Department of Architecture & Building Science

University of Strathclyde,

Glasgow, Scotland,

United Kingdom

A VR System for the Early Stages of the Design Process in Architecture

GIUSEPPE CONTI

A thesis presented in fulfilment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

October 2002

Acknowledgments

I would like to thank a number of people from ABACUS who contributed in various ways to the work presented in this thesis. First of all I wish to thank Prof. T. W. Maver for believing in this work from the very beginning, for his supervision and for all the resources provided during these three years. Thanks to Malcolm Lindsay for his constant support and help, to Dr. Jelena Petric for her suggestions and enthusiasm and to Dr. Mike Grant for his help with Sgi computers.

I would like also to express my gratitude to those involved in testing of JCAD-VR. My gratitude goes to people of the Design System Group at TU/e Eindhoven, The Netherlands and especially to Dr. H. Achten, Prof. B. de Vries and A. J. Jessurun. Thanks also to those involved in the experiment at the University of Strathclyde including Dr. Jelena Petric, Ross Marshall, Christoph Ackermann and Edward Wright.

Thanks to all others who have provided precious suggestions and feedback during the development of the software including Dr. Wolfgang Dokonal at TUG in Graz, Prof. Jonas af Klercker at LTH in Lund and Dr. Raffaele De Amicis at Fraunhofer IGD in Darmstadt. A special thanks goes to Jose Ripper Kos at UFRJ in Rio De Janeiro for his suggestions about the work and for his sincere friendship.

I want to thank those close and far who supported me with their friendship including Jasna, Popaki, Ombry, GP, Alain, Albertito, Cecotti and Alessandro. A special thanks goes also to Massimo for his friendship and great sense of humour. I also wish to express my gratitude to the members of my family who always supported me with care and love.

Finally a special thanks goes to Giuliana: no words could can describe my gratitude and love.

Abstract

This thesis presents a Virtual Reality (VR) system specifically designed to assist architects in the early stages of the design process. Modern CAD packages are designed for the detail design and construction stages and are not suitable for the creative work typical of the early conceptual process. The use of Virtual Reality systems promotes efficient real-time exploration of design proposals and modern Virtual Reality Aided Design (VRAD) systems allow users to create and manipulate 3D-shapes within the virtual environment and to experiment rapidly with different design solutions. Further, the development of desktop-based VR systems contributes towards the potential use of this technology in everyday practice

The goal of this thesis is the development of a framework for a VR-based collaborative environment. This thesis, together with its companion research (Ucelli, 2002), describes the details of the working prototype called JCAD-VR (Java™ Collaborative Architectural Design tool in Virtual Reality).

JCAD-VR has been designed to provide the user with an effective tool to create basic geometries in a quick and simple way by using mouse commands without the need to type in values. This provides a significant level of abstraction over the rigid mathematical representation that is typical of traditional CAD systems.

This work presents a description of the technical solutions tested during the development of JCAD-VR and in particular it focuses on the interaction techniques used for the implementation of its user-friendly Human-Computer Interface.

Finally, the last part of this work gives the results of an experiment carried out to test the capabilities of the system in a possible scenario. This shows that JCAD-VR, even at prototype level, could be used as an effective tool to create and share design ideas among members of a design team.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AEC	Architecture Engineering Construction
API	Application Programming Interface
BG	BranchGroup
CAAD	Computer Aided Architectural Design
CAD	Computer Aided Design
CAVE	Cave Automatic Virtual Environment
CRT	Cathode Ray Tube
CSCW	Computer Supported Cooperative Work
DAG	Directed Acyclic Graph
DOF	Degree of Freedom
FPS	Frames Per Second
GUI	Graphic User Interface
HCI	Human-Computer Interface
HMD	Head-Mounted Display
HUD	Headup Display
JCAD-VR	Java™ Collaborative Architectural Design tool in Virtual Reality
JMF	Java™ Media Framework
JVM™	Java™ Virtual Machine
LCD	Liquid Crystal Display
NPR	Non-Photorealistic Rendering
OO	Object Oriented
OS	Operating System
OOP	Object-Oriented Programming

POV	Point Of View
RTP	Real-time Transport Protocol
TG	TransformGroup
UI	User Interface
VE	Virtual Environment
VR	Virtual Reality
VRAD	Virtual Reality Aided Design
VRML	Virtual Reality Modeling Language
WIMP	Window, Icon, Menu, Pointer

1 Introduction

1.1 Introduction

This chapter introduces the general aim of the thesis, its general framework and the most important concepts at the foundation of the project.

The research outlined in this thesis addresses the issues concerning to the support of the architectural design process by computer applications. In particular this project investigates the use of software that supports architects at the early stages of the design process, at the so-called conceptual modelling phase. Regarding conceptual design, Shukur (2000) stated, that designing a building or other artefact involves really great effort to produce an original concept. In fact due to its nature, concept modelling is to a certain extent more complex than the later design stages since it has less reference to reality. This can potentially cause many aspects and features of its design to be lost or inadequately handled depending on the subjects' capabilities (Shukur, 2000).

