be introduced by adapting the way interaction was done in the Studierstube project [FS] and [FSH00] - by means of a pen (and a pad). The pen is thus a simplified representation of the user's hand. Combined with added functionality like color and mass changes, we can create a relatively powerful tool by which users can communicate hand commands. In a CSCW perspective this way of interacting is clearly co-ordinated and in accordance with [SS96] we can call such an item an artifact to denote its special nature. As such we are able to dedicate a particular graphical 3D model with a very specific purpose to articulate hand commands.

So far we have introduced a CSCW theory based on activity theory which we will use to analyze an application area. We have stated that the basic units of operation are human activity and the objects they utilize. We have described the basic concepts of that theory and explained why dynamic behavior (transitions) are needed in order for the theory to be applicable in real life. In addition we have described some general attributes of the existing GCVR system, that we need to consider when applying theory to practice later on.

5.2.2 COMPARING THEORY AND PRACTICE

In this section we will apply AT to our application context, EMD. This part of the analysis is carried out in relation to chapter 3 in which the design aspects of EMD are discussed and in part section 4.8.

The structure of the rest of this section will be to use the bullets listed in chapter 3 and apply one (or more) of the three groupings of collaborative work, co-ordinated, co-operation, co-construction, to that context if applicable.

- **Site inspection**
  The site inspection is conducted by a person from EMD. One of the customers could be accompanying him to ensure that they have talked about the

same site. The EMD consultant has two tasks in this situation: Take pictures and ensure persistence between the actual world and the geological[2] data EMD has on record. Prior to the inspection EMD employees may move to the co-operative level of collaboration - they use their collective experiences and the data from the initial talks with customers to reach an agreement on how the site inspection should be conducted. Should the geographical data be inaccurate the consultant incorporates any changes into the data records at EMD and saves that data. If EMD discusses these changes collaboratively they move to the co-constructive level of cooperative work.

- **Creating the energy map**
  Again primarily one person is involved in creating the energy maps. This is usually also the one who did the site inspection. This person has a set of tools at his disposal to help him accomplish this task fast and effectively, many of which are incorporated in WindPRO. The rest are able to interface with WindPRO. The creation of the energy maps is a one man job as usually one person is able to handle all these calculations. This phase has a lot of variables to which the consultant must assign values. Some of these can be handled by consulting existing data records while others depend partly on the knowledge and experience of that consultant. Should the consultant require assistance or a second opinion during which they pit their experiences against each other, they exhibit co-operative behavior. More than one consultant from EMD might work on the same energy map over a period of time. However these two persons will *not* work on the energy map at the same time and as such these instances are only co-ordinated if information on how one employee changed the energy map is handed over to the second employee, which it is.

- **Positioning the wind turbines**
  As with the above two mentioned, positioning the wind turbines also only requires one person - putting this phase into one of the three levels of collaboration would thus be wrong. The tasks of creating the energy map and positioning the wind turbines are interrelated, so generally the same person who created the energy map also positions the wind turbines. However, this is not required as the energy map has a distinct protocol and syntax so any competent person should be able to position the turbines given the energy map.

  Another way of positioning the wind turbines is to execute an optimization routine. This tells WindPRO that is must try out every possible combination of turbine placement with the sole goal of achieving the optimum energy production. No aesthetics are considered in this calculation, so if a customer does not think that the placement is pleasing to the eye, changes might still have to be made by EMD.

- **Calculating production**
  All formulae used to calculate energy production are accessible from WindPRO. As such the consultant just ’presses the button’ and the calculated production is shown. However, at this stage in EMDs work there is the possibility to return to ’creating the energy map’ and ’positioning the wind turbines’ for either of two reasons:

[2]Both the structure of the earth and obstacles such as houses and forests.

  - To make changes to data basis. Experience could tell EMD that something is wrong with the calculations in which case they would have to iterate to resolve the situation, or
  - to make additional calculations on the existing data basis. This enables EMD to see additional configurations. Coupled with the history feature, EMD develops a database of different settings.

  These different configurations serve as a basis for in-house discussions for selecting the most appropriate configurations. Often though, only one possible configuration is created, but it happens that multiple equally useful configurations exist.

  As with the rest, this way of working primarily requires only one person. If problems or errors are encountered the employees move to the co-operative (i.e. have another employee look over the data basis) or even co-constructive level (i.e. change the data basis) in order to reach the goal.

- **Presenting the results**
  In our estimation this is the most dynamic stage. This stage is characterized primarily by shifts from co-operation to co-construction and back. The material that EMD provides acts as information broker between EMD, customers, government and other relevant parties. In deciding which configuration is best, discussing pros and cons they participate in an co-operative endeavor. When or if any participant comes up with new data or new ideas (i.e. one less turbine or different turbine model) - something that requires everyone involved to change their perception of the field of work (reconceptualize the field of work) this requires a move to the co-constructive level, not just co-operation. If changes are required, these changes (might) impact on the entire project, requiring EMD to reiterate to one of the first steps as described in chapter 3.

What we have described is that EMD during an entire project with exception to the last stage relies heavily on the work done by one man. They occasionally use co-operation when problem solving is required along the way. Errors occur and to solve these they use co-construction. However, with exception to the last stage these shifts to co-construction are rare and far between.

The last stage is the largest possible source of shifts between co-operation and co-construction. In a way, the way EMD's work is done now is that their experience is used to reduce the amount of time spent at the co-constructive level in addition to the number of times collaboration reaches co-construction. This is not to say that EMD tries to avoid co-construction, but we believe that EMD does not actively seek the co-constructive level.

This leaves the question of how they actually coordinate their individual efforts. As also described in [JB] and mentioned earlier in this chapter even the most static and well organized organization cannot operate solely on one or the other of the three levels of collaboration, which is why we believe the apparent abundance of co-ordinated behaviors is not indicative of the whole spectrum of events that EMD go through in order to bring a project about up to but not including the last stage where they present their results to customers. We do not see this as a problem as we are not trying to help EMD coordinate their efforts in the first steps, but help them visualize certain aspects of their artifacts.

The last stage where iteration and communication is used to achieve the common goal *is* a possible target for a mock-up tool which the GCVR system is certainly capable of. Using the GCVR system as a mock-up during this stage will enable EMD to give customers and government alike a more dynamic environment in which to express themselves. If it enables users to quickly make different suggestions readily available not only during the presentation, but certainly also during the initial talks at the start of the project the gains will be twofold:

- EMD is able to move to co-construction more rapidly and easily. They can make the transition more times over, that is create more mock-ups. In addition users will be able to visually present their opinion run-time so to speak.
- EMD is also able to utilize the dynamics of co-construction already during the initial stages. This way of working is not used presently. It will enable EMD and other participants to both show ideas up front.

In the following section we will locate the design issues which are consequences of the findings during the past chapter.

5.2.3 Design Implications

From this chapter, and the last sections in particular we can extract the following general characteristics which will have implication on the way we do our design:

- **Interfaces**
  Should EMD want to use additional data - more specifically the data they already have access to through WindPRO for instance, a need arises for the new GCVR system to have interfaces to- or loaders for that data.
- **User interface**
  Not a whole lot of UI functionalities have been identified other than it must give access to an amount of functionality enabling the user to alter basically everything in the world. We can lean on Kjeldskov (see [JK02]) for a design pattern on how to organize the virtual desktop[3]. This will only give an indication on how, not what to organize though.
- **Communication (speech and gesture)**
  For GCVR to be useful for EMD in the way we intend it to be used there is a definite need for verbal communication. The definite need for speech was stated in section 5.2. We see no need for creating an additional interface between collaborating people when they already have speech, which we can readily support through an external speech application. Having to translate ones thoughts to graphical objects (or gestures) and for the target of the communication to reverse the procedure is complicated for the implicated people. To be useful it will require some form of protocol which in the end would only impede communication (see [SS96] or [Sch94]). See section 6.2.6 on a discussion on supporting communication in GCVR.
- **Visualization**
  GCVR must provide a means for visualizations that enables useful transitions to the co-constructive level of collaboration. The need for seamless

[3]Shared space for coordination, seperating shared and physical space, zones in the shared space and shared documents and tools.

transitions was documented in section 5.1 and concluded in section 5.2.2. This implies at least visualization of height contours, turbines and additional objects affecting aesthetics. A thing worth mentioning is that we do not have enough data to make a self conducted theoretical foundation on the visualization of remote users. We can however lean on the work of Kjeldskov (see [JK02]) for instance and his design pattern on mutual focus in interpersonal communication for information on what constitutes a 'good' avatar. We need to show avatars in a way that enables other users to see direction of view and focus of the remote user.

This will have to be tested visually as we have no immediate means of grading this feature.

- **History and save/load**
  For EMD to be able to retrieve what has been agreed upon during the co-constructive sessions or the effects of new ideas surfacing, GCVR needs a save and load feature. The functionality would be much like the history feature of WindPRO and was found to be a valuable addition to the system in section 5.1.

- **Attach information**
  As stated in section 5.1, to be able to move to the co-constructive level of collaboration users need to be able to document the process (the whys and the hows) of how a certain state of the VE came into existence. This could be accomplished by attaching messages to objects in the VE.

- **Edit models on the fly**
  Also as concluded in section 5.1 the system should be able to reflect the changing nature and behavior of objects. As such the system must support changing objects themselves or be able to reflect such changes seamlessly.

The features described above will be discussed in detail in the following chapter.

6 FEATURE DISCUSSION

The purpose of this chapter is to determine the consequences of the theoretical analysis described in chapter 4 and 5. The results of the analysis described in these chapters were that the system should be persistent, passive, describing the state and oriented towards the field of work as described in table 4.6. Also, it should support transitions towards the co-constructive level of collaboration.

We will start out by taking a practical point of view on the combination of the GCVR system and the Virtual Round table analyzed in section 4.6.1. We will describe the basic structure of the combination and the basic features which must be present in order for the system to work as intended. On the basis of this description and the analysis of EMD we will suggest additional features which can support EMD employees in doing their job. Following this, we will have EMD evaluate the features and provide feedback in section 6.3.

Finally, we will discuss which features are necessary or interesting when taking into account the classification of the system performed in chapter 4 and perform a selection based on this classification.

6.1 MERGING THE PARADIGMS

In this section we will describe how the general concepts (the GCVR system and the VRT) described in chapter 4 can be combined from a technical point of view. We will base the combined system on the GCVR system, since it contains the basic properties needed for such a system.

Specifically, we will outline a basic design of the system structure which identifies the general components of the system and a network topology which makes it possible. The final selection of features will be based upon the results of a presentation of the outline and a discussion of user-needs in this context.

Basically, the system must provide interaction in a shared virtual world using a combination of three basic scenarios - the round table, single-user and multi-user as described in section 1.1. We are in possession of three components which can be helpful in achieving this goal:

- The GCVR system is able to create a representation of the shared virtual world, distribute it to clients and keep it consistent throughout the network. Also, it is able to visualize the virtual world and handle user input.
- A vision tracking system which is able to determine the position and orientation of physical objects with a certain well-defined appearance.
- A component which is able to track the head positions of the users. This information is crucial for coordinating the viewpoints of the users when looking at the same virtual world from different angles.

At this point, we suggest running the vision and head tracking systems as servers separate from the GCVR server since they currently run on different operating systems and at least the vision tracking system is very dependent on CPU resources.

A possible structure of a system with the capabilities described is outlined in figure 6.1.

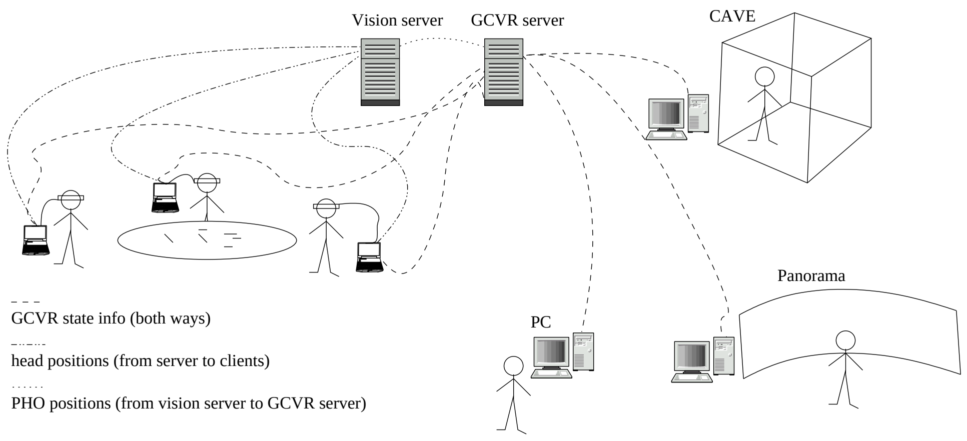
Vision server
GCVR server
CAVE
Panorama
PC
GCVR state info (both ways)
head positions (from server to clients)
PHO positions (from vision server to GCVR server)

Figure 6.1: *A possible system structure.*

The users in the round table subscenario will each be wearing a HMD which is connected to a personal computer - a laptop, for instance. The client programs of these users need two types of information: PHO data and head positions. The head tracker is connected to the vision server which transmits the head positions to the round table users using an unreliable protocol[1]. The head tracker information should mainly be transmitted on the local network of the round table users, since this is where it is most important. However, other users may be interested in seeing head position and direction of view of the round table users, as described in section 5.2.3. Therefore, it should be sampled at some rate and transmitted to the users in the single- and multi-user scenarios.

The main purpose of the vision server is to track the position and orientation of the PHOs (indicated as small line segments on the legless table of figure 6.1). This information is needed by all clients in the system. Therefore, it should be transmitted directly to the GCVR server which ensures that all clients have an updated version of the state of the virtual world. It is not imperative that the vision tracking information is correct at all times. However, it is important that all users see the same virtual world. Therefore the GCVR server could relay the data stream to all users of the system and only sample the latest changes of each 3D model at some rate (i.e. 10Hz) and then implement that change into its own instance of the virtual world and using a TCP connection to ensure that all clients have a synchronized abstract world model. In that way all users can receive as much state information as their network connection allows. In addition, the system-wide consistency can be kept as good as possible. In both the head and vision tracking cases, the unreliable datagram protocol (UDP) is a good candidate for transmitting the data.

[1]An unreliable protocol does not ensure that packages arrive at their destination. However, unreliable protocols ensure that if a package is received, its contents are valid. The packages sent using unreliable protocols generally use less space, since their headers are smaller, and therefore an unreliable protocol allows more packages to be sent. In the case of this particular part of the GCVR system, it is not crucial that all head and vision tracking information arrives, but it is important that as many packages as possible are received with as little delay as possible. Therefore, we choose an unreliable protocol.

6.1.1 Basic Design Tasks

In this section, the basic design tasks which must be completed in order to incorporate the VRT into the GCVR system will be identified. When these features have been implemented, a functional system is available, which supports both the VRT and GCVR approaches towards collaboration and on top of which a specialized system can be constructed. The needed features include:

- Incorporation of the vision tracking system: This will provide a basic interaction method which can be used in the round table scenario. It should work as follows: When a user moves an object on the table, a line perpendicular to the table going through the position of that object is generated. The 3D model closest to the object being moved which is also on the line will follow the movements of the object. The approach is illustrated in figure 6.2. We hope that this will be the preferred interaction method, but further testing is required in order to evaluate it.

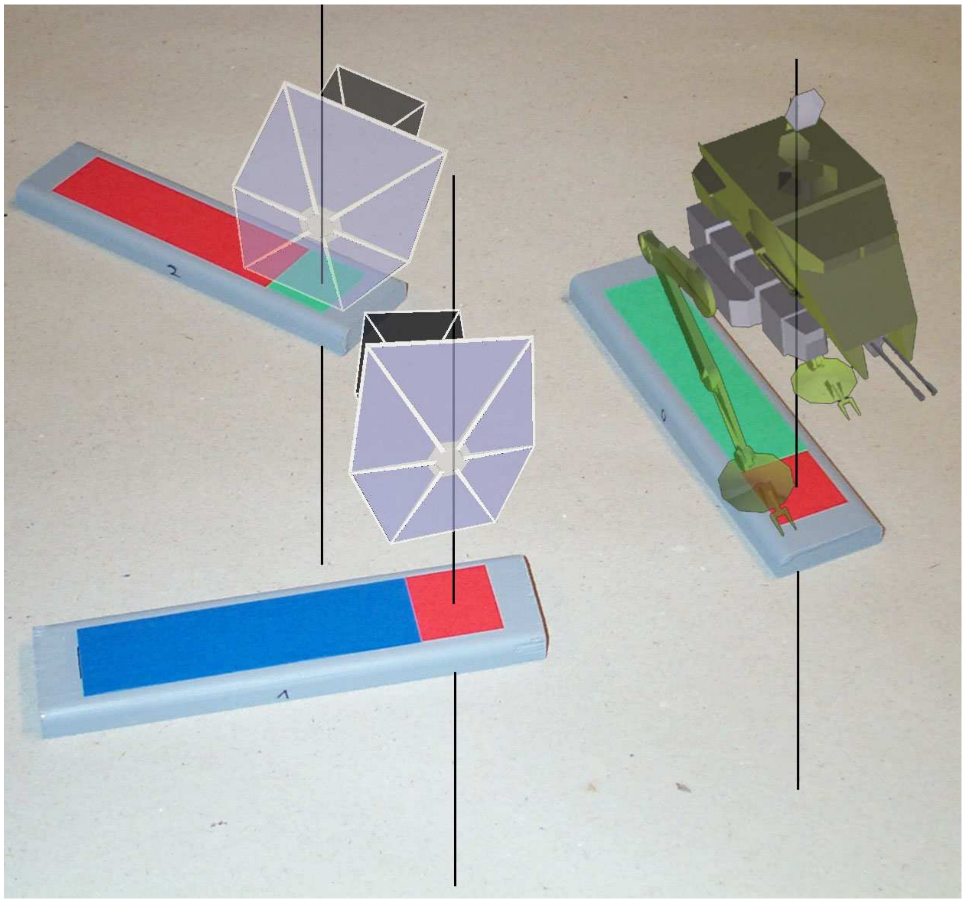

Figure 6.2: *The PHO interaction method. The PHOs (the grey wooden pieces with colors on top) are associated with 3D models by determining which objects are intersected by a line going through each place holder.*

- Mapping of placeholder states to the states of the 3D models that they represent: Some scheme must be developed which enables a number of PHOs to be mapped into the set of 3D models in a useful way.

- Coordination of the real world coordinates and the coordinate system of the virtual world: Some procedure for coordinating the relative movements of PHOs with the relative movements of the 3D models in the virtual world is necessary because PHOs and 3D models reside in different coordinate systems. This also applies to a users head position in the real world relative to the virtual world.

- Implementation of a simple pointing device interaction scheme: The mouse interaction scheme will be a basic input method for the scenario where the user is located at an ordinary PC. By positioning his mouse, a line will be generated automatically in the system and the 3D model closest to the user's position which is also located on the line will follow the movements of the mouse. Interaction facilities for performing changes on other attributes of 3D models supported by the communication protocol (scaling and orientation) will be described later. For the CAVE and Panorama arenas the pointing device would be a wanda.

- Development of a heterogeneous network topology: A heterogeneous network topology allows the vision tracking system to be located on a dedicated computer system and transmit changes of the state of the real world to the GCVR server (which maps the changes into the AWM) through a network. Also, it will enable us to use a head-tracking system running on a dedicated system for coordinating the viewpoints of the round table users with the physical world.

- Utilization of headmounted stereo goggles: This step is necessary in order to create the illusion of a spatial 3D stereo projection in the round table scenario.

Also, 3D models which represent the items needed for a specific session of collaborative work, must be constructed.

When these tasks have been completed, we are able to present a working system (as described in section 4.6.1) to users who will then be able to comment on the existing implementation. Also, they will be able to suggest new features, which will enable them to do their specific tasks.

6.2 Advanced Features

We have already described movement and interaction in the virtual environment at great length in the first report on GCVR [JUV02]. In this section, we will describe some features which seem interesting in general or have come up during our research of EMD.

6.2.1 Movement in the Virtual Environment

We will focus our attention on three patterns of movement: The rollercoaster metaphor for automatic movement, flying and teleporting. All three of these were also described in [JUV02].

- **Rollercoaster**
  The rollercoaster metaphor will be most applicable during the last phases of a project in which most (if not all) of the world has been built and all pertinent objects are present. Preferably also after any animation has been activated. This could be used when i.e. EMD wants to show ’the big picture’ to a large group of people in a panorama-like environment. The controller of the rollercoaster is thus free to speak and use his hands while the computer makes sure that everybody sees what needs to be seen.

- **Flying**
  Flying entails users to mark a point (or object) in space and then over some time frame be moved towards that point (or object). This is basically to make it easier on users to move great distances. An addition to this flight metaphor could be to also select a focus - as the avatar flies through the environment, it will at all times look straight at what the user selected to be in focus.

- **Teleporting**
  Teleporting provides an alternate way of movement. A more complicated and perhaps useful teleport could be the one mentioned in [JUV02], in which we had the idea that a teleport could be used to travel between worlds. A twist of this interpretation would be to teleport between different instances of the history. Making a gradual teleport could be used to flick between two possible configurations, in a way overlaying one history instance over another. Thus teleporting can be movement in the both time and space. The possibility to move between previous states (movement in time) of the virtual world makes the system more process oriented because it is possible to investigate the work process at a later point in time. In order to make teleporting possible, a logging system could be designed, which records and timestamps all changes made in the virtual world by all users.

As for the usability of the first two methods of movement they will be most usable in a setting where the arena is either the CAVE, the panorama or a standard PC - in general everywhere the mortal mode (see chapter 1 for a description) is chosen. In deity mode you don’t fly through the world, you observe it from above. This fits better with the semi-transparent HMDs applied at the round table.

6.2.2 Interaction in the Virtual Environment

From [JUV02] we also have two functionalities that will be required according to our knowledge of EMD: Selection of objects (including mass selection) and the ability to change turbine model on-the-fly. Additionally we have a large area pertaining to changing the total set of attributes of the objects and data we have mentioned in this chapter. This functionality is aimed at manipulating the virtual world, and ultimately the physical field of work represented by the virtual world.

- **Selection of objects**
  Users must be able to move objects around. In WindPRO, EMD are able to select multiple objects. This is very usable when wanting to move multiple turbines all at once. There are basically two methods for interactively selecting multiple objects - either select the objects individually or mark an area as described in [JUV02]. Another method would be to specify a set of

attributes (i.e. all turbines lower than 30 meters), from which a result set would be generated. This set of objects could then be moved collectively. The two first mentioned methods will probably be the most intuitive as these are heavily used in operating systems today. The last method is more relevant in the single- and multi-user scenarios. As already described, users in the VRT environment will be able to select objects by taking hold of a PHO.

- **Change model**
  The ability to change turbine model on-the-fly will also be very useful for EMD. EMD must be able to change the model - different customers might have a preference towards different turbine models. The ability to shift between these models enables EMD to show the aesthetic impact on the environment immediately. If the system had access to the formulae for energy production it would be possible to also show gain and loss between different models. This will most likely be used during the initial phases, when the discussion turns to model selection. We will not take this feature into further consideration due to its context specific nature.

- **Change attributes**
  From what we have seen at EMD it is specifically the ability to move turbines relative to the ground level that will be useful. Additionally, if height contours are used, the need for movement straight up or down also shows itself - the bottom of the turbine should be on the ground, not under it. This placement is not something on which there is much to debate, so this should preferably happen automatically as users move turbines in the plane. As for direction and orientation, the user should not be allowed to change these - they should be adjusted automatically. These variables can only have one state, namely that which shows the turbine as protruding straight up from the ground.

6.2.3 Deity and Mortal Mode

As mentioned in chapter 1, it is important that users are able to experience the virtual world from two different perspectives. One perspective should enable users (mainly in the round table scenario) to interact with the virtual world in the same way as when one moves around objects on a table or reads a map. This is the deity mode.

The other perspective - mortal mode - is interesting when evaluating a virtual world from the viewpoint of those people who inhabit the physical world represented by the virtual one.

The deity mode enables general planning to be done, whereas mortal mode enables users to evaluate it from a natural perpective. In the single- and multi-user scenarios it should be possible to switch from mortal to deity mode, but in the round table scenario, the main mode of interaction will be deity mode because of the semi-transparent HMDs. By adjusting the transparency and interpret the head tracking data differently, it may be possible to support mortal mode for users who are mainly in the round table scenario.

6.2.4 Environment Features

In this section we will describe a set of features which contribute to the realism and therefore in some cases to the usefulness of a visualization. In general, they improve the realism of the virtual world, which in turn makes the representation of the field of work more complete.

Weather

The weather effects will improve the quality of a visualization because they enable the user to evaluate the aesthetics of a scenario under different weather conditions. The weather effects in question are:

- Rain, snow, etc.: An animated visualization of these effects could be included.
- Fog, clouds: An animated visualization of clouds moving at a rate determined by the velocity of the wind.
- Wind: In a very realistic scenario it could be possible to change the direction, velocity and other wind properties and let it have effect on the virtual world. For instance, the number of revolutions performed by a wind turbine could be visualized on the basis the velocity of the wind and the friction between the air and the terrain around the particular wind turbine.
- Season: A feature which enables the user to view a landscape at different times of the year could be useful when discussing or experimenting with for instance the color of objects being placed.

Common for these factors is that questions regarding the aesthetics of objects being placed in a landscape under different weather conditions could be discussed by using a concrete example as opposed to the pictures appearing in the minds of different people.

Lighting

Realistic lighting effects and the shadows which are a consequence of them could be of some importance in the EMD case. For instance, imagine being able to position the sun relative to the time of year and day. A realistic simulation would predict where the shadow of a wind turbine would fall. This would be useful in a negotiation where a wind turbine was to be placed near a place where people live (if other issues such as sound are of less importance).

Terrain

The generation of terrain visualizations which are consistent with the terrain which they represent are crucial in a system in which decisions are based on the placement of objects in those landscapes. Therefore, in a specialized system, this feature or some equivalent procedure must be present. One could imagine a terrain generator which generates a 3D visualization on the basis of a map and the height contours indicated on it. The terrain generator ensures that the virtual world contains a well-defined representation of the field of work.

Another issue which comes up in this context is the issue of auxiliary objects. These include objects which are present in an environment but which are not necessarily tangible. Examples are trees, bushes, houses, lakes, parking lots, roads, etc. When evaluating the placement of a wind turbine, it may often be interesting to perform the evaluation in the context of the location of a house owned by a certain participant. Also, this participant might also want to know whether or not he would be able to see the wind turbines despite the fact that a group of trees is located in the line of sight.

This requires a tool which is able to generate realistic objects for creating custom made auxiliary objects. Also, it would be necessary to support realistic texturing of these auxiliary objects in addition to texturing of the terrain itself.

ANIMATION AND INHABITANTS

In order to provide support for rain and snow and automatic movement of the wings of a wind turbine, some scheme for supporting animation is required. In the current form of GCVR, this would be possible by loading a 3D model for each drop of rain. This is not a viable approach due to the immense overhead which would be the result of the current architecture. In the case of rotating wings of wind turbines, it would be possible to load the tower of the wind turbine and the wings separately and then use a timer for updating the rotation of the wings using the GCVR API. This, however, would also not be a viable approach because (unless a grouping scheme is developed) movement of a single turbine would require the users to move two objects and place them very precisely together at their destination. Another solution could be to use "flip-book" animation, switching between fixed models of a turbine having its wings in various positions. However, this approach would not take advantage of the scene graph, which is used in the GCVR system. The conclusion is that animating objects with moving parts requires a new scheme to be developed, whereas movement of entire objects is very well supported already.

When discussing animation, the idea of supporting inhabitants in the virtual world comes to mind. It would be possible to support traffic and animals in the virtual world, increasing the realism even more. The next step would be to equip these critters with artificial intelligence, thereby increasing the realism of traffic or the movements of flocks of animals.

SNAP-TO-LANDSCAPE

Supporting a feature which makes objects snap to the surface of a landscape would make the placement of objects easier in a two-dimensional scenario where all objects are placed on the terrain. However, this particular feature may not be especially useful in scenarios where this is not the case. For instance, if one was to explain how the internal parts of a wind turbine are assembled, gravity would not necessarily be an advantage.

BACKGROUND NOISE

Yet another set of features which would contribute to the usefulness are those related to background noise. In the EMD context, 3D sound could be supported in order to give the users an impression of the noise level at different locations in the terrain. Another way of achieving this, which may lead to more objective observations would be to visualize the noise levels. One way of doing this is to show noise lines on the terrain. Another method which utilizes the medium more effectively is to use semi-transparent spheres or series of them located inside each other which are located with center in the source of the noise. In the wind turbine example, the center of noise would be somewhere near the center of the wings. The noise level could then be visualized by generating the semi-transparent sphere in such a way that the color intensity increases proportionally with the intensity of the noise. That way, the sphere would be very colored near the wind turbine and very faint far away from it. People would then be able to see intensity of the noise they would have to live with in case the wind turbines were placed at that particular location.

INFORMATION OVERLAY

In situations where precise figures, messages using human language or information not being a part of the virtual world is necessary, one could imagine an information layer between the user and the virtual world. In the case of wind turbines, the information overlay could display a number for each turbine indicating the predicted power production at the current location. Using the coordination mechanism terminology, this makes the system more active because the system autonomously performs the updates. A Post-It-like strategy (see section 5.2.3) can be useful during both co-ordination and co-operation, enabling users to attach messages for each other to the information overlay about what has been done (co-ordination), why is has been done (co-operation) and which aspects need specific attention (co-operation). Finally, each user could color his map according to different indexes (e.g. energy production, avg. wind speed etc.).

6.2.5 ACTIVE LAYER

If the application area of the software is within an area in which rules govern, a layer or module in the system that could contain rules would be an advantage for the users. It can help collaboration by describing simple rules on how to change the state of the common field of work that everybody must adhere to. The rules are in many cases already part of the process and it would only seem natural to implement these rules in the system itself so that it could react on actions that have an impact on a rule. One scenario in which this feature could be desired are architects who could position a new structure in a town and in real time they would be able to be warned that the underground was not strong enough for a building of that weight. Other rules such as maximum height, or minimum distance to neighboring houses etc. would also be able to guide the architects as they position the building. In the wind turbine perspective rules regarding noise, minimum distance to neighboring wind turbines, shadow flickering etc. could also be put into the system making it easier for the consultants and especially clients to place the turbines.

An active layer would thus provide the system with a very usable feature in which actions take place based on input from the user. Even a very inexperienced user would ultimately be guided towards a correct site layout obeying the rules. However it is not in all scenarios in which the active layer is needed and therefore it would be best to make it an optional module in the system so that users in no need of rules can remove these from the system.

If one takes a look at first person shooter games today, such rules are already part of the process in that users hit by a bullet or users who fall a great distance lose health points due to such actions. That is, in first person shooters the rules are static and set from the beginning. In a cooperative system context in which rules are subject to changes it would however be more appropriate to have a module that is able to check whether user defined rules are violated - the user should then have the possibility to alter the levels of which the rule module should act. What is common for the above descriptions is that it is implicitly assumed that the system warns the user. In the following section a short description of the possibilities of providing the user with feedback when a rule is broken.

As the addition of an active layer would give the application the ability to act on its own this layer would move the application towards being active.

Rule Feedback

There are many alternatives as to how to give the user feedback when a rule is broken or a rule is about to be broken. Both audio and visual feedback could be provided and possible combined. An indication of something serious could be anything from a line of text in the command prompt to flashing objects in the virtual environment. If audio and visual feedback is available at the same time a warning could be sounded when the rule is broken after which a visual indication such as color overlay or flashing objects could indicate the area in which the rule was broken. Even force feedback could be applied to give an indication to the user that he has done something wrong and that he must stop and take notice of the system warning. However these are all post operation warnings, if the rule system could also monitor when a rule is just about to be broken the system could visualize this by one of the mentioned possibilities. However this kind of pre-operation warning should be easy to tell from the other more serious warning - a warning that is issued before the rule is broken could e.g. be visualized by coloring the area in slightly more contrast and thus indicate to the user that he is close to breaking the rule.

6.2.6 Speech and 3D Audio Effects

No interpersonal audio communication capabilities have so far been mentioned, so in this section we will introduce the options and tradeoffs regarding audio in the application.

Basically there are two options concerning audio in the application, one is integrating the audio closely within the application and giving virtual objects sounds or otherwise integrating sound into the world. The other is keeping the audio outside of the virtual environment and run it as a separate individual module.

3D Audio

In this case the sound can be used for enhancing the immersive experience in the world, e.g. by giving cars in the virtual world the sound of a car. By associating sounds to objects the users will experience a more lively environment in which e.g. birds are singing in the nature and the wind blowing in the trees can be heard. By integrating sound into the virtual world it is possible to associate the location with a sound so that the intensity decreases the further away from the source the user gets. This type of audio could also be applied to dialogues between co-located avatars. Even though the people controlling the avatars are located far away from each other they could have a private conversations by talking in the part of the virtual world where they are located. This however presents the problem that when a user wants to talk to a person whose avatar is located in the other end of the virtual world he will have to move his own avatar to the area of the virtual world in which the other person’s avatar is located. However it has the advantage that when cooperating on some mutual model and the avatars are gathered around that model they can all hear what is being said by the other avatars (users). Other people whose avatars are located at a different location in the virtual environment by another model will be able to converse without hearing what the other team is saying. Thus this method of integrating audio minimize the amount of audio information each user hears, and thus the amount of audio information that must be transmitted to each user. A drawback of this approach is that it requires that the system is able to support both audio and graphics. GCVR does not currently have audio capabilities and in order to support audio a 3D sound module would have to be implemented as well as the AWM would have to be expanded so that sound is also an option. If we take a look at the two types of user modes - deity and mortal - 3D audio is not an interesting option when in deity mode, since there is no reason to be able to hear what kind of noise an object makes from above. 3D audio is only interesting in mortal mode in which one can walk around on the ground and hear the sounds.

SPEECH

As introduced in section 5.2, we believe speech is essential during collaborative sessions.

The other option regarding sound is to run an external application that control audio communication. This type of application is the same as what is known in the computer game world. It works basically like an intercom in which all users connected to a given channel can hear everything all other users on that channel utters. This type of audio is in no way suitable for providing the users with an experience of the decibel levels in a given area since it is not linked with the 3D environment. This type of sound is however excellent for supporting speech communication between the users no matter where they are located in the virtual world and not matter whether in deity or mortal mode.

Amongst the most popular voice communication systems for this type of use is Roger Wilco. Roger Wilco is basically just an internet phone. However what makes Roger Wilco different is that is builds on a codec developed for the US Department of Defence in the late 1980's. This codec allows for 25 times compression of the bandwidth usage compared to a non-compressed channel. Therefore even a modem of 14.4 K is able to use the software and chat with others. However a 28.8 K modem is recommended. A newer and open source internet phone is Speak Freely. The Speak Freely project started in 1991. After 1996 the program was released to the open source community and it is still under continuous development leaving it to be a very advanced and solid tool. Speak Freely exists in both a Windows and a Linux/UNIX version which also makes it a very versatile tool. In contrast to Roger Wilco, Speak Freely was developed for not only very low bandwidth communication but also for high quality low bandwidth communication and therefore more different codecs, described in appendix A, are available in Speak Freely.

In table 6.1 a small comparison of the bitrates of the two programs. The values in the *Sound Fidelity* column need a bit of explanation. The Speek Freely team has rated the quality or fidelity of the sound given each codec. They defined a base value of *Poor* as being able to hear what was spoken, but not being able to recognize the voice.

As can be seen the Roger Wilco protocol is a very good performer if one only wants

	Inbound	Outbound	CPU req.	Sound Fidelity
Roger Wilco	2400	4800/7200	Slow	Poor
Speak Freely: No compression	80.000		Slow	Best
Speak Freely: Simple	40.000		Slow	Poor
Speak Freely: ADPCM	40.000		Slow	Good
Speak Freely: Simple+ADPCM	20.000		Slow	Lousy
Speak Freely: GSM	16.500		Fast	Good
Speak Freely: Simple+GSM	8.250		Fast	Lousy
Speak Freely: LPC	6.500		Fast	Depends
Speak Freely: LPC-10	3.460		Very Fast	Okay

Table 6.1: Comparing speech communication APIs. The values are in bits per second. Speak Freely only published the full duplex transfer rates whereas Roger Wilco supplies the transfer rate in both directions. The values and the layout of the table stem from the developer section on http://www.speakfreely.org and the Roger Wilco row has been adapted to fit within this table. For an explanation of the codecs used in Speak Freely the website above should be consulted.

transfer rate. However if the system is to be used for coordination it is important that users can hear what is being said and just at importantly who is saying it. Roger Wilco can only guarantee that it is possible to understand what is being said (and the sound fidelity of Roger Wilco is therefore rated as *Poor*), they say that higher bit rates would be needed if one were to be able to recognize the voices, however they do not state how much more bandwidth would be needed. Therefore it is a better choice for a coordination application, such as the one we are developing to choose one of the protocols in Speak Freely that as a minimum has a sound fidelity rating of *Okay* in table 6.1. Which codec is ultimately chosen will be a tradeoff between quality, CPU usage and bandwidth requirements.

6.3 EMD Feedback

In order to be able to evaluate the different features and modes of interaction we have described in section 6.2 we have had EMD give feedback on each individual feature in reference to relevance and usability. In the following we will sum up this feedback. We will do this by using the same bullet names as in section 6.2 and whenever possible we will mention any modifications provided by EMD.

Let us start with the different modes of movement in the virtual world:

- **Flying**
  As the basic mode of movement nothing about this mode was changed. They agreed that flying should be the default way of movement.

- **Rollercoaster**
  The basic idea as we presented it would be very useful during presentations. EMD suggested a modified version in which users could point out i.e. four cameras which they could return to at any time. One could say that this would enable dynamic snapshots - they are always up to date, presenting the

world as it is. If what these cameras see could be shown close to those working it would enhance co-constructive collaborative work and would drive the application towards being active. The reasoning for the co-constructive aspect is that when iterating or changing the objects users can see multiple different aspects of the same set of objects all the time. We will not pursue this subject any further as it is just an extension to flying and we feel there are other more important features, which we will concentrate on.

- **Teleport**
  As the size of the worlds is often small (1 - 2 km^2) the need for teleports is not large. However one could say that being able to mark points in space with the rollercoaster, the additional effort put into teleporting would be small (not taking into account how it should be visualized).

The next set of functionalities is interaction in the virtual world:

- **Selection of objects, change model and attributes**
  Nothing was said about any of these that led us to choose one method over the other or limit the amount of functionality made available to the user. This functionality basically encompasses everything a user can do in order to interact with in the virtual environment. When EMD was introduced to the idea of an administrator (the only one able to do complex tasks in the virtual world) it was pointed out that the tasks requiring specialized knowledge (how a roughness map is created for instance) need not be restricted to the administrator. Everybody could change these roughness data but to ensure the correct handling of them an administrator could be chosen by the collaborators. As GCVR describes a collaborative environment everybody should be able to do more than just move PHOs around - change model or add a turbine for instance.

The next set of functionalities concerns environment features:

- **Weather and lighting**
  In general these were given one common label - impressive but generally unusable. They would of course make the environment more lively, thus immersing user more into the environment. The benefit would be that they would not so readily throw away the visualization as not being true to life. We are not trying to make a true to life visualization, so these feature would be added later should the incentive be present.

  EMD added that just showing skies moving by would improve the visualization (and immersion). The reason being that if the rotors on the turbines moved one would expect the skies to also move.

- **Terrain**
  Generally one of the most applicable features encountered so far, the terrain generator was a must-have. As the visualization in the system is about aesthetics and how the landscape looks with turbines it is crucial that the terrain is factual.

- **Animation and inhabitants**
  As with weather, animations and inhabitants would improve the immersive

aspect of GCVR. However it would primarily to be able to impress the people present. As such these were also prioritized less.

- **Snap-to-landscape**
  Talking about wind turbines this feature should also be present in the system. It would not make sense to place a turbine 5 meters above ground and as such it should happen automatically. However the system might be used in a context where a helicopter should be visualized. Having it snap-to-landscape would not be logical in all instances. Our conclusion is that it should be a feature in the system but users must be able to turn it on and off per object.

- **Background noise**
  Again users being able to hear birds singing and the wind blowing in the trees would enable users to immerse themselves in the virtual environment better and it could help impress customers. With the same argument as mentioned in the weather bullet, sound has been given a low priority.

  One exception was the ability to visualize sound (noise lines), however not using a three dimensional, semi-transparent sphere. EMD proposed just overlaying these lines on the terrain - that is drawing lines on the surface - a user does not need to know how loud a turbine sounds 200 meters up in the air. One addition would be to indicate increments in decibel (i.e. every 10 dB with a line. This way everybody can see that the set of turbines are far enough away from local inhabitants. In Denmark, for instance, the law states that no turbine may be placed such that the noise inhabitants receive from the turbines exceed 40 dB. EMD can see these 40 dB line using WindPRO.

- **Information overlay**
  This feature can be split into two parts: data indicators (i.e. energy production shown on top of each turbine) and interpersonal messages.

  Data indicators could also be useful however there are many other ways of obtaining this data and since we are most concerned with the aesthetics this too could be down prioritized. Additionally the change in energy production would be negligible within a few hundred square meters.