Traditionally many architects have addressed the problem of abstraction that is typical of conceptual modeling by proving their design proposals using physical models. Unfortunately the most accurate 2D paper representations are not usually suitable to explain and transmit the complexity of some architectonic ideas.

Nowadays advances in computer hardware and software have significantly contributed to this issue allowing 3D virtual modelling and animations to be used daily in the architectural practice. Similarly the relatively recent introduction of Virtual Reality (VR) has represented another step towards a fully comprehensive simulation technology. The next section will discuss the introduction and application of VR to the architectural design process.

1.1.1 Virtual Reality as a new Means for Designing Architecture

Although today the use of the *third dimension* has become a daily practice, the *CAAD community* is only now experiencing the move from static representation,

based on 2D renderings or pre-recorded animations, to dynamically generated 3D representations. Real-time navigation and interaction, typical of Virtual Reality environments, provide the fluent interface and that facilitates the exploration of the design proposal, that is the main omission of all the CAD packages commonly in use.

Furthermore the access to desktop VR applications makes them a feasible approach in everyday practice thanks to the increasing growth of computational resources and hardware power. Moreover, the recent growth of network-based virtual communities and the use of avatars have brought a new level of complexity to the meaning of *virtuality*, providing the technology for remote presence and collaborative experiences.

The use of VR in design broadens the boundaries of traditional perception by providing experiences of worlds not necessarily real or material. Therefore it gives the user the freedom to simulate and eventually to build up knowledge and skills dangerous or too expensive for human beings to acquire. In an architectural context the use of Virtual Reality provides the designer with an appropriate, quick and practical feedback that facilitates the search for design solutions. In fact, due to its visualisation power, it enables the capturing of more information than would be possible with the use of the traditional media and it makes the checking of the design solutions more efficient by enhancing simulation capabilities. Furthermore VR provides a natural and user-friendly interface between practitioners and clients enabling them to check the functionality of the design and ensuring that the design meets the clients' expectations. Consequently VR could become the ideal simulation medium for architects investigating design solutions and it could contribute to the production of a better-built environment by addressing sustainability through environmental simulations and appraisal, and engaging design creativity through immersive design.

Thanks to all these advantages it is highly predictable that in the near future VR will become the interface for the next generation of Computer Aided Design (CAD) applications, the so called VRAD (Virtual Reality Aided Design) systems thus promoting VR from a mere presentation medium to a more powerful and

effective design tool. Moreover current research interest is in multidisciplinary working activities and collaborative networks which broaden the concept of VR itself through the development of multi-user applications that allow several remote participants to interact within the virtual environment and to accomplish, collaboratively, complex design tasks.

1.2 The Aim of the Project

The importance of VR technology with its revolutionary shift in visualization and the effect that VR-based applications could have for the architectural design process is stressed by Dagit (1993, p. 514) who states that "architects, as a group, are more aware than most of the profound impact that Brunelleschi's invention of perspective had on society following the fifteenth century. Perspective initiated a fundamental change in the way humans perceive themselves and their environment. Some are looking to virtual worlds as a similar key to opening up new levels of human perception."

In particular, the identification of the role of VR and its place within the practice of architecture has been the subject of a previous experience of the author published at the UK VR-SIG conference in Glasgow (Ucelli et al., 2000). During the first year of the PhD course the author had the chance to work part-time at the Glasgow office of the internationally renowned engineering consultants company Ove Arup & Partners. This office was particularly interested in coming closer to the new visualization and simulation capabilities allowed by Virtual Reality technology. The time spent working at Arup provided the opportunity to investigate problems and issues about the use of 3D modeling and VR in ongoing projects. This experience gave the chance to work closely with both architects and engineers who were for the first time experiencing VR technology. During a 6 months time a number of 3D models were created and architects and engineers were invited repeatedly to evaluate and discuss the progress of the projects at the Virtual Environment Laboratory (VEL) in the ABACUS unit of the University of Strathclyde, in Glasgow (University of Strathclyde, 2002).

This experience confirmed what other authors had stated (Dorta et al., 1998, p. 144): "the design process is made up of two activities which are graphic ideation or conceptual design and communication [...] the first is a formative process dealing with creating and evolving ideas [active phase]; the second is a descriptive process aiming at presenting to others fully-formed ideas [passive phase] ".

Unfortunately the results of the experience were such that, while proving the huge potential of VR for reducing designs "flaws", they highlighted that the present use of VR is limited only to the last phase of the design process, which Dorta et al. (1998) call the passive phase, when the visualization takes place and the result of the design process is eventually shown to the client. Figure 1.1 resumes the most important phases regarding the development of CAAD and VR models. The schema is the result of direct observation of project dynamics in the real working environment experienced by the author.