  Messages would be most useful during sessions where remote users are participating. The ability to attach voice or textual messages to objects or the entire world was seen as being potentially very useful. Two different people working at different hours could attach a message to the world explaining what has been changed and why. As such it was seen as an important feature during distributed settings that involve users working asynchronously.

This last set of bullets describe the feedback received on the sections pertaining to active layer, speech/3D audio and menu control:

- **Active layer**
  The notion of an active layer was also appealing to EMD to some extent. Instead of having to wait until calculations are made on the entire site to see whether or not certain rules have been broken (i.e. someone has placed a turbine on top of a lake or placed a turbine outside the site area) it could be advantageous to have these rules checked run-time. This could only be applied to the most basic of rules as some of them might require extensive

calculations (noise lines) and would thus impede the visualization. It would also require a user to be able to understand the syntax of specifying rules requiring extra knowledge of the user. EMD proposed that the rule feedback only consist of warnings. That is, breaking the rule specifying that turbines cannot be placed outside the site area should not make the turbine impossible to move beyond the area (i.e. creating an invisible barrier), but should show up as a warning color for instance.

- **Speech and 3D audio effects**
  Speech should also be supported in GCVR. Users situated far apart must be able to talk to each other to be able to work together. EMD did not see the need for 3D audio as they expected that collaborating users to work together when in the virtual world. If the users saw the need for private conversation it should be done outside the virtual environment. Given the choice of advanced graphical communication (hand gestures) and speech, speech was chosen as the only viable choice.

As can be read from the above feedback from EMD these items were of immediate interest to EMD - flying and rollercoaster, interaction, terrain generator, snap-to-landscape, information overlay, active layer and speech. According to EMD, implementation of the above mentioned features will result in a system which is capable of supporting their work - both in short and long distance communication scenarios. Of theses features a subset will be selected in chapter 7.

7 Requirement Specification

The requirements for a system which uses the GCVR system for implementing the round table, multi- and single-user scenarios and provide the basic features for it to be usable by EMD will be discussed and then finally determined in this chapter. The requirements will be discussed in sections dedicated to the scenarios of use described in chapter 1 (round table, single and multi user scenarios) and features useful for EMD as described in chapter 6. Then hard- and software requirements which are consequences of using the GCVR system and of supporting the selected scenarios will be described. This chapter will be concluded by a classification of the final system using the coordination mechanism framework described in chapter 4. In the remainder of this report, we will call the new system GCVRT , short for the combination of the GCVR system and round table concept.

7.1 Supporting the Scenarios of Use

In this section we will discuss the implications of supporting the scenarios of use in relation to hard- and software requirements and functionality. The GCVR system is the common basis for all sub-scenarios.

7.1.1 Supporting the Round Table Scenario

Some basic requirements must be fulfilled in order to provide the round table experience in a suitable way. The basic requirement is (as described in section 6.2.3) to create a deity mode of the virtual world in such a way that:

1. Several people are able to see the perspectively correct semi-transparent visualization. The visualization must be perceived as being located on top of a table and each user must have a point of view which is consistent with his position relative to the table as described in section 6.1. The GCVR system was designed to work with 5-10 people or 20 at the highest and this limitation will continue to exist.

2. It must be possible to associate PHOs with virtual objects in such a way that when a user moves an object, the associated virtual object moves proportionally, as described in section 6.1.1.

Regarding item one, some hard- and software requirements can be defined:

- The visualization will be made using semi-transparent HMDs. In order to provide the best experience, they must support active stereo but for a lower quality experience, HMDs which do not support stereo can be used. In the case of active stereo, a suitable graphics card for which active stereo drivers exist must be used.

- In order to enable a perspectively correct visualization, some method for measuring the viewpoint for each user must be available. One such system exists: The InterSense (see appendix B for a description). It works by the means of ultrasound. The InterSense emits the tracked information on a network and therefore some part of our system must be able to pick up this information. The GCVRT system should however be independent of the method for determining head positions. This can be achieved by transmitting the head positions via a network using a well-defined protocol as described in section 6.1.

Regarding item two, another set of hard- and software requirements appear:

- A vision tracking system has already been made. It requires a camera (its type and quality not yet defined, experiments must be conducted) and PHOs of the type shown in figure 6.2 on page 47.
- The vision tracking system runs on the Microsoft Windows 2000 operating system. Therefore, a dedicated system must be set up and a subsystem for transmitting the tracked information over a network must be developed together with a GCVRT subsystem for receiving it.
- It must be possible to change position and orientation of virtual objects using the PHO interaction scheme (described in section 6.1).

The GCVR system originally assumed that the VR-Juggler system would support the OpenSG scene graph which is currently a basic component of the GCVR system. At the time of writing, some test versions of VR-Juggler with OpenSG support are supposedly in a more or less working state, which is not adequate for our purpose. We will not spend more time at this point pursuing the integration of VR-Juggler, OpenSG and the GCVR system, and in the case of supporting the round table scenario this integration is irrelevant. The basic implementation of the GCVR system using GLUT for visualization will be adequate for supporting this scenario. However, the design must be general enough for the system to use VR-Juggler later on.

7.1.2 Supporting the Multi-user Scenario

The original design of the GCVR system already supports the multi-user scenario where multiple users are located in the same room in front of a common arena such as a panorama (using either active, passive or non-stereo visualization). In some cases, a panorama is run by a computer using the SGI Irix operating system. Therefore, the GCVRT system must be portable to this system. The GCVR system was designed to be portable but it has not yet been ported to neither Irix nor Windows. We will support the multi-user scenario on the Linux operating system, but keep the design portable.

In addition to the features already supported, some interaction features enabling users to interact via a pointing device must be implemented to support collaboration. In this scenario support for a wanda must be present. Flying will be the default way of movement in the virtual world in this scenario but as described in section 6.2.3, both the deity and mortal modes of use must be supported.

Also, as mentioned in chapter 5, avatars indicating position and direction of view must be available such that the users in the round table scenario are visible to the users in the single and multi user scenarios.

Regarding the software requirements for the GCVRT system, we will take the same point of view on replacing VR-Juggler with GLUT as in section 7.1.1.

7.1.3 Supporting the Single-user Scenario

The requirements for supporting the single user scenario are almost equivalent to those needed in the multi-user scenario. The system should, however, work well on a standard PC as defined in the GCVR report (see [JUV02]) using a low quality network connection.

The software requirements for the GCVRT system in the single user scenario will also in this case take basis in GLUT instead of VR-Juggler as planned originally.

7.1.4 Supporting EMD

In order to support collaboration in the context of EMD - real-life collaboration, that is, some features which are not directly related to the GCVR system or the round table scenario are needed. These features are:

- Recording a session of work: In order to determine the details of a session of collaborative work at a later occasion, it is necessary to be able to record what has been going on and who did what at any given time during the session. The need for this feature is documented in sections 5.1, 6.2 and 6.3.
- Speech transmission: It is necessary to be able to communicate via a speech transmission system in order to support collaboration between the single and multi user scenarios and the round table scenario. As described in section 5, visual communication alone is inadequate for performing long-distance collaborative work. Therefore, we recommend Speak Freely based on the analysis in section 6.2.6.
- Terrain: In the case of EMD, precise representations of the landscapes involved in a project are of high importance. Therefore, a method for creation of landscape representations is necessary for the system to be usable in practice as described in section 6.3.

The two first requirements listed will contribute to the general usability, whereas the last is a special feature introduced for increasing the usability for EMD. However, the terrain generator may also be useful for other potential users in need of accurate landscape models.

7.2 Software Requirements

In this section, we will sum up the software premises which are the results of the discussions in the previous sections.

- Linux: The project will primarily be aimed at running on the Linux operating system. However, no design decisions will be made, which prevent portability to SGI Irix. Also, with as little as possible effort, the system should be portable to Microsoft Windows.
- GLUT version 3.7: Used for creating windows for showing graphics in and for receiving input from interaction devices.

- OpenSG version 1.0: Contains the scene graph definition, scene manager and renderer.
- The GNU project C and C++ Compiler version 2.95: The final system will compile using this compiler.

Common for the three last items is that they all run on Linux, Irix and Windows. Therefore, no compromises on the fundamental software have been made regarding portability.

7.3 Hardware Requirements

In this section, we will sum up the hardware premises which appear from the discussions in the previous sections.

Requirements for the round table scenario:

- HMDs: We will not specify any particular requirements for the HMDs. The users must decide which properties are appropriate for their specific use - in some case a high-resolution monochrome display may be the best choice and in other cases a low-resolution color display providing stereo visualization may be the only option.
- Head tracking equipment: The GCVRT system must be independent of the exact method for retrieving head positions and orientations. At least two possibilities exist for obtaining this information: Head tracking by utilization of head-mounted stereo cameras or the InterSense.
- One standard PC with or without active stereo vision capability per user.
- Network connection: A standard ethernet LAN bandwidth of at least 10Mbit/s will be assumed.

Additionally, the system will require a camera and PHOs but we will not specify these items any further, since their specifications are dependent on the vision tracking system, which we are not involved in the development of. We will only assume that the vision tracking system delivers position and orientation for those PHOs on the table, which comply with the continuously changing PHO specifications. The PHO specifications can change because the vision tracking system is still in an early development stage.

Requirements for the multi-user scenario:

- Standard PC or SGI computer with or without active stereo vision capability.
- Large screen, panorama or projection facility.
- Network connection: The network quality defined in [JUV02] will be assumed to be present. The bandwidth requirements are dependent on the update frequency but the conclusion was that the clients were able to work on consumer-level network connections such as ADSL, while the server in most cases would have to use a more powerful connection. That is, a connection which provides a total (combined up- and downstream) bandwidth of 440 KB/s.

Requirements for the single-user scenario:

- Standard PC or SGI computer with or without active stereo vision capability
- Network connection: The same assumptions apply as in the multi-user scenario.

7.4 Final Classification

In this section, we will conclude the analysis part by classifying the system using the classification framework introduced in chapter 4, which will be the result of the requirements discussed in the previous sections.

- **Persistent/Non-Persistent**
  The GCVRT system is based on the GCVR system which was classified as being persistent in section 4.4. Therefore, the new system will support persistence too.
- **Active/Passive**
  The GCVRT system is intended to support collaborative work, which makes the interaction of people its primary task. No software layers which make the system active have been listed in the requirements.
- **Work Arrangement/Field of Work**
  The GCVRT system is based on systems and ideas which we have classified as being oriented towards the field of work. However, the recording feature enables users to determine who performed a change at any given time in the session of collaborative work. This orients the system towards the work arrangement. Nevertheless, the main concern of the system is still the field of work.
- **State/Process**
  The GCVR system is state oriented in its foundation. However, the new recording feature enables a user to monitor and analyze the work process at any time. Therefore, while the system is mainly state oriented, process oriented features exist in the GCVRT system. If we were using the redefined process concept described in section 4.7, the GCVRT system could be defined as being clearly process oriented.

The results of the discussion of the items above are summarized in table 7.1.

	The GCVRT system
Pragmatics	+persistent, +passive
Semantics	+field of work, (+work arrangement), +state, (+process)

Table 7.1: The properties of the GCVRT system.

In conclusion, the GCVRT system will be a passive, but persistent system, which is mainly oriented towards the field of work in a state oriented way. The features pulling in other directions (work arrangement and process orientation) are present in order to enhance the usability in practice and are closely related to the recording feature.

Part II

SYSTEM DESIGN

In this part, a design matching the requirements for the system specified in chapter 7 is described. Each module will be illustrated with a UML diagram. After a short presentation of the original design of the GCVR system, the structure of the new system is discussed in chapter 9 and the general system design is determined. The tracking systems needed in order to implement the round table scenario are designed in chapter 10 and in chapter 11 the functionality needed for the GCVRT system to interface with the trackers is designed. Then the design of the selected auxiliary features are described in chapter 12. Finally, in chapter 13 the new server and client modules are assembled from the modules described through chapters 9 to 12.

8 DESIGN APPROACH

The system which we will design in this part of the report takes basis in three areas:

- The GCVR system.
- The round table approach to collaborative work.
- The needs of specific users - in this case EMD.

We will start out by describing the design of the original GCVR system on which we base the final system in this chapter. In the following chapters we will discuss possible system structures and network topologies and determine which design will best support our needs. Then we will describe the design of the modules needed in order to support the demands listed in the requirement specification in chapter 7. Finally, we will combine the original design of the GCVR system with the new modules and present UML diagrams illustrating the combined system - GCVRT . The design, which we will describe, will support all of the scenarios of use (round table, single- and multi-user). However, some arenas (the CAVE, for instance) will not be supported until VRJuggler support for OpenSG has been completed.

8.1 THE DESIGN OF THE ORIGINAL GCVR SYSTEM

The original GCVR system was designed to support collaboration in a common virtual world. The design resulted in the system being split up into two parts - a client and a server program:
(The remainder of this section is almost identical to section 2.1 in the GCVR report (See [JUV02]))

- **Server**
  The server is responsible for keeping a model of the shared virtual world up-to-date and distribute the changes made by one client to all other clients.
- **Client**
  The clients are responsible for passing all local changes of the virtual world to the server and for providing input and output facilities for interaction with the virtual world.

This relationship could be illustrated like in figure 8.1. The meaning of the arrows in figure 8.1 is explained below:

1. The arrows between the input data records[1] in the server and clients respectively, indicate that a client which loads a 3D model must make it available to the server which then distributes the model to all clients.
2. The arrow between the input data records and the local world generator indicates that a model is loaded into the local world generator.

[1]In practice, the input data records will consist of files describing 3D models using common formats like VRML and Alias Wavefront Objects.

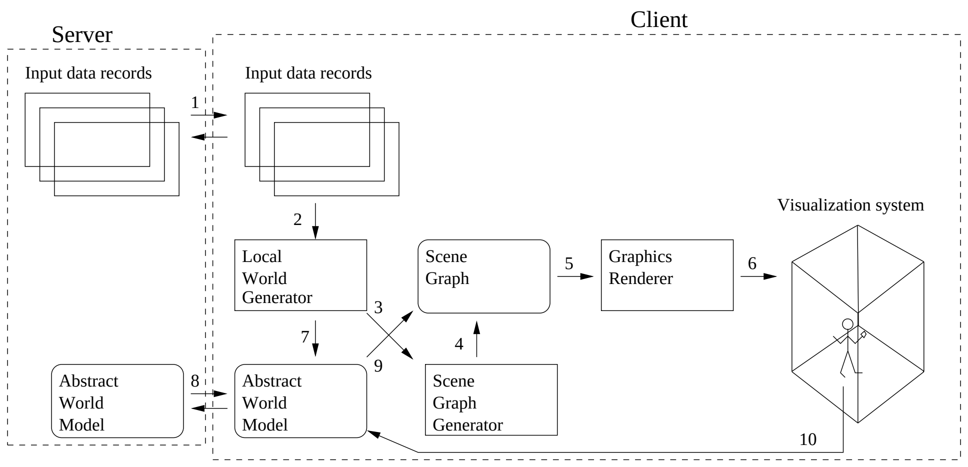

Figure 8.1: *The responsibilities of server and client parts. Internal data formats are indicated with rounded boxes. The abstract world model concept is described in section 2.2 in [JUV02].*

3. Then the scene graph generator in the client module is notified...
4. ... and it expands the existing scene graph with the information loaded from the input data records.
5. The graphics renderer reads the scene graph...
6. ... and the result is shown using some visualization system.
7. When a new 3D object is added to the scene graph, the local world generator adds a pointer to the node in the scene graph where the new information is being placed to the abstract world model. This pointer is associated with an ID which enables us to identify 3D objects across all clients. The abstract world model also contains information about geographic position, scaling, rotation etc. of the 3D model inserted. The details of the abstract world model was elaborated in section 2.2 in [JUV02].
8. This information is communicated to the server which propagates it to all clients. That is, when a user changes the position of a 3D object in the virtual world, the new position is transmitted to the abstract world model in the server. The server then notifies all clients that a change has occurred and they receive the information necessary to keep their local representations of the world up-to-date.
9. When the abstract world model of a client has received new information from the server, this information is used to update the local scene graph.
10. Input from the user is inserted into to the abstract world model and therefore into the scene graph.

In the next two sections, an outline of the internal structure of the client and server programs will be given, since this is the design by which we are going to build the GCVRT system on.

8.1.1 The Original GCVR Server

The modular design of the GCVR server is illustrated in figure 8.2. The system is constructed around the central GCVR server module which is responsible for starting up all other services. The properties of the modules:

- The AbstractWorldModelMgr: Keeps a representation of the virtual world called the abstract world model. This is in essence a vector consisting of pointers to GeoObjects. GeoObjects contain the attributes of each 3D model in the virtual world which can be altered by users. These attributes constitute a state of a 3D model. The attributes are ID, position, scale, orientation and lock.
- The ServerCom: Maintains connections to all clients by the means of the modules: Socket, Client, ServerIdQueue and ServerSocket. The responsibilities of these modules is discussed in [JUV02]. The ServerCom receives and distributes changes to the AWM.
- The IDControl: This module ensures that the IDs which are assigned to the 3D models in the virtual world are unique.

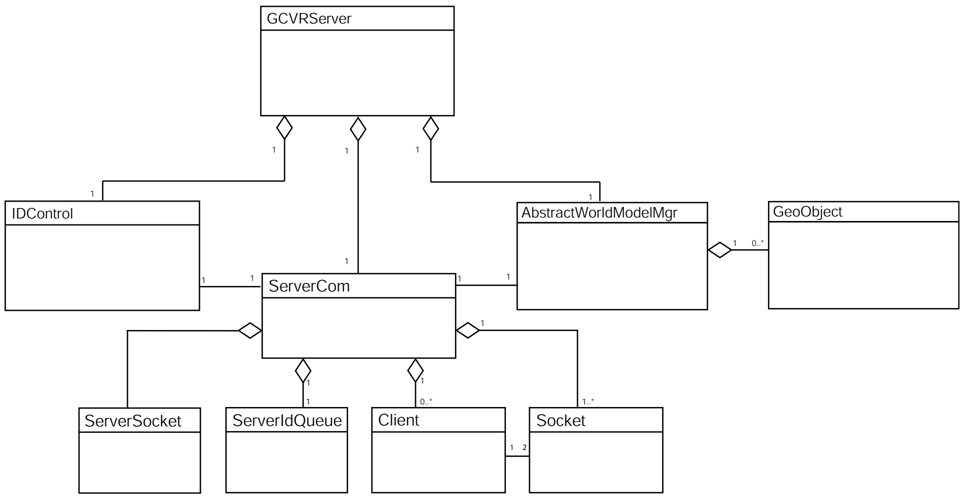

Figure 8.2: *UML diagram of the GCVR server as designed in [JUV02].*

8.1.2 The Original GCVR Client

The modular design of the GCVR client is illustrated in figure 8.3. Like the server, the client is constructed around the central GCVR client module which is responsible for starting up all other services. These include the ones which are shown in the figure as well as starting up a scene manager which is responsible for rendering the scene graph. The properties of the modules:

- The ClientCom: This module is responsible for receiving changes from the server and implement them in the AWM using the AbstractWorldModelMgr module. It uses the ServerSocket, IdQueue and Socket modules to reach this goal. These modules are described thoroughly in [JUV02].

- The AbstractWorldModelMgr: In the client the AbstractWorldModelMgr module is responsible for keeping a local representation of the state of the virtual world. Instead of using objects of the GeoObject type, it uses objects of the GeoObjectOSG type. The GeoObjectOSG objects have the side effect that when the state of an object has been changed, the change is also carried out in the scene graph and hence in the visualization of the virtual world.

- The ChangeController: This module is responsible for receiving input from the input devices and implement them in the local AWM and in the server AWM via the ClientCom module. Since we do not use VRJuggler in this project, the ChangeController, which is tightly coupled with it, has been removed from the class diagram. This is indicated with grey in figure 8.3.

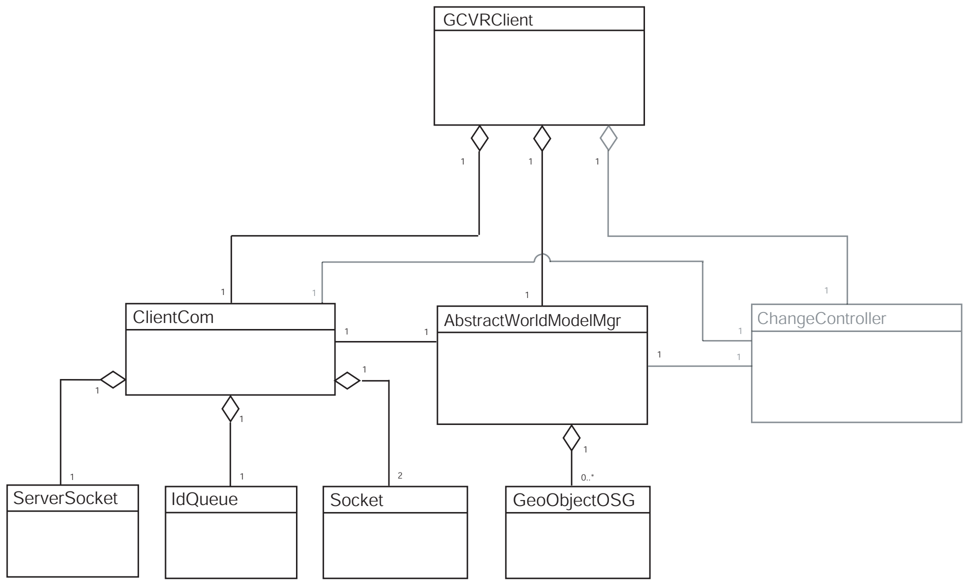

Figure 8.3: *UML diagram of the GCVR client as designed in [JUV02]*

9 SYSTEM STRUCTURE

As defined in chapter 7, one of the objectives of this project is to merge the GCVR system with the round table approach to collaboration. The GCVR system is structured in a client-server topology as described in chapter 8 and consistency throughout the network is considered to be very important. The abstract world model defines the state of the virtual world. On the other hand, in the round table scenario it is natural to perceive the state of the virtual world as being defined as the state of the PHOs being spread out on the table. These two perceptions must be combined in order for this project to succeed.

Since we want users to be able to participate in a session of collaborative work independent of whether they are located at the table or at a PC somewhere else, the state of the virtual world must still be defined by the centralized abstract world model.

As mentioned in the analysis, what differentiates the round table scenario from the single- and multi-user scenarios from a software point of view is:

- The need for tracking the point and direction of view of each user. That is, tracking of the physical position and orientation of the user's head. We will retrieve this information from a head tracking system (the InterSense - see appendix B). The retrieval of this information will be described in detail in section 10.2.
- The need for tracking the position and orientation of the PHOs. This information will be retrieved from a vision tracking server, which will be described in section 10.1. When users move a PHO, the visualization should be updated as fast as possible in order to ensure that the association between the PHOs and the virtual objects appears natural.

When the single-user scenario takes place in a CAVE, head tracking is also necessary. In addition, management of multiple spatially immersive displays is necessary. We will not discuss this any further because this specific variation of the single-user scenario will be possible when the system is integrated with a layer (VRJuggler, for instance), which provides the support. In section 9.1, we will identify the problems, which must be solved in order to incorporate head and vision tracking systems into the new system. On the basis of this, possible general structures of the system will be discussed.

9.1 STRUCTURAL PREMISES

In this section we will discuss the technical problems involved in integrating head- and vision tracking data into the existing system. The integration of the head and vision tracking systems must be completed such that four design principles are not compromised: Synchronized virtual environments, an adequate level of immersiveness, fast updating and modularity.

When discussing the combined system, we will make two distinctions regarding the nature of the networks involved, since they ultimately determine the structure of the system:

- **Local Area Networks (LAN)**: The client programs of the users taking part in a round table session, will have no problem being (and are most likely to be) connected to a LAN. In that way they will have plenty of bandwidth locally and there will be no latency caused by routing.

- **Wide Area Networks (WAN)**: In order to support long-range collaboration (mainly for the single and multi user scenarios), the remote users must be connected to the server via a WAN. In this case the bandwidth consumption limitations described in [JUV02] apply.

There are three fundamental problems, which need to be considered:

- **Mapping**: A mapping between PHOs and virtual objects must be performed. The mapping is performed by calculating which virtual object is located above or beneath the PHO. For this purpose, access to the scene graph is needed because this is the only place where the spatial properties of the virtual objects are described.

- **Perspective**: Each user must see the virtual world from the same perspective as he does in the real world. The scene graph also defines the viewpoint of the user and therefore it is necessary to have access to the scene graph in order to implement the head tracking data. In order to ensure that the visualization of virtual objects stays in the position relative to the PHOs which was intended, the head tracking information must be propagated to the round table clients as fast as possible.

- **Consistency**: The head and vision tracking data are interesting for the round table users as well as the users in the single- and multi-user scenarios and therefore the results of implementing the data must be kept consistent throughout the network.

Regarding the head tracking system, we see only one feasible solution: To send the data directly to the clients. This is due to the circumstance that delays in the update of viewpoints can be very disturbing, so speed is essential. Therefore, there are in general four possible system structures, which satisfy the premises stated above:

- One where the PHO-to-virtual-object mapping is performed on all clients.

- One where it is done on the vision tracking server. We will not discuss this solution any further because we wish to make the GCVRT system as independent of other systems as possible. It will be difficult to replace the vision tracking system with another type of system if we assume that the system has access to the scene graph.

- One where it is done on a dedicated system.

- And finally one where it is done by assigning the job to a single client.

These system structures will be discussed in the next section.

9.1.1 Possible Solutions

One way of designing a system supporting just the round table scenario would be to let the clients perform changes on their local AWM according to head and vision tracking information broadcasted on a LAN. However, this approach does not ensure that the local representations of the virtual world are consistent throughout the network because some computers in the setup may be faster than others. Some synchronization is required.

This problem can be solved by introducing the GCVR server into the system. In order to avoid redundant network traffic, one of the clients could be selected as master for the round table clients. The master would sample the broadcasted traffic at some frequency (say 20 Hz) and relay this information to the GCVR server which would then update and correct the other clients. Introducing the GCVR server will allow remote users to participate. The structure is illustrated in figure 9.1.

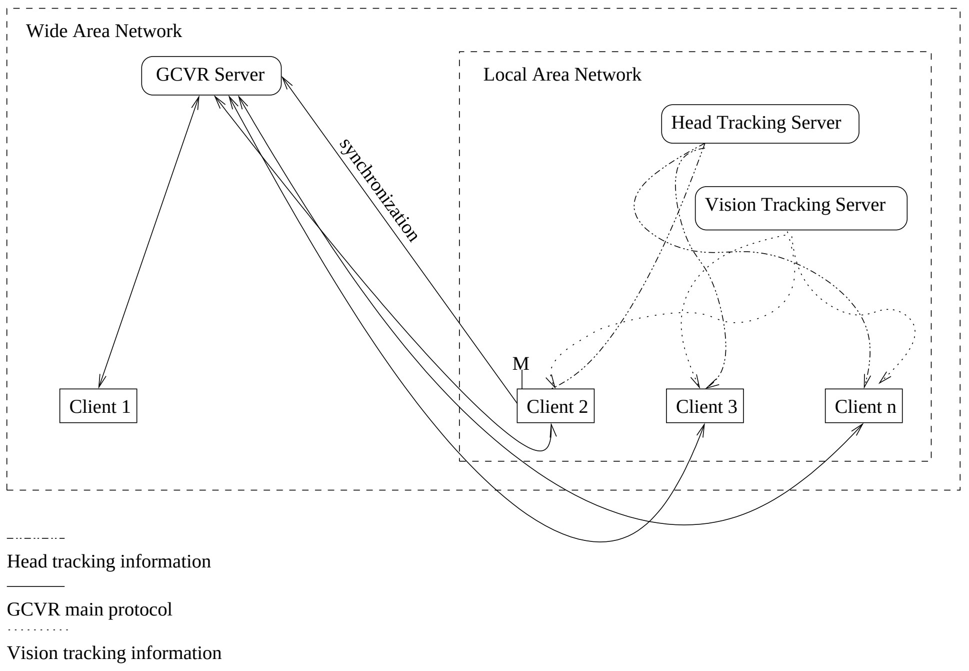

Figure 9.1: *A possible structure of the GCVR system supporting the round table scenario using a synchronization approach.*

The disadvantage of the approach taken in figure 9.1 is that changes in a visualization are allowed to happen even though they may have to be corrected later. Also, all clients must calculate PHO mappings even though some may be faster than others. If a fast client bases its mapping calculation on the latest information in its message queue and a slow client bases its calculation on a later message, the result may be that two different objects are being changed. Consider the event flow in figure 9.2. At T1, both a fast and a slow computer receive a message indicating that a new mapping should be calculated. At T2, a message describing the position of the PHO is received by both computers. At T3, the fast computer maps PHO 1 to VO 5, using the position received at T2. The slow computer has still not been able to process the initialize message received at T1. At T4, the fast computer acquires a lock on VO 5, while both computers receive a new position of the PHO. At T5, the slow computer has finished processing the initialize message and starts mapping the PHO to a VO. On the slow computer, the PHO is now mapped to the VO positioned above the new position of the PHO, which is VO 3. The slow computer then acquires a lock on it. Now, when moving PHO 1, both VO 3 and 5 will move. This scenario can happen if the buffers match those of the original GCVR design.

	Fast	Slow
T1	Init PHO 1	Init PHO 1
T2	Pos PHO 1	Pos PHO 1
T3	Mapping PHO 1 to VO 5	
T4	Lock VO 5	Pos PHO 1
T5		Mapping PHO 1 to VO 3
T6		Lock VO 3

Figure 9.2: *Event flow for a fast and a slow computer performing a mapping calculation.*

Another system structure could solve this problem. A centralized mapping between PHOs and virtual objects could be created. It would be natural to perform this mapping on the GCVR server. However, it is necessary to have access to the scene graph in order to calculate the association between PHOs and virtual objects. One solution would be to add a scene graph to the server, but this would have some consequences: The system requirements of the server become higher and the server becomes less portable.

Therefore one could introduce a round table master as illustrated in figure 9.3. The RTMaster would have access to the scene graph and the broadcasted head and vision tracking information. But this requires a dedicated system especially designed for its purpose. Also, it introduces some network latency compared with the approach taken in figure 9.1, since all changes must go through the server.

For the sake of simplicity in the design and use of the system, we propose a system structure which takes into consideration both the need for consistent representations of the virtual world and the need for fast updates of the visualization. It is crucial that the positions of virtual objects relative to the PHOs remains the same despite the fact that users may change their direction and point of view rapidly. Since the remote clients do not need this information - at least not at the same frequency - we will let the clients involved in the round table scenario get the updates of the head positions directly from the head tracking system. The PHO positions and orientations, however, must be mapped to virtual objects and the changes performed on them must be consistent throughout the network. Therefore, the PHO changes will be calculated by one of the clients, which is the designated master and propagated to the server, and from the server to all clients of the system. Then, and only then, the changes will be visible. This system structure, which is the one we

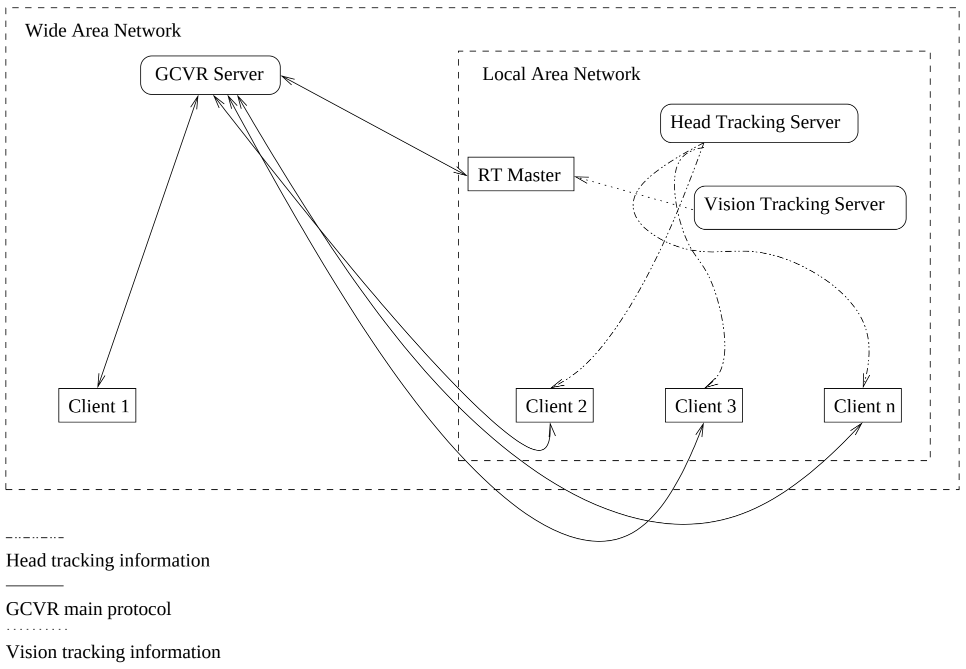

Figure 9.3: *A possible structure of the GCVR system supporting the round table scenario using a dedicated master.*

select to base the system on, is illustrated in figure 9.4. Using this strategy it is also possible to assign a master client to act as the RTMaster in figure 9.3.

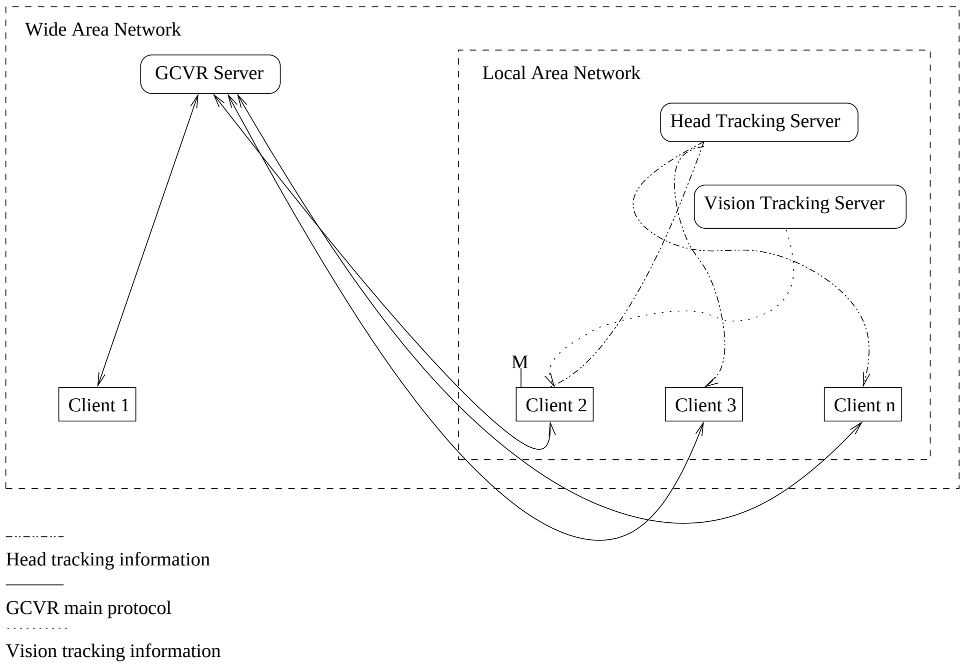

Figure 9.4: *The selected structure of the GCVR system supporting the round table scenario.*

This system design satisfies the demands we mentioned in section 9.1 even though there is more network latency than in the first suggestion. But regarding manipulation of virtual objects, we choose to prioritize consistency higher than fast updates. From the point of view of the GCVR server all instances of the virtual environment are kept synchronized, since an client is just another GCVR client. The mapping is kept in only one place and as such it is easy to keep updated and no synchronization of different mappings needs to be applied. clients are given head tracking data as fast as possible and the general structure keeps vision and head tracking as separate from the GCVR system as possible. Additionally, modularity of the entire system is preserved.

9.2 System Overview

In this section the general data flow in the selected system structure will be described using the same type of diagram as in section 8.1. The first 9 steps are identical with those in the description of the original GCVR system in figure 8.1 on page 70. The last steps in figure 9.5 are explained in the following list:

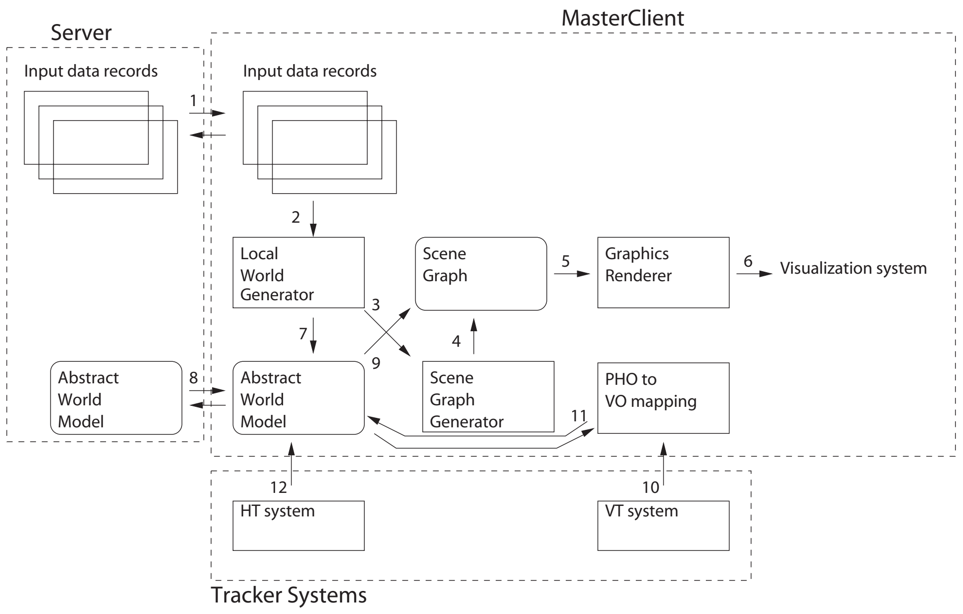

Figure 9.5: *The responsibilities of the different parts of the system. Internal data formats are indicated with rounded boxes. The abstract world model concept was described in [JUV02].*

10. The vision tacking system (VT) continuously tracks the PHOs on the table. When a placeholder is made visible to the camera, the VT system sends an initializing message.

11. The PHO to virtual object (VO) mapping module queries the scene graph through the AWM in order to find out which virtual object corresponds to the position of the PHO in the virtual world. The ID of the virtual object is obtained from the scene graph. After the PHO to VO mapping module has obtained the ID of the virtual object the PHO is mapped to, all changes

received for that PHO are translated into changes on the virtual object sent directly to the AWM.

12. The head tracking system (HT) continuously sends the position of the head of the user. This position is used to set the viewpoint for the user in the virtual world, as well as setting the position of the users avatar in the AWM.

In the following chapters we will design the system so that it reflects the overall structure in figure 9.5.

10 TRACKING SYSTEMS

This chapter concerns the design of the tracking systems needed in order for round table users to be able to collaborate in the virtual environment. The two systems provide support for vision and head tracking and run on the Windows platform. In the following two sections, we will describe how the systems retrieve and transmit data. In chapter 11, we will describe how the information is received and used.

We foresee that the vision and head tracking systems will need a module that can send messages over the network. We will make a module, containing a queue in addition to the functionality needed to add messages to and remove messages from this queue. In addition this module should be able to execute concurrently with normal system execution - this will help reduce the impact on the existing system in reference to what needs to be altered in the that system.

To enable this functionality a set of global, static functions can be created. Basically this set will consist of two public methods, `enqueue` and `dequeue`. When the enqueue method is called with a message as argument this method must put the message in a queue. We can then have a thread read this data using the `dequeue` method and send it to computers on the LAN. This functionality will be placed in a module and we will call it `WsUDPSnd`.

There are two possible ways to transmit UDP data over a LAN or WAN. We will not comment on multicast, as it is a form of broadcast.

- **Broadcast**
  The header specifies that every machine on a network is allowed to receive and process this message. The sender need not know any details on who receives the packages.
- **Unicast**
  One message must be sent to each client. This implies that the sender must know the receiver.

We will use broadcast instead of unicast because then the HTServer does not need information about which clients are listening. Also it is simpler on the clients since all they need to do is listen on a socket, and it is faster due to smaller packages.

The last thing which needs attention is that users should have the freedom to choose the host and port on which the vision and head tracking systems run. We suggest using a configuration file for this purpose which is read at startup.

10.1 VISION TRACKING

To be able to use PHOs as physical interfaces to virtual objects, the GCVRT system needs an interface to the vision tracking system, which is able to track and identify these physical objects. This vision tracking system currently runs on the Windows platform. PHO position and orientation needs to be sampled at an appropriate rate and sent to the master client. In short the vision tracking system must:

- Sample the position and orientation of the PHOs. A filtering scheme can be constructed so that the sample rate can be adjusted without stopping the application.

- Send state information to the master client. This client must be able to map between physical objects and virtual objects, so that the idea of physical objects stays within the virtual round table scenario.

First of all, we need to sample the position and orientation of the PHOs. At the time of writing this data is already gathered at a specific point in the vision system. We will use the transmission scheme mentioned in the beginning of this chapter for sending the data to the clients.

The vision tracking system data, which must be sent over the network must enable the following three distinctions to be made:

- **Grab**
  The vision server does not have access to the AWM so it must send the identifier of the PHO along with enough information to calculate which virtual object this PHO points at in the virtual world. In the master client this can be accomplished by using the data in the change message.

- **Release**
  When the PHO becomes invisible to the vision system (i.e. covered by a hand or removed from the table) the mapping to the virtual object must be removed, in effect unlocking the object. The only thing needed by the master client is then the identifier of the PHO.

- **PHO state**
  As with the GCVR system the information needed in a change message is position, orientation and roll in addition to the PHO identifier.

The grab and release distinctions above can be implemented in such a way that they do not impose any changes to the protocol and maybe more importantly a minimum of changes needs to be made to the vision system. Whenever a PHO is introduced into the area tracked by the vision tracking system, the vision tracking system generates an initializing state message, having the format shown in figure 10.1.

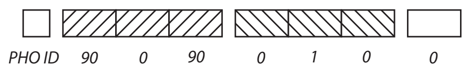

Figure 10.1: *The format of the message that the vision system transmits to the master client when a new PHO is detected by the system.*

The idea of describing an event using a format (see figure 10.2) which also describes another type of events (changes to PHO state) can be perceived as an error in the vision tracking system. The error lies in the fact, that the configuration in figure 10.1 can occur in reality. That is, an object can be positioned in (90, 0, 90) and oriented in direction (1, 0, 1). Though other and better solutions exist for describing an initialized PHO, we will use the existing one, since we discovered this too late for us to have any influence on the implementation. Therefore we will use this message, since it is the only way, initialization of a PHO can be detected. One solution to the problem could be to use negative PHO IDs in the message from figure 10.1 for indicating when a PHO is initialized. A better solution to this problem would be to send a negative PHO ID but also include the exact position, orientation and angle of the PHO, instead of using default values.

When the master client receives an initializing message, it must perceive this as a grab message and consequently try to map the PHO with a virtual object. Note that the state message and the initializing message are much alike. It is also worth to note that the GCVRT system uses roll to indicate a rotation about the directional vector - the vision system uses angle to indicate rotation about the normal of an object.

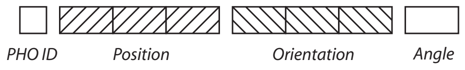

Figure 10.2: *The format of the state message that the vision system must transmit to the master client.*

When the master client receives this message, it must perceive it as a change message and consequently make the appropriate change to the mapped virtual object available to the server. No matter which message the vision system needs to send it must use the `enqueue` method of the methods described in the beginning of this chapter.

10.1.1 Filtering

Depending on which network connection users have, how powerful their PCs are and the quality of the camera, users might want more or less state information. If for instance it is a poor camera, thresholding must be applied in order for the virtual object mapped to a PHO not to jitter. Additionally if the clients are not capable of managing the entire amount of messages there is a need for being able to reduce the amount of messages received over time, so that the clients can be kept up-to-date with the most recent data all the time.

To accommodate this we propose two filtering schemes in which users or systems administrators can adjust the amount of filtering as needed.

- **Thresholding**
  The idea is only to send a description of a PHO state when a PHO has moved a certain distance. With a low threshold the inherent (though small) discrepancies in the vision tracking will cause objects to jitter even though the PHO is not moved. A high threshold will require the user to move the PHO a large distance for the appropriate virtual object to move in the virtual environment. Tests will indicate an appropriate value.

  Thresholding can be implemented by always saving the last position, orientation or roll for each PHO that was sent as a change. Each time a PHO is moved, its new position is compared with the old position - if the distance between the two positions is greater than the threshold value a change message should be sent and the new position should be saved.

- **Time interval filtering**

This filtering scheme consists of placing a numerical bound on the amount of messages sent over time. If for instance the vision tracking system sends with 100Hz but must only be allowed to send with 20Hz, this scheme must make sure that only every fifth message is sent. This must only be applied to state messages, not initializing messages.

To be able to execute the scheme there is a need for saving the number of messages sent last time frame (one second for instance) and use this amount to predict which messages to send during next time frame. For a more accurate result one could utilize the average amount for the last few time frames, in effect limiting the impact of peak amounts.

We will only implement the thresholding scheme for now. Putting a numerical bound on the amount of messages will be implemented if tests show that it is required.

We suggest handling thresholding as follows. Users must be able to set a default value in a configuration file, which is read at systems startup, and they must be adjustable at run-time.

10.2 HEAD TRACKING

For users to constantly maintain the correct viewpoint on the virtual world, we will create an interface module for an existing head tracking (HT) system. The HT system receives data from the InterSense (described in appendix B) which it sends to the clients.

We will not design a filtering scheme for the head tracking system, because it is important that the user's view of the virtual world is as synchronized with the real world as possible. As many updates as possible should arrive at each client.

In short, this module must:

- Retrieve and filter the head tracking information.
- Distribute this data to all interested clients.

As with the vision tracking system we wish to keep the impact on the HTServer source code as small as possible since it is an external system. What is needed then is only the handle to the data structure inside the HTServer in which the head positions and head tracker identifier of all users are located.

The HTServer can use the `enqueue` method mentioned in the beginning of this chapter to send state messages to all clients. The format of this state message is almost identical to that used by the vision tracking system. It can be seen in figure 10.3.

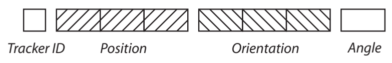

Figure 10.3: *The head tracker message format. The angle represents rotation about the normal.*

11 GCVRT CLIENT FUNCTIONALITY

The GCVRT client must be able to work in the round table, the single- and multi-user scenarios. In the round table scenario, one client must be able to receive and interpret data from the vision tracking system and relay this data to the GCVRT server for processing. Also, in the round table scenario, all clients must be able to receive and implement head tracking information into the scene graph. The latter can also be the case in certain instances of the single- and multi-user scenarios but we will defer the exact design taking this into consideration until a suitable layer (e.g. VRJuggler) can be implemented to handle this. All clients - also those clients outside the round table scenario - should be able to work as a master client. The design of the original GCVR client already allows the functionality needed to support the single- and multi-user scenarios and therefore, we will now focus on the functionality needed in order to support the round table scenario.

The master client is the only participant with knowledge about both PHOs and the AWM. Therefore, it will be the responsibility of the master client to associate PHO states with virtual objects. All clients should have this functionality but only one should be using it actively.

Summing up, what makes the selected master client special in relation to other clients is the following:

- It must be able to receive and interpret messages from the vision tracking system.
- It must maintain a mapping between PHOs and virtual objects.
- It must be able to transmit changes and lock requests based on the vision tracking information to the GCVRT server - and hence the entire system.

The flow of events will be as follows: The master receives a PHO state from the vision server. The master calculates which (if any) virtual object the PHO should be associated with. If a virtual object is found, a mapping between those two should be created. Then the master issues a lock request (using the format described in [JUV02]) to the server. The server can reply in two ways:

- **Deny**
  The master client must delete the mapping.
- **Accept**
  The master now knows which changes to a physical object should be visually manifested in the virtual world and all changes to that object must be relayed to the server, which will distribute the changes.

In both cases, this can be done unambiguously because the reply message will contain a synchronization variable[1] of the corresponding lock message. In the accept case, the PHO states are converted to change messages using the GCVR

[1]The synchronization variable is a part of the GCVR protocol, which ensures that accept or deny messages will be associated with the requests they were meant for.

protocol (described in [JUV02]). The GCVRT system will handle this message just as any other change message and the existing functionality will make sure that every client receives the appropriate changes. In the following sections, we will describe how the PHO states can be associated with virtual objects, how the head tracking information can be implemented into the scene graph and finally a common scheme for receiving and decoding the messages sent from the head and vision tracking systems.

11.1 Mapping PHO States to VO States

Currently, the vision tracking system supports tracking of six different place holders. We expect this number to increase in future versions. In order to maintain the mapping between the PHOs and the VOs, but also handle an increasing number of PHO mappings, we suggest using a data structure which is capable of being expanded without human intervention but which is also easily indexed and relatively fast. An array provides this functionality, except for the ability to be expanded. A vector does support this and this data structure will therefore be a good choice. Consequently, we will use a vector of records containing PHO IDs and the associated VO ID. The records are illustrated in figure 11.1. The vector will be named `map` and it will be placed in a class called Interaction along with its associated functions.

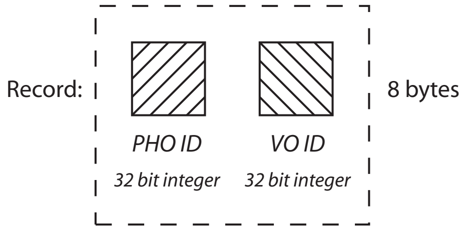

Figure 11.1: *The format of the mapping of PHOs to virtual objects.*

We will now go into detail with how the actual mapping should be performed. A PHO state message is received by the client. This message contains a PHO ID, a position, an orientation and an angle. In order to calculate which object the PHO should be associated with, the position must be used. We are interested in associating the PHO with what appears to be the closest virtual object placed either directly above or directly below the PHO in the physical world.

In the OpenSG API, there is a function called intersect, which is able to detect whether the bounding box of a virtual object is intersected by a line. If the line, which is going through the position of the PHO and is perpendicular to the plane on which the PHO is placed, is given to the intersect function together with a virtual object, the intersect function is able to say whether or not the line and the bounding box of the VO intersect each other. If this is done for all VOs in the AWM, it can be easily determined which VOs are candidates for being associated with the PHO. All that needs to be done is to calculate the distance to each of the candidates and associate the PHO with the object which has the shortest distance to the PHO.

In order to implement this functionality, we will add a function `getPerpendic-`

`ular` to the AbstractWorldModelMgr. This function will generate a line which goes in the direction of the coordinate being 0 in the position of the PHO, and therefore perpendicular to the PHO plane and the round table. In this way, we will get a line perpendicular to the table, independent of differences in coordinate systems. The `getPerpendicular` function will call another function in the AbstractWorldModelMgr called `whichObject`. This function is able to scan the AWM and for each VO, find out whether or not it is intersected by the line, calculate the distance from the PHO to the VO and return the nearest object being intersected by the line, if any such VOs exist. The `whichObject` function can be used for supporting other input devices such as mice or wandas. This takes care of the actual mapping.

The vision tracking system is able to send messages containing the following information:

- The ID of a new PHO, which has been introduced.
- The ID, position, orientation and angle of known PHOs.

We want the users of GCVRT clients to be able to associate a PHO with a VO, remove the association and associate the PHO with a new VO. But the vision tracking system does not provide any information about releasing an association. Therefore, this information must be implied in some way in the existing information. We suggest the following approach for making associations:

1. A user takes a PHO and covers it with his hand.
2. When the user's hand (and PHO) is located over or under the VO he wants to manipulate, he removes his hand and lets the PHO be visible to the camera of the vision tracking system.
3. The master client makes the association and the users are now able to move or turn the VO by moving or turning the PHO (if they do not cover the PHO by accident).
4. When the users wants to associate the PHO with another VO, they cover the PHO and proceed with item two.

One has to remember that more than one PHO can be moved simultaneously. Therefore, simply remembering that the last message received from the vision tracking system was an initialize-PHO message is not enough to determine whether or not a certain PHO should be released or associated. Therefore, another vector of records containing a PHO ID and a variable indicating whether or not the PHO has been initialized, will be needed. When an initialize message arrives, it is possible to check whether or not it should be perceived as a grab or release message. The data structure and associated functions will also be put in the Interaction class, which can be seen in connection with the GCVRT client in figure 11.2.

A VO can be unlocked by covering the PHO with the hand and moving it to an area of the table visible to the camera and not occupied by any VOs.

11.2 Utilizing Head Tracking Information

When a client in a scenario where head-tracking equipment is used has decoded the tracker information, the next step is to move the user's viewpoint. In OpenSG, the user's viewpoint is present in the form of a camera beacon. To move the user's viewpoint, the position of the camera beacon must be moved to the coordinates and orientation specified in the message from the head tracking system. If the user has a presence in the virtual world (an avatar) this too needs to be moved. As for the implementation, the client must implement a thread, which listens for head tracking messages. It is the responsibility of this thread to decode the tracker data and call a function in the client, which implements the new camera position, orientation and angle into the scene graph.

11.3 Receiving Head and Vision Tracking Information

Having described how the head and vision tracking information is used and in chapter 10 how it is collected and transmitted, it appears that the two types of information are similar. We will therefore design a module, which is capable of running a thread, which receives the messages and puts them in a queue from which they can be read by the parts of the program which use them.

Since the vision and head tracking systems use UDP for transmitting the messages, the clients must also open a socket capable of receiving UDP data. We will therefore create a module called ClientComUDP. Two instances of this module can then be invoked - one for receiving vision tracking data and one for head tracking data. The module will be started by the GCVRTClient class, which corresponds to the GCVRClient class in figure 8.3.

The last item to mention is that each client must know which data is meant for the individual client. All the vision tracking system knows is a number identifying each unit. We will make each client aware of which unit that client should receive data from by writing this number in the configuration file.

In figure 11.2, the GCVRT client with the Interaction and ClientComUDP classes attached is illustrated.

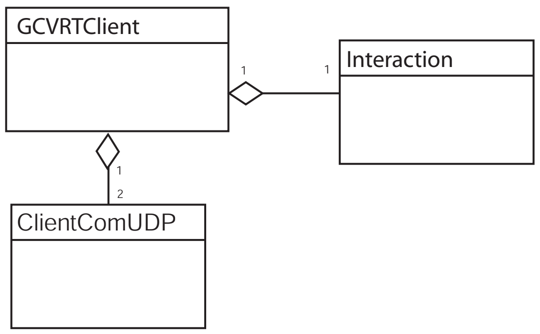

Figure 11.2: *UML diagram of the communication module of the master client. The rest of the client modules has been left out for simplicity.*

12 Auxiliary Features

In this chapter, the selected auxiliary features of the system will be designed. These features include calibration of the coordinate systems, landscape generation, recording the collaborative sessions and distribution of 3D models.

12.1 Moving in Different Coordinate Systems

When working with the GCVRT system the different types of scenarios allow for a multitude of different input values. The single user who works with his mouse in front of the screen, the multi-user scenario with the panorama or the round table user with the round table and the place holders does not necessarily send input data within the same coordinate system.

Therefore we need functionality in the system to map input values from input devices (mice, wanda, PHO, etc.) to a uniform size, so that their movements are small enough to be accurate in the virtual world and yet large enough for the system to allow swift movement and synchronize the movement in the coordinate systems.

Since these alterations must be performed directly on the input data from the input device there are basically two possibilities:

- To design a client side module which receives movements from the input device and then maps these movements to coordinates that are meaningful in the virtual world.
- Or to create a server side module, which maps changes from each client in different ways.

We will not discuss the latter any further since it would require a tighter coupling between the server and the clients, which we strive to avoid.

The objective of the module is to receive input data from an input device and then to translate them into meaningful movements in the world coordinate system. This coordinate system is the same as is used in the AWM. In some games (first person shooters in particular) the user is given some control of this mapping because they can only change the c, e and g scale factors described below. By adjusting a value (typically with a slider in a graphical user interface) the user can adjust the input module so that very small movements with the input device amount to large movements in the virtual world, or the opposite. We will refer to this functionality as coordination mapping.

When mapping from the movement of the input device to the movement in the virtual world, the only values that we need to know is the scale of the movement (the slider value) and the shift of all values in the x, y and z directions.

The translation can be described as follows:

$$x_w = cx + d$$

$$y_w = ey + f$$

$$z_w = gz + h$$

c, e and g are scale factors and d, f and h are values that adjusts the position of the movement (shifts). The shifts are used for adjusting so that we can map origo from the normal world to origo of the coordination system of for instance the vision tracking system. The scale factors could e.g. be used if all virtual world coordinates are positive values and the input device returns negative values, then the shifts can be used for adjusting this so that the input device does not cause the object (or the user) to move "out of the virtual world". The scale factors ensure that the user is given as fast or as slow a movement as he desire. As with the shifts each axis has its own scale value allowing movement with different accuracy in each direction.

Even though both x, y and z values are specified, most input devices only supply an x and a y value. It is only in the case of a 3D input device that the z value is of any use - however our system is designed to be as general and portable as possible which is the reason for having the z value, and the corresponding scale value g and the shifting value h.

This functionality must be entirely encompassed by the Interaction class. The mapping method should also handle when a user moves his input device to the left, then it should also result in a movement to the left seen from the users viewpoint.

The scale and shift values should be written to a configuration file, and functionality for adjusting the corresponding attributes in the Interaction class should be implemented. The data structure containing the shift values should be a record for each mapping containing the shift and scale values. Since several mappings can be in effect at the same time, the data structure could be made as a vector or array of records.

12.2 3D Landscape Generator

In a context in which the system is applied to an outside area, it is a big advantage for the users to be able to use the information they already have, instead of having to draw a new map of the landscape only to be used in our system. Therefore, we need a landscape generator capable of receiving a standard height contour file format and convert this format to a 3D landscape.

Common for these formats is that they are actually grids with a height for each grid-point. Some formats specify the heights in a humanly readable (although hard to understand) text file, while other formats specify the shape of the landscape through displacement maps. These displacement maps are typically grayscale images in which either an 8 bit or a 16 bit grayscale is used. From the grayscale value of each pixel (point in the grid) the height of that point can be determined. With regards to mapping these text files or displacement maps to real world coordinates, the files consist of coordinates in the UTM coordinate system[1]. Since the displacement map approach is widely supported throughout the scientific community and it is backed by USGS (United States Geological Survey) through the digital eleva-

[1]The UTM coordinate system is based on coordinates in metres rather than degrees, minutes and seconds. A coordinate is mentioned as an Easting and a Northing. The Northing value is the amount of metres north of the Equator. The easting is the amount of metres east of the previous zone - the circumference of the Earth has been divided into 60 zones.

tion map[2] (DEM) and by NASA (National Aeronautics & Space Administration), which both provide satellite imagery in the mentioned displacement map format, this format would be the obvious choice. What speaks against this selection is that EMD makes use of an ASCII format developed by the wind energy department at Risø National Laboratory, Denmark. The format was developed for the Wind Atlas Analysis and Application Program (WAsP) software package which is designed for performing wind resource calculations over a specific area with a specified roughness. This format like the other ASCII formats contains a grid with heights for each node. However, since our objective is to create a generic system that should be available in not only EMD's context, our choice is to implement support for the displacement map height contours. More specifically, we will support the DEM format. Elevation maps of the entire USA are available for free on the internet. However, international DEMs are harder to come by, but we expect this to change over the coming years since USGS is already providing DEMs for certain parts of Europe and are continuing to expand the areas covered outside the USA. We will call the module containing functionality for importing the DEM file and generating the terrain for GenerateDEMTerrain.

We can not take credit for the functionality of the module in its entirety, since the main ideas have been adopted from an OpenGL guide on the Gamedev.NET web site. We will take the basic algorithms and implement them into the GCVRT system.

In order to explain the internals of the system, we will, however, design it using the same approach as the other modules in the GCVRT system.

In order to parse a DEM file the system must be able to read a DEM file and extract the relevant information from it. We shall name this method `LoadDEMFile`. From this data, the system must create a 3D landscape with hills, valleys etc. thereby allowing the user to use existing information rather than having to reimplement existing information in a static 3D model.

For the users to be able to make use of this landscape they also need to be able to see where there are roads, lakes, trees etc. All this information could be inserted afterwards as objects in the virtual world. A just as simple approach would be to enable texturing of the landscape. This way an aerial photography or simply just a regular map with indications of trees, houses etc. could be applied. Although this information would not generate a 3D image of the trees and houses, it would in some cases be sufficient as an indicator of the location of these items. If the users wish to be able to see the trees stick out in the landscape it is easy for them to place a 3D model on the correct location after the terrain class has been loaded into GCVRT .

The requirements for the GenerateDEMTerrain class are:

- It must be able to read and parse the DEM files by the means of the `LoadDEMFile` method.
    - It must be able to generate a 3D model in which all values are scaled

[2]The DEM files provided by USGS can be retrieved from http://www.gisdatadepot.com, and similar files provided by NASA can be retrieved from the Aster project at http://edcimswww.cr.usgs.gov/pub/imswelcome/. For a general introduction to working with DEM files http://www.terrainmap.com is a very informative site, which we would like to recommend.

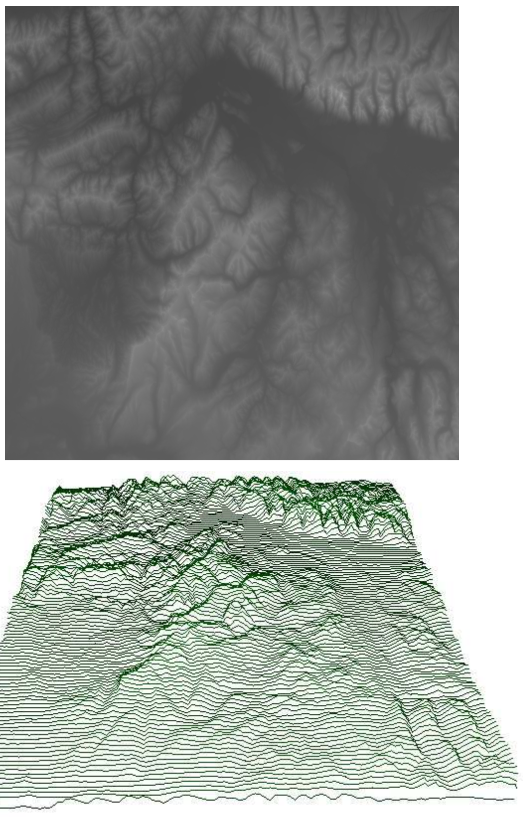

Figure 12.1: *A displacement map, and the corresponding 3D landscape. Height values in the model have been exaggerated by a factor of 2.0 for making the height differences in the landscape more apparent. The white areas in the grayscale image represent the high points in the area and the black areas are the lowest points as can be seen in the 3D model of the landscape. Perspective have been added in order to make the 3D effect in the landscape more apparent.*

correctly according to the rest of the models inserted into the world. This functionality we wish to obtain by implementing a method which takes the information retrieved by the `LoadDEMFile` method and generate a 3D model of the terrain ready for insertion into the scene graph. This method shall be named `GenerateTerrainModel`.

- It must be able to color or texture the terrain model.

When `LoadDEMFile` reads the DEM file, the heights of each point should be stored in a data structure which is easily accessible. For this purpose we have chosen an array, since it is easy to store and access the data this way. The standard DEM file as provided by USGS is quadratic and therefore the size of the array must be the square of the width of the DEM map.

The amount of height values in each row is denoted WIDTH from here on. When `GenerateTerrainModel` needs to access the height at each (x,y) value in the map it can be accessed in the one-dimensional array by calculating a corresponding index using this simple scheme:

$$index = x + (y * WIDTH)$$

When generating a 3D terrain, one has basically the option of either using triangles or quadrangles. We choose to use quadrangles, since the DEM map is also based on squares. In order to be able to generate the 3D model of the map, the map needs to be divided into a lot of small quadrangles which has a height for each corner. The only thing `GenerateTerrainModel` must do is to run through each X and Y coordinates in the map and assemble quadrangles from them as described above, and then construct a large 3D model of these. If the map is small there will be no problem in stepping through each value of the coordinate system. However, if it is a large map a very high amount of polygons will be created. In order to lower the amount of created polygons in the landscape we could introduce a step size, so that only a value for each step size is used for generating a quadrangle point, in effect increasing the size of the quadrangles.

A pseudocode algorithm for going through the entire array and constructing quadrangles of it.:

```
1. For(X := 0; X < WIDTH; X := X + STEP_SIZE)
   (a) For(Y := 0; Y < WIDTH; Y := Y + STEP_SIZE)
         i. Bottom left point in quadrangle:
            x := X
            y := Height(X,Y)
            z := Y
        ii. Top left point in quadrangle:
            x := X
            y := Height(X, Y + STEP_SIZE)
            z := Y + STEP_SIZE
       iii. Top right point in quadrangle:
            x := X + STEP_SIZE
            y := Height(X + STEP_SIZE, Y + STEP_SIZE)
            z := Y + STEP_SIZE
        iv. Bottom right point in quadrangle:
            x := X + STEP_SIZE
            y := Height(X + STEP_SIZE, Y)
            z := Y
```

In figure 12.2 the scheme for loading the data and creating the quadrangles is visualized.

A consequence of the algorithm is that `GenerateTerrainModel` uses quadrangles. Since each quadrangle must be connected to the neighboring quadrangles in order not to have gaps in the surface, the same height value will be used for 4 different corners in 4 different quadrangles. This way each point will have to be retrieved from the array up to 4 times. This will happen if it is a point with neighboring points on all 4 sides of it in the DEM map.

In figure 12.2 the process of first loading the values from the DEM file into the array and then creating a 3D terrain consisting of quadrangles is shown. In the lower right corner the height values for each corner in the first two quadrangles are created. Correspondingly the remainder of the quadrangles could be created. The numbers in the square to the right indicate in which order each point will be calculated. E.g. the points 3, 8, 14 and 17 are in fact the same coordinate in the

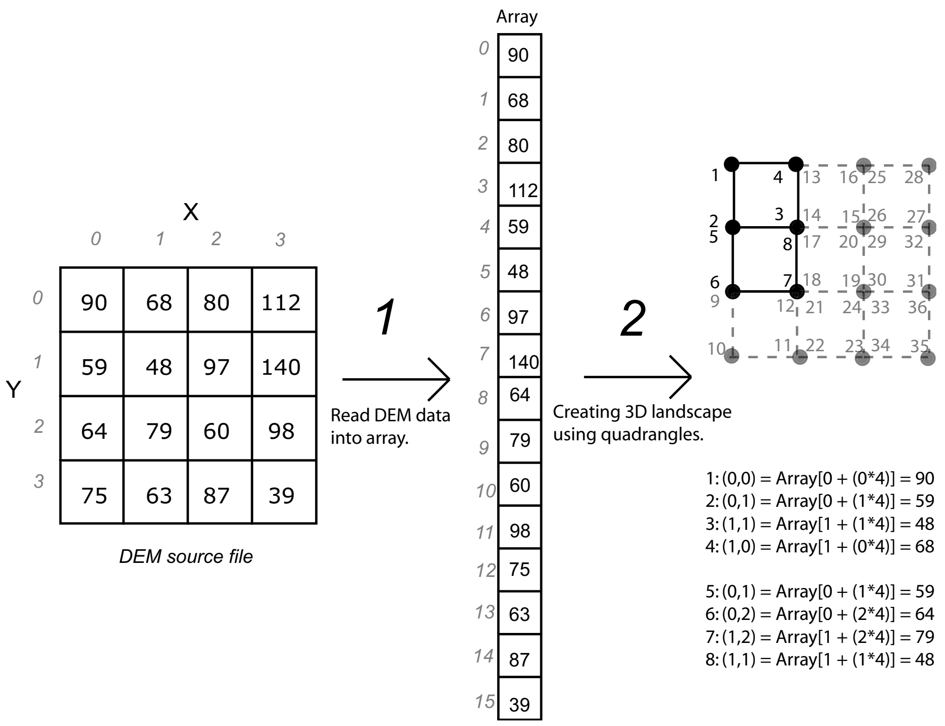
X
Y
DEM source file
1
Read DEM data into array.
Array
2
Creating 3D landscape using quadrangles.
1: (0,0) = Array[0 + (0*4)] = 90
2: (0,1) = Array[0 + (1*4)] = 59
3: (1,1) = Array[1 + (1*4)] = 48
4: (1,0) = Array[1 + (0*4)] = 68
5: (0,1) = Array[0 + (1*4)] = 59
6: (0,2) = Array[0 + (2*4)] = 64
7: (1,2) = Array[1 + (2*4)] = 79
8: (1,1) = Array[1 + (1*4)] = 48

Figure 12.2: *The figure shows the way the* GenerateDEMTerrain *module must work. First the DEM file data to the left is read into the array in the middle. Next the landscape is generated by creating quadrangles for the entire map. To retrieve the height in the (x,y) coordinate of the original DEM file, a new index,* $index = x + (y * WIDTH)$, *must be calculated to retrieve the value from the array. In the lower right corner it is described how the height values for point number 1 through 8 are retrieved from the array. It is also indicated which coordinate in the DEM file that value corresponds to and finally the retrieved height value appears. Note here that points 2 and 5 are actually the same coordinate in the DEM file, the same can be noted about points 3 and 8. Point 3 and 8 is also the same coordinate in the DEM file as point number 14 and point number 17 in the terrain map.*

DEM file, but the value needs to be retrieved in iteration one, two, four and five of the inner loop in the first iteration of the outer loop. Iteration one creates the quadrangle made up of points 1, 2, 3 and 4. Iteration two creates the quadrangle made up of points 5, 6, 7 and 8 and so forth.

Since we need to access the height of each coordinate several times, we suggest creating a method for extracting the height from a specific coordinate. We suggest naming this method `Height`. Basically it must take care of calculating the proper index in the array and return the height at that index. If a snap-to-landscape feature is needed in a terrain generated by the module it could be implemented in a simple manner in that all that would be need was to call the `Height` method with the current position and the height would be returned. This would, however, only work when the terrain is generated by the GenerateDEMTerrain module, and a general method for performing snap-to-landscape would be needed for this.

In order for the `GenerateTerrainModel` method to generate a landscape it must have access to the scene graph, which is located on the clients. However, it can be implemented either as a module that generates a graphical model in a format which OpenSG can interpret, such as VRML 2.0, or it can be implemented as a client module which works by reading the file and then generates a description of the landscape, which can be inserted directly into the scene graph of the client.

By making a VRML file, only one computer needs to run the landscape generator instead of every client having to generate the landscape from the displacement map and then inserting it into the scene graph. However, we choose to give each client the functionality for creating the landscape since there are no synchronization issues in creating a landscape from the same source data using the same algorithm. For this design choice to make sense it requires that the DEM files are shared between users in the same way as the 3D models, described in section 12.4.

The GenerateDEMTerrain class can be contained in a single module. The main methods needed are for reading the input from the DEM files and the other for handling the generation of the 3D landscape from the data in the DEM file. In order for the GCVRTClient to insert the terrain into the scene graph the `GenerateTerrainModel` method must return a node pointer to the scene graph describing the generated landscape.

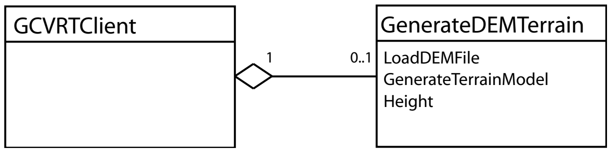

Figure 12.3: *UML diagram for the GenerateDEMTerrain class.*

12.3 The Recording Module

In order for users to be able to have some form of persistent documentation of the process other than the virtual world at the current time, a logging module will be designed. In [JUV02] a save function was planned, therefore we will not design a new save function, instead we will focus on a log. The purpose is to log all relevant messages sent from the server to the clients in order to be able to recreate any state of the virtual world during a session of collaborative work at any given point in time.

The information sent from the server to the clients are available at both the clients and at the server, so we need to make a choice of where to place the functionality:

- **Server-side log**
  The logging module could be placed on the server. A reason for placing the log module server side is that the server has a less complex job to perform than the clients, so placing it in the server would relieve the clients of having to perform this task. In case that the server crashes or looses its network connection all clients will no longer be able to communicate and the log will still be complete because it has everything in it that the server sent to the clients.

- **Client-side log**
  The logging module could also be placed on the clients. This way the log would be saved locally. However, if the client looses the net connection or crashes, the log will be incomplete, since messages sent by the server after the crash will not be logged and even though the client could load his log to recreate the virtual world up to the point of the crash, it would not be identical to the log of the clients which did not crash.

Due to the reasons mentioned in the list, we will design a server-side logging module. In order for such a logging module to be of any use, the module must make it possible to restore the state, to the state described by the log entries. Apart from the messages needed in order to restore the state a further information namely the time a given message was sent would be of interest.

In order to restore the state the following is needed:

- In order to determine time, place and person responsible for any event, the client ID must be saved along with all change messages, as well as a timestamp for each message. By logging the change messages we will obtain the position and orientation of each object in the virtual world.
- In order to restore the world's state the add and remove messages must also be saved so that we are able to add and remove objects during play back.

In regard to the first item, a way of being able to determine which user did what we could introduce a new mapping on the server. Even though a client will have a unique ID at any given time, it can be difficult to identify which user was controlling that client, therefore it should be possible to identify which user did what and not just which client ID did it. In order to do this, a mapping between the user name and the client ID is needed on the server.

In the log file, the client ID that performed the action could then be replaced by the name of the user and make it easier to see which person did what. Because of this information need, a new message is needed in the network protocol of [JUV02] containing the name of the user and the message type. The server can identify the client by the socket the 'name' message was received on. Therefore, it is not necessary for the client to send his client ID in the message.

Locks should not be saved since they are of little importance when it comes to restoring the world to a previous state. Also, avatars in the virtual world are represented as any other object in the real world. Therefore it is not necessary for the log module to save an avatar's position explicitly since this is already done in the first step. However, in order to be able to see who moved a certain object, the avatars must be visualized so it is possible to see which object they are working on. This visualization could e.g. be a line from the avatars hand to the object or similar. This functionality is however not something the logging module should contain, it is a job for the designer of the avatar to provide such information in the visualization.

With the information mentioned in the list, the system is able to recreate any state of the world, and it is even possible to see how the world has evolved over time. This way it would be possible to play the log back at a certain speed to see how work progressed in either real time or a predefined speed setting. If the world has evolved over a great period of time, the log will eventually become very large. However, we do not see this as a problem unless disk space is a major issue, since the contents of the text file is very basic and only contains the contents of the messages sent by the server.

From the needs identified above we can say that a method for writing log entries to the log file must be made available. This method we shall name `writeLogEntry`. Also a method for playing the log back must be made available. This method we shall call `playLogBack`. Correspondingly a message must be added to the protocol that is able to start the playback.

The main class of the logging module, should be named accordingly and we shall give it the name Log. The logging functionality in the `writeLogEntry` method should be called by the ServerCom class every time that ServerCom receives an add, remove or change message and thus create a log entry in the log file.

The `playLogBack` must contain functionality to start the playback. Since the log is located on the server, the playback could be performed similar to when a client joins a virtual world after others have already worked in it for some time. That is, a series of add and change messages will be sent to the client bringing its virtual world and AWM up to date. By utilizing the same functionality we could have `playLogBack` send the messages from the server, and then use the timestamp to indicate when to send the messages. In the Log module a variable should be made available indicating how fast the playback should take place. This speed setting could e.g. be encapsulated in the message sent from a client indicating that he wishes to play the log back, so apart from a message containing the playback command, it must also contain a number indicating the speed of playback.

From the description above, the UML diagram in figure 12.4 of the log module can be made.

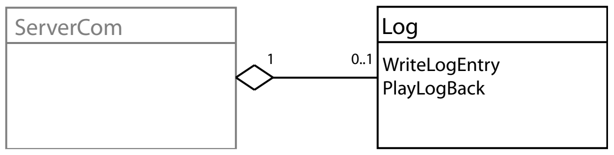

Figure 12.4: *UML diagram for the Log class. The greyed out module is the existing class in GCVR, and the darker one is the new module.*

As mentioned, when a restore is executed the entire world must be updated in much the same manner as when a new client connects to an existing visualization. A series of add, remove and change messages are sent to each client overwriting their existing world. The existing world is however not lost since it also exists in the record module on the server, so if need be the replaced world can be restored as well. If a stop request is made during playback e.g. if the users wish to continue working from this point, a new log should be started in order to have log that reflects the work made on the restored world.

12.4 File Distribution

In the system, which we are designing, there is a need for all clients to have access to the files which contain representations of the 3D models that are part of the shared virtual world. There are basically two ways to ensure that these files are accessible to all clients:

- Network transfer: The user of all clients use mainstream file transfer utilities to get the needed files before a session of collaborative work is being started. This is a very inflexible solution because it will require a user to switch application to actively ensure that the files representing new 3D models added during work are accessible. A similar solution would be to incorporate the file transfer program into the GCVRT system and enable the system to distribute the files automatically. This requires some amount of design, implementation and test and does not bring up any new technology despite the amount of work put into it.

- Network sharing: Network sharing techniques have a major advantage over transfer techniques. Network sharing methods allow multiple users to access the same copy of a file from a single location (although only one can have write access to the file at any particular instant). In essence network shares allow for a file on a remote computer's disk drive to be accessed as if it was on a disk drive in your computer.

The network sharing solution requires next to no work to implement since it is a well-established general technique. Therefore, we will proceed with that solution. The next problem is to select the specific system. The GCVRT system is aimed mainly towards the Linux and Irix operating systems which are both instances of Unix. Therefore, the immediate choice would be the Network File System (NFS). It is typically used for Unix to Unix file sharing although clients exist for Windows and Macintosh. NFS generally requires administrator access on both server and client which may be a problem for some users. Another possibility is Samba, which is a clone of the Microsoft SMB filesharing system. Unix systems can serve "SMB" shares to Windows and other SMB clients using the Samba SMB Clone. Since the GCVRT system has been designed to be portable, and a Windows version of the GCVRT system may come up one day, we recommend the Samba system. However, since the file sharing system is not a part of the GCVRT software, ultimately the choice is up to the users of the system.

Common for all file sharing systems is that a directory named "models" should exist in the directory containing the GCVRT software. Any shared drives should then be mounted as this directory.

This approach enables all users to have immediate access to all 3D models introduced into the system. No new network protocols are needed for supporting this strategy - the add and remove messages defined in [JUV02] will suffice. The physical drive which is shared can be located on the computer running the GCVRT server or any client. The only restriction is that possible fire walls and proxy servers must not prevent the drive to be mounted across the different networks.

13 Assembling the Server and Client Components

At this point, the design of the modules, which form the GCVRT system has been completed. In this section, we will describe the assembly of the server and client parts of the GCVRT system from the building blocks described in the previous sections.

13.1 Assembling the Server

The design of the server does not differ much from the original design of the GCVR server described in [JUV02] because the vision and head tracking information is handled by the master client. The advantage of this approach is that the server is not dependent on the scene graph and is able to run on a computer system which is not configured especially for this purpose.

The main difference lies in the recording system which is indicated in the right upper corner of figure 13.1 which also shows the entire class hierarchy of the server component. The Log class is able to log all add, change and remove messages sent from the server. In order to ensure modularity, the module is started up by the central GCVRServer module and is then associated with the ServerCom, which is then able to use it for logging the traffic.

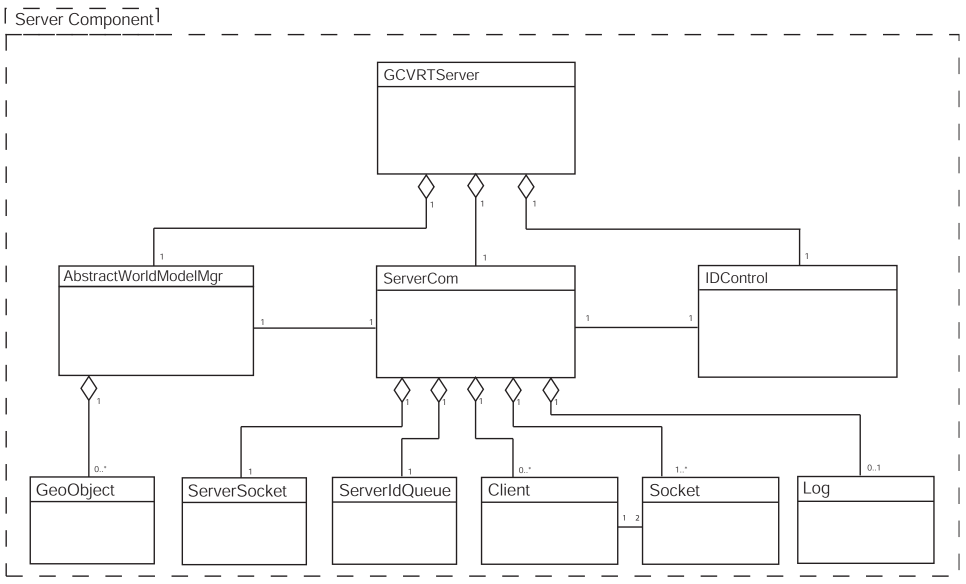

Figure 13.1: *The assembled server component.*

Apart from the issues mentioned, the responsibility of the server is (as originally designed) to keep the abstract world model consistent throughout the network of clients which have logged on to the server.

13.2 Assembling the Client

The original responsibilities of the client were the following:

- To ensure that the global (the one on the server) abstract world model is constantly updated with changes generated by the users of the clients.
- To generate a scene graph on the basis of changes on the abstract world model, which are received from the server.
- To visualize the virtual world on the basis of the abstract world model and the 3D models involved.
- To receive input from interaction devices.

These are still the main responsibilities of the client. However, all clients must now be able to act as master client. That is, the clients must all be able to receive information from head and vision tracking systems, interpret its meaning and translate it into the messages which are part of the GCVR protocol. Also, all clients must be able to generate landscapes from commonly used file formats. This is handled by the GenerateDEMTerrain class.

In order to be able to receive head tracking information, the WsUDPSnd module has been added to both the vision tracking system and to the head tracking client. Also, these systems have been altered such that they are able to transmit the tracked information on the network. In order to receive it, a ClientComUDP class has been added to the system. It is started up by the GCVRClient and should be used in two ways:

- When used for receiving head tracking information, the ClientComUDP is used by the GCVRClient module, which is able to implement the changes to the head position.
- When used for receiving vision tracking information, the GCVRTClient module should use the ClientComUDP module and the Interaction module for determining the association between the PHO and a virtual object. Once the association has been made, the ClientCom module should send a lock request to the server, and when an accept message is received, the changes received by the ClientComUDP module are transmitted to the server through the ClientCom module. Finally, the server implements the changes into its abstract world model and distributes the changes like all other changes to all clients (including the master client, which actually generated the change). In this way as few as possible inconsistencies between the clients are generated.

The complete class diagram for the client component is illustrated in figure 13.2.

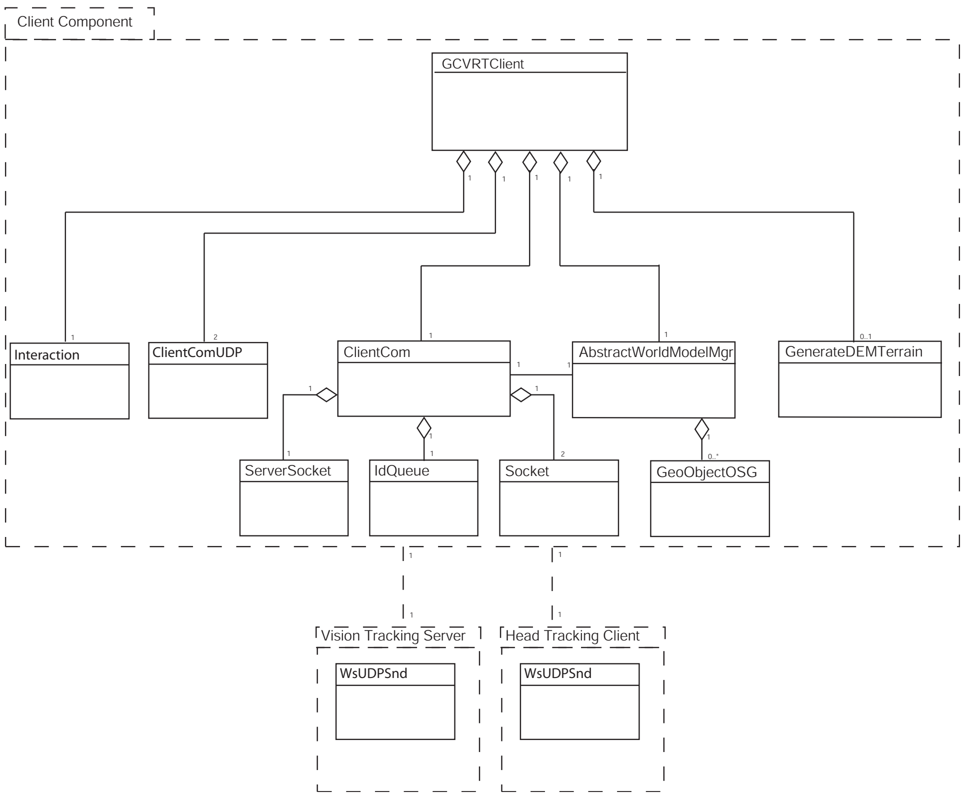

Figure 13.2: *The assembled GCVRT client component and its associated head and vision tracking components.*

Part III

TEST, STATUS AND CONCLUSION

This part contains four chapters. Chapter 14 contains a specification of the way we wish to test the system. In chapter 15 the status of our implementation is evaluated, and in chapter 16 the future work that should be performed on the system is identified. Finally, in chapter 17 the conclusions of the project can be found.

14 Testing Specification for the GCVRT System

Now that the design has been completed a specification of how to test the system will follow in this chapter.

However, first we will introduce different ways of testing adopted from the Extreme Programming (XP) paradigm:

- **Functional testing**
  Functional testing is a test of the functionality specified during the analysis and especially the requirement specification. In XP they typically use stories, small descriptions of what a module should be able to do instead of a requirement specification. However, we are confident that a functional test is what is needed in order to evaluate how the system works. The functional tests are basically a test of whether the system provides the functionality the users wanted it to provide. The test cases should be stated by a user and it can be performed by either a user or a programmer of the system.

- **Unit testing**
  Unit testing is applied on a method by method basis. If a method is specified in the design, then during implementation it must be tested that the method performs as designed. This means that a unit test is performed by the programmer(s) who implements a method.

- **Performance testing**
  During performance testing the performance of each module is tested. This test must be performed and specified by the programmers.

- **Stress testing**
  In a stress test we test what happens if a module is overloaded. This test must also be performed and specified by the programmers.

- **Parallel testing**
  Since our objective has been to merge GCVR with the round table scenario a parallel test is required in order to determine whether the GCVRT system is still able to perform at least the same duties it was able to perform according to the GCVR design. This way we can be confident that the GCVRT system can replace the old design without trouble. A test of what the round table scenario was able to do, and if it is still able to be done should also be performed in order to determine this. This test must be specified by the programmers in close contact with the user since they ultimately know how the system behaves in work setting.

The test types mentioned above will be applied to the GCVRT system and we will make a section for each of the above bullets. However, given the overall nature of the design, it does not always make sense to perform a unit test on each method in the design as much of the functionality is described on a module basis and not a method basis. Instead of the method by method unit test we will perform a module based unit test.

Also, a test for how well the GCVRT system supports transitions towards the co-constructive level of collaborative work could be performed. However, a test of this type requires that users use the system for quite some time in order for us to be able to evaluate what happens on a general basis. A specification of such a test would require us to first analyze how collaboration actually takes place and then compare the new work form with the first analysis. This test will not be performed because it is not within the immediate scope of this project to analyze several areas of application in detail in order to be able to determine to what degree the different transitions are supported.

14.1 Functional Testing

In the requirement specification in chapter 7 a number of functional requirements are mentioned. We will now list these and specify a success criterion for whether the requirement is fulfilled or not. The functional requirements should all be evaluated by the users of the system since the requirements state their goals for the system. If the users regard a functionality as not performing as it should, the functionality is incomplete and needs further work, and should possibly be redesigned. However, if the users agree that the functionality is in order, the requirements have been fulfilled and the system is able to perform the actions it was intended to do.

In the following list all the functional requirements for the GCVRT system are listed:

1. The first functional requirement we need to test is whether the system actually solves the problem of providing users at the round table with a perspectively correct visualization of the virtual objects. This involves the head tracking system as well since it delivers the position of the user's head.

    (a) The success criterion for this requirement is that we must test whether the user's viewpoint in the virtual world is moved according to the head movements. It will be a subjective evaluation by each user whether the perspective is correct for them. As a guide some physical objects other than the PHOs could be placed on the table and the user can then compare the perspective of the physical object to the perspective of the virtual object and use this as a reference for their evaluation.

2. The second functional requirement that needs testing is whether it is possible to associate PHOs and a virtual object, and whether it is possible to remove the association.

    (a) The success criterion for this requirement is whether the user can grab an object and later on release it using the description given in chapter 11.

3. The third functional requirement that must be tested is whether it is possible to receive the changes of PHO states and move the virtual objects around accordingly.

    (a) This requirement will be seen as fulfilled when the movements of the virtual objects match those of the PHOs. The users must be given a test case in which they have to map a virtual object to a PHO and move the virtual object to a predefined location, on which they are to release the virtual object of the mapping. The users should also evaluate whether they felt like the virtual object followed the movements of the PHO fast enough. If they are able to perform the actions and they feel the responsiveness of the object movements is fast enough, the functionality will be seen as working successfully.

4. The fourth functional requirement we must test in order to convince ourselves that the customer can use the system is whether the recording feature provides a complete log of all the necessary information.

   (a) This functional requirement is harder to test. We imagine however that a video recording of an entire work setting in which the log feature has been enabled all the time should be compared to a playback of the log from the server. If what can be seen is identical we evaluate the feature as being successfully implemented.

5. The fifth functional requirement that needs testing is whether an external speech transmission system is actually good enough for providing the users at the round table with a way of communicating with users outside the round table, and vice versa.

   (a) This test must be done by the users of the system. They must subjectively evaluate whether the quality of the speech is good enough, and they must evaluate whether network latency is to high. If the words a user utters do not get spoken until a few seconds later it might be more confusing than helpful. If the users evaluate the sound quality as being OK, and the latency as not being a problem we regard the feature as being successfully incorporated.

6. The sixth and final functional requirement that must be tested is the terrain generator. Is the generated virtual terrain sufficiently accurate, or must the terrain have further improvements such as trees and other objects sticking out of the ground.

   (a) This system must be tested against images of the actual area. Are the hills in the correct location, the lakes, the valleys etc. It should be tested by a user who is used to working with landscapes since they are more aware of what details are needed. If they evaluate the terrain as being accurate we will regard the module as being implemented.

   (b) The user should also evaluate whether it is necessary to insert trees and other objects sticking out of the ground. If it is it can be done by loading and positioning them like other objects in the virtual world. It should not be the task for the terrain generator.

Some further tests of the functionality could be thought of as including testing which arena that provides the best work setting: The CAVE, The panorama, The PC or The round table. The test could be performed in such a way that the same task was given to people in each arena and then the time or the way in which they solve the task could be measured and compared. If users in an arena is performing much better or worse than the other users in the other arenas the reason should be located and it should cause improvements for the arenas that did not perform well, so that they are equally good at supporting collaboration for the GCVRT system. However, since the GCVRT system does not support the CAVE and the panorama we do not yet feel the time is right to perform such a test. It should be performed when the support for the CAVE and the panorama is ready, though.

Another functionality that should be tested is whether the deity mode is a good enough mode for working at the round table, or a more schematic way of working, such as in a CAD system would be better. The expert users will probably not mind the more schematic way, however, new users that see the system for the first time, might find it intimidating and will be reluctant to work with it if it is too technical. This test would require further user testing, and would require a test case to be implemented so that the users could compare the two ways of working. At this time, however, we feel that such a test should not be performed until we have heard dissatisfaction from the users with the deity mode of working in the virtual world.

14.2 Unit, Performance and Stress Testing

As mentioned in the introduction to this chapter, the unit test is a test of each method in the design. However as also mentioned in the introduction to this chapter, we will in some cases specify a test of the module since functionality is described by a large number of small methods. It must be determined whether or not the method or module performs as intended. Also we have decided to include the performance and stress tests in this section since they will be performed on some of the same modules.

Tracking Systems

The methods for the external head and vision tracking system of interest are:

- **WsUDPSnd**
  - `enqueue`:
    Is responsible for putting either VT or HT messages in a queue.
  - `dequeue`:
    Is responsible for removing the VT or HT messages from the queue and sending them over the network.

These methods should be tested for their behavior when a very high but realistic amount of messages is enqueued. The result of the test of `enqueue` should show that even though the queue or buffer is flooded the `dequeue` will continue to send the latest messages in the queue. If this does not happen and the system crashes, the module has a flaw and must be debugged.

Currently both the head tracking system and the vision tracking system only contain a small amount of tracked objects. The VT system only has 6 PHOs it is able to distinguish between and the HT only has 8 receivers so that no more than 8 heads can be tracked at the same time. Therefore it will be hard to test performance and stress testing with only these modules, which is why a separate class for generating the same messages as those generated by the VT and HT systems when PHO and head changes are received should be implemented. This way we will be able to test just as many messages as we need to. The class should have no other purpose than to function as a testing module.

GCVRT Client Functionality

In the GCVRT client the messages sent by the tracking systems must be received, and applied. The methods made for doing this are:

- **AbstractWorldModelMgr**
  - `getPerpendicular`:
    Responsible for finding out which object directly above the PHO is closest to the PHO.
  - `whichObject`:
    Responsible for finding out which object intersected by a given line is closest to the avatar/user.
- **ClientComUDP**
  The class is responsible for receiving the messages from both the vision tracking system and the head tracking system. It must create the sockets on which communication is to take place, and then it must communicate the messages further on in the system.
- **Interaction**
  The class is responsible for receiving user input, from e.g. the mouse or a wanda. The Interaction class is responsible for incorporating the scale and shift values defined in section 12.1.

The way for testing the `getPerpendicular` method is to perform mappings on the boundary of the vision tracking area. If these can be performed flawlessly then the function must work. A performance test of this method could be to call it in a loop in a very large virtual world in which it would have to go through a large search area. A stress test could be to have it in an infinite loop and have it search through the large virtual world after intersected objects and see how it would react to the stress load.

The ClientComUDP module could be tested by sending one of each type of message to it. If it understands and implements the messages, the functionality of the class is correct. If it does not, something basic is wrong with the class and further design and a reimplementation of it is necessary. After it has been established that the module is capable of receiving and handling the messages correctly it could be tested whether the system is capable handling heavy traffic by causing another module to generate messages very fast and send them to ClientComUDP to observe what happens when there is buffer overflow. If the module handles overflow nicely the module can be considered complete and ready for use.

The Interaction module should be tested by testing whether it receives the input from the input device. If it does then it should be tested whether the values received are corresponding to the movements made with the input device. When this has been established further testing of how these values should be scaled or shifted to make sense in the virtual world could be performed and thus the default values could be determined this way. There is bound to be very little load on this module even if it has to receive input from the input devices given that the calculations it must perform are so simple, so a stress test of it would not make sense.

3D Landscape Generator

In the GCVRT client the GenerateDEMTerrain class is a very important class as to providing the users with an accurate terrain. The methods of interest in the class are:

- **GenerateDEMTerrain**
  - `LoadDEMFile`:
    This method is responsible for retrieving the data from the DEM file and inserting it into an array.
  - `GenerateTerrainModel`:
    Is responsible for generating the 3D terrain out of quadrangles.

The method `LoadDEMFile` should be tested by giving it DEM files of very large sizes. If it is able to parse them with no problems, it works. If it is too slow it should be reimplemented and made faster. However, as mentioned in section 12.2 the design is made so that the time complexity of the indexing is very low and it is not necessary to search through the array but rather individual values can be accessed directly. We expect the access time to be very low, but it should be tested nonetheless. `GenerateTerrainModel` consists of an inner and an outer loop which each iterate the same amount of times. Therefore the running time of this loop is $O(N^2)$, where N is the amount of iterations in each loop. The complexity of each operation in the loop is very low since it only accesses `Height` four times in each iteration. The method should be tested on very large DEM maps, and it should be measured how long it takes to generate the 3D model of these worlds, compared to smaller worlds. The method is only meant to be called once so it does not have to be done in less than a second. However if it takes more time than the user is willing to wait, we will have to reduce the running time of the algorithm further.

The Recording Module

This module only has two methods of interest: `WriteLogEntry` and `PlayLogBack`.

The requirements are:

- **Log**
  - `WriteLogEntry`:
    This method must write each and every add, remove and change message ServerCom sees, to the log file.
  - `PlayLogBack`:
    This method must send all the messages in the log file to all the connected clients. According to the timestamps delays between each message should be inserted.

The way of testing this module should be to stress test the `WriteLogEntry` method, since it is very important that it is able to write every message to the file with the correct timestamp. If it looses messages or gives messages incorrect timestamps, the method should be considered inadequate and must be subject to redesign and optimization in order to solve the problems. If, however, the method is able to write all the messages to the log file, the method has passed the stress test. The `PlayLogBack` method does not need a stress test in that will only be called very rarely. The method should only be tested for whether it is able to read the log file and send the messages in it on to ServerCom which must then distribute them to all the clients. The method must insert pauses between each message according to the timestamps in the log in order to visualize the progress of the work during the playback.

File Distribution

This module is entirely based on Samba, so therefore the only thing we wish to test is the transfer time for a file on the server to the client. And if multiple add messages are received how will this affect the transfer time.

We therefore propose to stress test the module by adding say 100 different 3D models of a relevant size at once and then measuring how long it takes on the LAN, and how long it takes on a WAN. If either takes more than an unreasonable long time (longer than the user wants to wait), we should reconsider the use of a shared file system service and perhaps reimplement the thought of having the clients download the models to a local library prior to starting collaboration. If it takes less than the user's threshold on both the LAN and WAN the module will be considered as conforming to the design and it is regarded as OK.

Overall Performance Testing

The overall latency of the system, with regards to sending vision tracking information through the server, should be tested. The test should both contain the actual delay in milliseconds and the users' evaluation of the feeling of the delay. If it is too high for the users to find acceptable a redesign of the way the master client handles VT changes is necessary or other optimizations might be tried out first before changing the system structure.

Another performance testing issue is a test of how many objects the system can handle. How many objects is the AWM capable of handling before it becomes too hard so search through it for the `getPerpendicular` and `whichObject` methods. Or will the load of the rendering system be the major problem before this becomes an issue. These tests must also be performed in order to say that the system is conforming to the requirements and design of the GCVRT system.

Further it should also be tested how many users the system is capable of supporting given the bandwidth requirements.

Finally the GCVRT system should, like the GCVR system, be tested on an ADSL line. The test should indicate whether the new functionality have caused the client to be unable to perform on the same amount of bandwidth or if the bandwidth requirements have increased.

14.3 Parallel Testing

In order to perform a parallel test of GCVRT and GCVR we need to list what GCVR was capable of and compare this to what GCVRT does in the same situation. If they both do the same, then the merging of the GCVR and the round table scenario has been successful and GCVRT can be considered a successful merge of the two scenarios.

Normally in XP such a parallel test is run over the course of several months in order to make sure the new system behaves exactly as the old one (or better), however since GCVR was never completed such a lengthy test is not possible. Due to the lengthy nature of such a test, we suggest waiting until GCVRT has been implemented entirely before testing what the behavior of the system is. Therefore the parallel test should not be performed until this is the case.

14.4 Final Remarks

There are two reasons for not including the test results in the report. First of all the tests require much time, and secondly, since we are under a certain time strain we have prioritized executing the tests lower than implementing the system and specifying the tests which eventually must be performed. Another reason why the tests have not yet been performed is that the implementation is not yet in a state in which it is possible to test and evaluate the complete system.

In the following chapter the status of the implementation will be explained further.

15 Status of the Implementation

The main focus in this report has been on analysis of the problem domain and a corresponding design of the GCVRT system. However, in this section, we will describe the progress of the implementation of the design. We will do this in the following sections, each describing a specific part of the design.

Tracking Systems

The common module for transmitting data from the head and vision tracking systems called WsUDPSnd has been implemented on the Microsoft Windows platform and is fully operational. At the time of writing, it has been used in the vision tracking system and the appropriate alterations to this system necessary in order to use the transmission system has been made.

Regarding the head tracking system, we will soon be in possession of the code and we will then be able to implement the WsUDPSnd module in this system too.

Client Functionality

The ClientComUDP module for receiving data from the head and vision tracking systems has been completed. The Interaction class has almost been completed: The part needed for maintaining PHO mappings is finished but the part responsible for calibrating the coordinate systems has not been started on. This does not seem to be a very time consuming task. At the time of writing, the PHO mapping works even though it is imprecise and there are problems with making the system perceive when a user wants to remove an association between a PHO and a VO.

The system is almost complete regarding the functionality for setting the right head position. The part of the system performing this task just needs to be merged with the rest of the system. This does not seem to be a major task, and can be tested in practice as soon as the functionality for transmitting the data is implemented in the head tracking system.

Finally, the GCVRT client must be able to communicate alteration to VOs associated with PHOs to the GCVRT server. This has not yet been implemented but most of the needed functionality has already been implemented in the original GCVR system. Therefore, this should also not be a major task.

In conclusion, the GCVRT client is not fully operational. Except for the imprecise PHO mapping problem, the individual problems have been solved but the solutions still need to be combined and tested.

Coordinate System Calibration

Only a skeleton method for applying the scale and shift values to movements by e.g. the mouse has been made. The exact functionality has not yet been implemented. The structure is there, however the functionality is still not complete. After the functionality have been implemented further testing is necessary in order to evaluate which default scale and shift values should be used.

Landscape Generation

The landscape generator is currently capable of reading DEM files and save the height values in an array. Also, the `Height` method is functioning according to the design. The algorithm of the `GenerateTerrainModel` method is also in place, however the integration with the scene graph is not. Also the transformation node is not yet inserted, but this will be a small task once the connection to the scene graph is established.

The Logging System

The functionality for creating the log system is complete, but it has not yet been incorporated into the server. The functionality has been experimented with on the client on which the method have been tested to work as intended. All that needs to be done is implement the module into the server and have ServerCom call the method for creating a log entry at each relevant message. The functionality for playing back a log is already there, all that needs to be done is to go through the log and create add, remove and change messages according to the format and send these messages to the clients, then the functionality of the log module is complete. The last thing that needs doing before the module can be used is to incorporate the playback message into the protocol. If it should be possible to stop playback half-way through, a stop message should also be implemented, but this was not intended in the design.

15.1 Current Status

At the time of writing, most of the features necessary for determining the feasibility of the design have been either completed or are at least in a state indicating the amount of work, which needs to be put into it:

- The common module for transmitting data from the tracker systems has been completed and integrated into the vision tracking system. This has not yet been done in the head tracking system, meaning that in the round table scenario, the virtual world will not be correctly mapped onto the table because the user's view point in the virtual world cannot be set correctly relative to the user's point and direction of view in the physical world. Our experience with the vision tracking system indicates that implementing the module into the HT system will not be a major problem.

- The client part of the system has access to modules for receiving data from both the vision and the head tracking system and the ability to implement these data into the scene graph. However, regarding the data from the vision tracking system, the GCVRT client is not perfect. There are problems with removing an association between a PHO and a VO and the calculation of the mapping is also not precise enough. Further experiments must be conducted in order to perfect this interaction technique. Calibration of the coordinate systems may be the key to resolve this problem. The same applies for mouse interaction. The connection between the PHO mappings and the GCVRT server has not yet been implemented.

- The functions for calibrating the coordinate systems has not yet been implemented. These are very important for creating a higher degree of usability.

- The landscape generator is at an experimental stage, which allows landscapes to be generated, but not yet to be inserted into the scene graph.

- The logging system is able to log all information, but not yet to play it back.

In general, the system is not yet operational. However, all parts of the design, except for the calibration of coordinate systems, have been implemented and experimented with to a degree where one can say that the overall design of each module is viable.

16 FUTURE WORK

The design and implementation of a VR system is a very large task often involving multiple external systems. In order to get a system, which is usable in practice, it is necessary to make the design and implementation an iterative process in which every iteration is followed by a test. Due to time constraints, this has not been the case in this project - but in order to get a usable system, it must be. Apart from making a full implementation of the GCVRT system and a full-scale test, we have encountered some subjects during the analysis and design phases of the project, which require further work. We will describe these subjects in this section.

INTEGRATION WITH A VR LIBRARY

One of the main problems in this project and its predecessor, the GCVR system, has been to integrate a VR library into the system, which is able to provide an abstract support layer for input and output devices. Specifically, we are interested in getting the GCVRT system to work with VRJuggler, which is an open-source VR library. Unfortunately the use of VRJuggler presented a problem because the GCVR system on which the GCVRT system is built, is based on the OpenSG scene graph API, which was not until recently supported by VRJuggler. The GCVR system was based on OpenSG because it was a complete, portable open-source scene graph and because the VRJuggler development team promised us that VRJuggler support for OpenSG would exist in the 3rd quarter of 2001. This was not the case. We could have changed to another scene graph API, but the point of this project was not necessarily to implement a fully operational system.

Now, support exists and has been confirmed by developers outside the VRJuggler development team. Therefore, an important part of the future work lies in integrating the GCVRT system with VRJuggler in such a way that input and output devices can be perceived as simple abstractions. Then, support for CAVEs, panoramas, stereo vision using HMDs and a variety of different input devices can be accessible and it will be up to the users to choose.

FURTHER TESTING

Once the VRJuggler integration has been performed, further testing becomes relevant and the following questions must be answered:

- Which arena type works best and do the different types support each other? At the time VRJuggler has been integrated, support for CAVE, panorama, PC and the round table should exist.
- Does the working habits of the people involved become more co-constructive, co-operative or co-ordinated?
- Is it in fact possible to collaborate in a meaningful way having both VR users and round table users in the same virtual world?

It seems that the success of a system like the GCVRT system depends on whether it can improve on or contribute in some way to the way people work locally. If this goal can be achieved, the real goal - to enable people to collaborate even though they are located far away from each other - will be far easier to achieve. If people close to each other can be persuaded into communicating using a computerized medium (that is, they communicate in a virtual world), the long distance version is not far away.

Means for Encouraging Creativity

In a system such as the one designed in this project, there is little room for being more creative than what can be described by manipulating already existing 3D models. Therefore, a more mature system should provide at least two new types of manipulation:

- Manipulation of sub-models: It should be possible to take a part of an already existing 3D model and treat it like it was a complete model. There are several ways to do this - one of them is described in [JUV02].
- Interactive 3D modeling: In a mature version of the GCVRT system, many interaction devices should be supported. One of those is tracking of hand gestures, which could be used for enabling users to form complex 3D models with their hands alone.

Especially the second item would greatly improve the opportunity to find creative solutions to problems being solved in a virtual world - but the integration of such a module would require at least a set of complex messages in the GCVR protocol, or in the worst case a complete redesign of the networking modules.

Interaction

We have not paid much attention as to how users can control parameters of the VE - for instance changing scale of a wind turbine or color of a house. Most applications in which user interaction is required have a minimum of user friendly interfaces to functionality, but up until now we have assumed this additional functionality could be accessed through a terminal. This approach would be inefficient in the long run so we need to integrate these interfaces into the environment more seamlessly. Therefore, we propose two different ways of accomplishing this that would be interesting to investigate in the future.

- **External GUI generator**
  We can use a GUI generator to create buttons, sliders, menus, text areas etc. Many different generators exist, capable of generating different source code. This means that work has to be put into finding the optimal one for the job, both in terms of its ability to integrate with the existing system and in terms of how it looks as opposed to what people would expect.
- **Extend GeoObjectOSG**
  Another way could be to integrate the interfaces in the existing GCVRT code by extending the class GeoObjectOSG and letting menus and buttons be just another node in the scene graph. Whenever these objects are activated in some way they would themselves know what to do - for instance selecting a menu item would cause a cube to appear, textured with filenames or moving a slider would make a turbine larger or smaller.

Also, it should be considered, which approaches can be used effectively in stereo visualizations and which cannot. To ensure portability it should be evaluated whether or not the selected approach can be used without problems in all the arenas we have discussed (the round table, CAVE, Panorama and PC) without major changes.

From the different arenas we have a lot of ways to interact with the virtual world and, in reference to this section, with menus, buttons and sliders. From what we have discussed in this report the PHO stands out. PHOs, just as a mouse or wanda, could be used to interact with the interfaces. For instance placing a PHO under a virtual button would activate said button. The immediate problem with the current status of the tracking system is that it does not allow PHOs to be tracked anywhere else than in the plane - either the users must look down upon the table in order to be able to place a PHO under a certain menu item or another interaction method must be devised. An interesting question that could be answered with future work is whether or not users want their interfaces the same place all the time (i.e. on the table in front of them) or whether they want the ability to move the interface to anywhere they deem appropriate. If the latter is chosen and the user is in the round table scenario the question arises on how to place the PHO correctly in relation to for instance the menu.

It should be analyzed what functionality users need access to and how that functionality can be made available in all arenas (both to maintain modularity and to reduce the learning curve of the interface system). This leads to two possibilities - either all interfaces could be made in one common way for all arenas or maybe it would be advantageous to optimize the interfaces for each arena separately.

INFORMATION OVERLAY

In cases where the ability to move objects is not adequate, more information is needed than can be inferred from the objects in the virtual world.

One way to provide such an overlay feature would be to duplicate the landscape layer and make the duplicate transparent so that it does not add further load to the rendering process. This transparent layer could then be raised above the ground level of the actual landscape and all information about production or whatever the user wants to show in the information layer could be drawn, or written on this layer. Since the layer is a duplicate of the original landscape the information written in that layer will follow the contours in the landscape and will thus always be visible since they are raised above the ground level.

However, in certain situations the users might rather be without the information in this layer, so it should be possible for the users to select whether they want the layer visible or not. The easiest way to provide the users with such a choice would be to have a menu for it. In WindPRO multiple information layers are applied instead of just one, each containing specific information which can be shown if the user turn the layer on through a menu. Similar functionality in GCVRT would make sense given that users are then able to enable several types of information to be overlaid or just a single type of information.

If it is not of interest to have the information written at the correct position in the landscape, then another way to provide the users with the needed information could be to have a small text area always visible to the user in e.g. the upper right corner of the display.

No matter if the layer option or the text area option is chosen it will require that new functionality must be added to the GCVRT system. This module must be capable of calculating things specific to the application area. At EMD it would as a minimum have to be able to calculate the production of the turbines, and possibly the amount of noise at given points and maybe the amount of shadow flickering for neighboring houses. However, in an architect scenario where they are designing a new office building that must provide a specified floor area, the total floor area of the building must be able to be calculated and shown in a designated area as they stretch or otherwise alter the structure. In this scenario it might also be interesting to know how many floors there are in the building which could be shown together with the floor area information and it might even be interesting for the architects to know how much ground area the building takes up, which is a further variable that could be shown in the designated area, there are many possibilities. In any case, such an application area specific module must be developed, in order to have the information overlay feature functioning. We imagine that in a future revision of the GCVRT program, the implementation could include an abstract class which the programming division from e.g. EMD or some other firm who had an interest in using the GCVRT system could then extend and provide methods using the same interface, so that it would not be a concern of ours. Rather, the people with the knowledge of how to perform these calculations who have already implemented the algorithms should also implement the functionality into GCVRT , instead of us having to reimplement it and possibly introduce errors to the calculations. This strategy would also leave the GCVRT system maintaining its generic features and it allows companies to extend the system with a module for calculating application area specifics, which in our minds is the best solution.

17 CONCLUSION

As described in chapter 1, the goal of the project was to devise a system, which by the means of the GCVR system allows individuals to collaborate with groups (and groups to collaborate with other groups) in a common three-dimensional world by manipulating three-dimensional virtual objects. Specifically, three scenarios of use were defined and the users in each scenario should be able to collaborate with users in the other scenarios as well as users in their own scenario. The scenarios were: The round table scenario, the single-user scenario and the multi-user scenario. In essence, the problem became to merge the concept of collaborative virtual reality with the concept of the round table and the associated scenario of use.

We have analyzed how the GCVR system and the round table concept can be merged both by investigating the theoretical CSCW aspects of collaboration as well as by identifying the technical implications of this. This resulted in the design of the GCVRT system.

During the analysis of the implications of the scenarios of use, we discovered that a system supporting the scenarios mentioned should provide a persistent environment such that both synchronous and asynchronous collaborative work can be performed. However, due to the nature of collaborative work, active elements were not perceived as being necessary - especially not in a general case. The system should mainly be oriented towards the field of work, because then the virtual world would represent that which users are collaborating on and enable them to make changes at will. The system was also classified as being state oriented because of the design of the GCVR system, which it is founded on. Regarding the last two points, the system was made more process oriented and also oriented somewhat towards the work arrangement by designing logging functionality, which could enable users to determine who was responsible for a specific change, as well as recalling the entire work flow.

We will now evaluate the design and implementation of the GCVRT system in relation to each of these concepts:

- **Persistent**
  The fact that all messages must be sent through the server, the fact that if the server is not turned off the state is preserved, and the fact that all the relevant messages are logged in the `Log` module all add to the persistence of the data in the system. However, regarding the implementation we have not yet implemented features for loading the created log, which currently renders the `Log` module inoperable, and therefore in effect the only persistence in the system is that the server maintains the state of the virtual world in the abstract world model.

- **Passive**
  We have purposely not made any active layers since it is not in the scope of this project. The design of the GCVRT system is consciously made very modular, which allows additional modules providing active layers to be implemented if or when the need appears. It will be up to developers who are experts in their specific fields to implement these layers, since they may be very different depending on the specific area of application.

- **Field of Work**

The system is in its foundation oriented towards the field of work because of the abstract world model, which describes the state of a virtual world. The virtual world and therefore the abstract world model are representations of the field of work. The design and partial implementation of the logging function does, however, pull the system slightly towards the work arrangement because it describes who is responsible for making specific changes in the virtual world.

- **State**
  The system is state-oriented in its foundation because of the abstract world model. However, when the Log module is able to play back the log, the system moves towards being process oriented. According to the revised process definition given in section 4.7 the system will be process oriented when this feature is implemented.

In section 9.1 the three fundamental problems are described, which the design had to solve in order to incorporate the round table scenario in the GCVR system. These were PHO-mapping, correct perspective and consistency. Solutions have been designed but due to time constraints, the system has not been fully implemented and a well-defined test has not yet been performed. Once the implementation and tests have been performed, it will also be necessary to verify that the design is able to support transitions between the three levels of collaboration (co-ordination, co-operation and co-construction).

For the system to be usable in practice, it will in addition to the existing design be necessary to design a user-friendly interaction scheme (with menus, for instance) in cooperation with actual users. Also, the GCVRT system should be integrated with VRJuggler or a similar system in order to provide support for a variety of arenas, which each have strong and weak points regarding both the scenarios of use as well as the mode of interaction (deity or mortal mode). Only then will the features designed be usable in actual situations of practical work. The features designed for the GCVRT system are those identified as being common for most application areas. Nevertheless, if users wish to use the GCVRT system in a scenario where further information is required, the information overlay should also be implemented.

At this point, we have proposed a new paradigm for collaborative work, which allows people who are not physically present to participate in meetings on almost the same premises as those who are. In order to enable people to collaborate over long distances in a system such as the one described in this report, collaboration in a local setting must be done in a virtual or at least partially virtual world. When users have been convinced that they can benefit from this locally, that is they have an incentive to put on their HMDs etc., they may choose to do so very often. Once this situation is common, the goal of enabling collaboration between people located far away and people physically present at a meeting is easily obtainable. Collaboration between people in a purely virtual environment as described in [JUV02] is still possible. However, to create a system, which convinces people that the trouble of starting up the system for collaborating is worth it, is a very big task. Such a system must be very mature in all aspects in order to enable users to solve problems effectively. From that point of view, this project only represents the beginning.

BIBLIOGRAPHY

[ACN00] Peter Bøgh Andersen, Peter H. Carstensen, and Morten Nielsen. Means of Coordination. *Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling. Fachgruppe Informatik der RWT: Aachen*, pages 41–61, 2000.

[AKea95] Klaus H. Ahlers, André Kremer, and et al. Distributed Augmented Reality for Collaborative Design Applications. Marts 1995.

[Bar] Daniel Barbiero. Dictionary of philosophy of mind. University philosophy web ring. http://www.artsci.wustl.edu/ philos/MindDict/tacitknowledge.html.

[BMS00] Wolfgang Broll, Eckhard Meier, and Thomas Schardt. The Virtual Round Table - a Collaborative Augmented Multi-User Environment. 2000.

[BN99] Olav W. Bertelsen and Christian Nielsen. Dynamics in Wastewater Treatment: A Framework for Understanding Formal Constructs in Complex Technical Settings. 1999.

[FS] Anton Fuhrman and Dieter Schmalstieg. Multi-context Augmented Reality. Workpaper for the Studierstube Workspace project.

[FSH00] Anton Fuhrman, Dieter Schmalstieg, and Gerd Hesina. Bridging Multiple User Interface Dimensions with Augmented Reality. 2000. Workpaper for the Studierstube Workspace project.

[JB] Jakob Bardram. Design for the Dynamics of Cooperative Work Activities. pages 89 – 98.

[JK02] Jesper Kjeldskov. Lessons From Being There: Interface Design for Mobile Augmented Reality. 2002.

[JUV02] Flemming Jønsson, Jacob Koch Uhrenholt, and Jens Peter Vester. G.C.V.R. - Constructing a Generic Collaborative Virtual Reality System. Technical report, Aalborg University, Denmark, http://www.cs.auc.dk/library/cgi-bin/detail.cgi?id=1010613283, 2002. Unpublished.

[LJD97] Jason Leigh, Andrew E. Johnson, and Thomas A. DeFanti. Issues in the Design of a Flexible Distributed Architechture for Supporting Persistence and Interoperability in Collaborative Virtual Environments. 1997.

[LJDV96] Jason Leigh, Andrew E. Johnson, Thomas A. DeFanti, and Christina A. Vasilakis. Multi-perspective Collaborative Design in Persistent Networked Virtual Environments. 1996.

[MK94] Paul Milgram and Fumio Kishino. A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems, special issue on Networked Reality*, E77-D(12):1321–1329, December 1994.

[RWC$^+$98] Rameshi Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs. The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. July 1998. SIGGRAPH 98.

[Sch94] Kjeld Schmidt. The Organization of Cooperative Work. pages 101–112, October 1994.

[SCSD96] Kjeld Schmidt, Peter H. Carstensen Carla Simone, and Monica Divitini. Ariadne - towards a technology of coordination. November 1996. http://citeseer.nj.nec.com/simone96ariadne.html.

[SR96] Frank Shipman and Brent Reeves. Tacit Knowledge: Icebergs in Collaborative Design. *ACM, SIGOIS Bulletin*, 17(3):24–33, December 1996.

[SS96] Kjeld Schmidt and Carla Simone. Coordination Mechanisms: Towards a Conceptual Foundation of cscw Systems Design. *The Journal of Collaborative Computing 5 1996*, pages 155–200, 1996.

Part IV

APPENDIX

A SPEAK FREELY CODECS

The codecs available in Speak Freely are: ADPCM, GSM, LPC and LPC-10 and of cause the default uncompressed speech transmission. Detailed explanations of the different codecs can be found on the web page of the Communications Research Group, University of Southampton in England[1]. In the following list we will present the main idea in the codecs.

- **Simple**
  When Speak Freely is in simple compression mode the compression is lossy, since it ignores every second sample, and thus *simple* provides a reduction of the required bandwidth by a factor of 2.

- **ADPCM and ADPCM+Simple**
  In Adaptive Differential Pulse Code Modulation the codec finds the difference between a predicted signal and the actual signal. Given that the predicted signal is correct, it can be described with fewer bits than the speech signal. On the receiving end the subdivided signal is added to the predicted signal which gives the reconstructed speech signal.

  ADPCM + Simple is the above codec in which every second sample is not transmitted, and thus halves the bandwidth usage of the codec.

- **GSM and GSM+Simple**
  Global System for Mobile communication - GSM.

  The GSM audio codec describes a compression technique based on physiological characteristics in audio signals. The codec is primarily known for its use in mobile telephones.

  GSM + Simple is the above codec in which every second sample is left out.

- **LPC, and LPC-10**
  Linear Predictive Coding is a codec for audio signals which is based on a model for human speech and therefore only stores the necessary parameters for the model. In this way a very high compression is achieved (however only for speech). The cost is that some of the characteristics in the voice is lost and it has a metallic robot-like cling to it.

[1]See http://www-mobile.ecs.soton.ac.uk/ for further details.

B The Intersense

The information contained in this appendix is taken directly from the site http://www.isense.com/. It concerns the InterSense IS-600 Mark 2 Plus.

IS-600 Mark 2 Plus Precision Motion Tracker

The IS-600 Mark 2 Plus delivers high-fidelity 6 Degree-of-Freedom (6-DOF) position and orientation tracking without the issues associated with other tracking technologies. Utilizing a hybrid of inertial and ultrasonic sensing technologies, the IS-600 Mark 2 Plus achieves performance and robustness superior to any single-technology tracking device.

The Mark 2 PLUS offers millimeter resolution, improved stability, and increased noise immunity from environmental interference. The Pentium processor allows four fusion mode stations to track simultaneously at 180 Hz. Hardwired SoniDiscs provide maintenance free operation with a battery powered option available for configuration flexibility.

Superior Accuracy and Robustness

The IS-600 Mark 2 Plus uses InterSense's SensorFusionŹ algorithms to obtain superior orientation accuracy, through accelerometry with ultrasonic drift correction, not just the pure time-of-flight trilateration used by others. This results in vastly improved update rates, resolution, and immunity to ultrasonic interference

Motion Prediction

The IS-600 Mark 2 Plus predicts angular motion up to 50 ms in the future, compensating for graphics rendering delays and eliminating simulator lag. InterSense is the only company to employ the proven benefits of inertial angular rate and acceleration sensors to provide accurate feed-forward motion prediction.

Jitter-Free

The InterSense IS-600 Mark 2 Plus tracker virtually eliminates the simulator sickness from jitter common to other systems.

Fast Response

The InterSense IS-600 Mark 2 Plus provides update rates of 180 Hz with extremely low latency. Tracker-induced lag is removed from your virtual environment.

No Slosh or Drift

InterSense's proprietary micro-machined inertial sensor unit and signal processing virtually eliminates the sloshy response common to inclinometers and the accumulation of drift error that plagues ordinary gyroscopes.

Distortion-Free

The InterSense IS-600 Mark 2 Plus offers smooth, steady response, even in noisy, metal-cluttered environments. Our patented inertial sensing technology is not susceptible to the electromagnetic interference that plagues competitive tracking technologies.