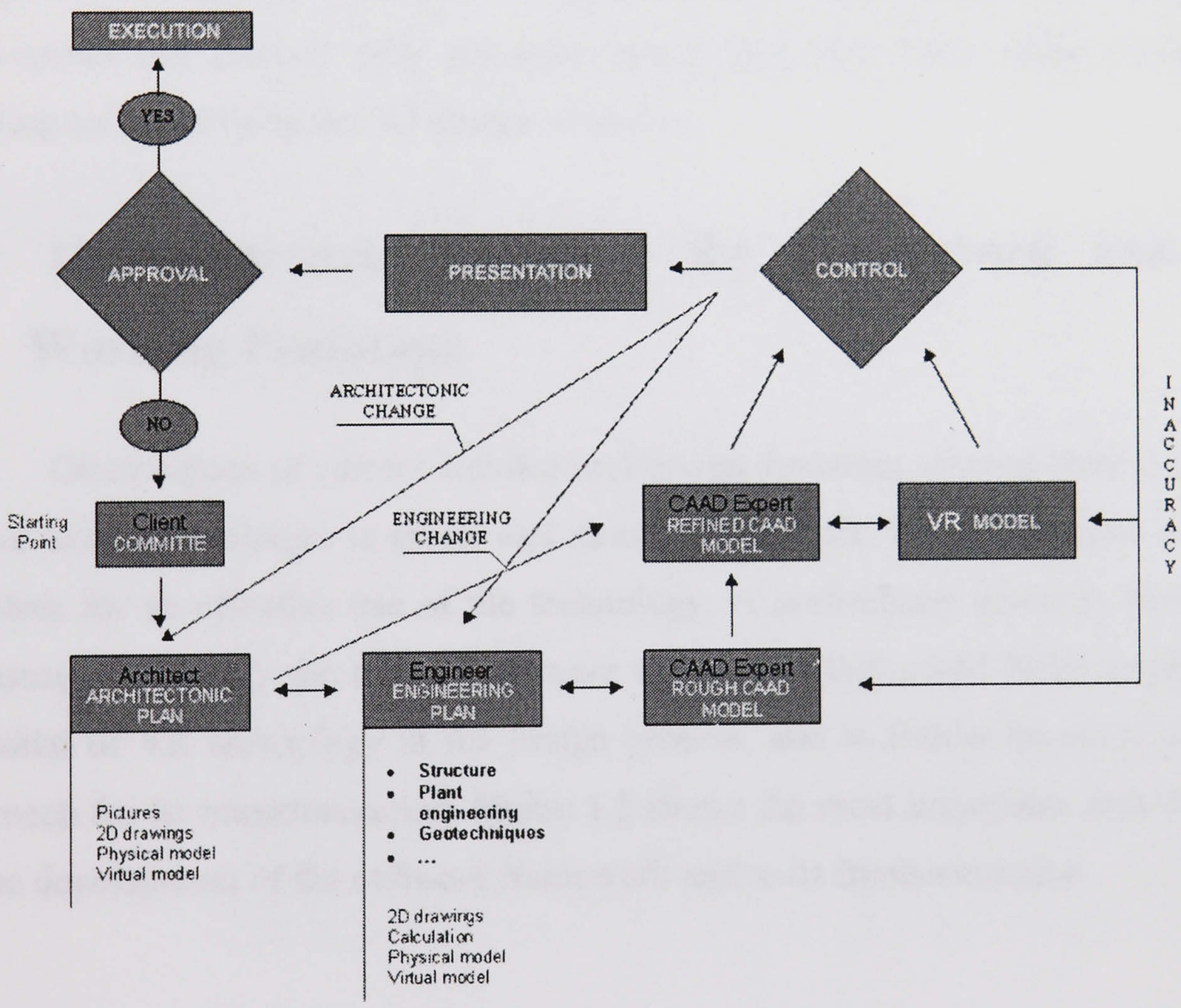


Figure 1.1: The architectural design process from a 3D modelling/VR point of view

It is evident from the graph in Figure 1.1 that the use of VR-based visualization techniques is confined to the end of the conceptual design phase after all the design choices have been taken. This is due mainly to the fact that the models to be visualized in VR have been created using traditional CAD/CAAD packages and not directly within the virtual environment, making the use of VR more time consuming and therefore less effective.

The knowledge gained through this valuable research experience was the base of the concept of the framework reported in this thesis.

The concept upon which the Java Collaborative Architectural Design tool in Virtual Reality (JCAD-VR) framework is founded is to try to anticipate the use of VR within the active phase of the design process thus taking full advantage of VR technology and exploiting its creative potential. The aim is to provide the designer with a tool for creating 3D-shapes in a shared VR environment, thus allowing the design to be shared as it evolves. VR then becomes a new design tool and multi-participants can interact with the environment and each other while discussing, creating and modifying the 3D design solutions.

1.3 The Approach Proposed: the Framework and the Working Prototype

Observations of current architectural design dynamics showed clearly that the use of external packages to create and modify models for VR applications is a key problem for an effective use of the technology. A preliminary research study was necessary to identify the type of software application that could better exploit the potential of VR technology in the design process, and to follow the most suitable approach for its implementation. Figure 1.2 shows the most important steps that led to the development of the software framework and to its implementation.

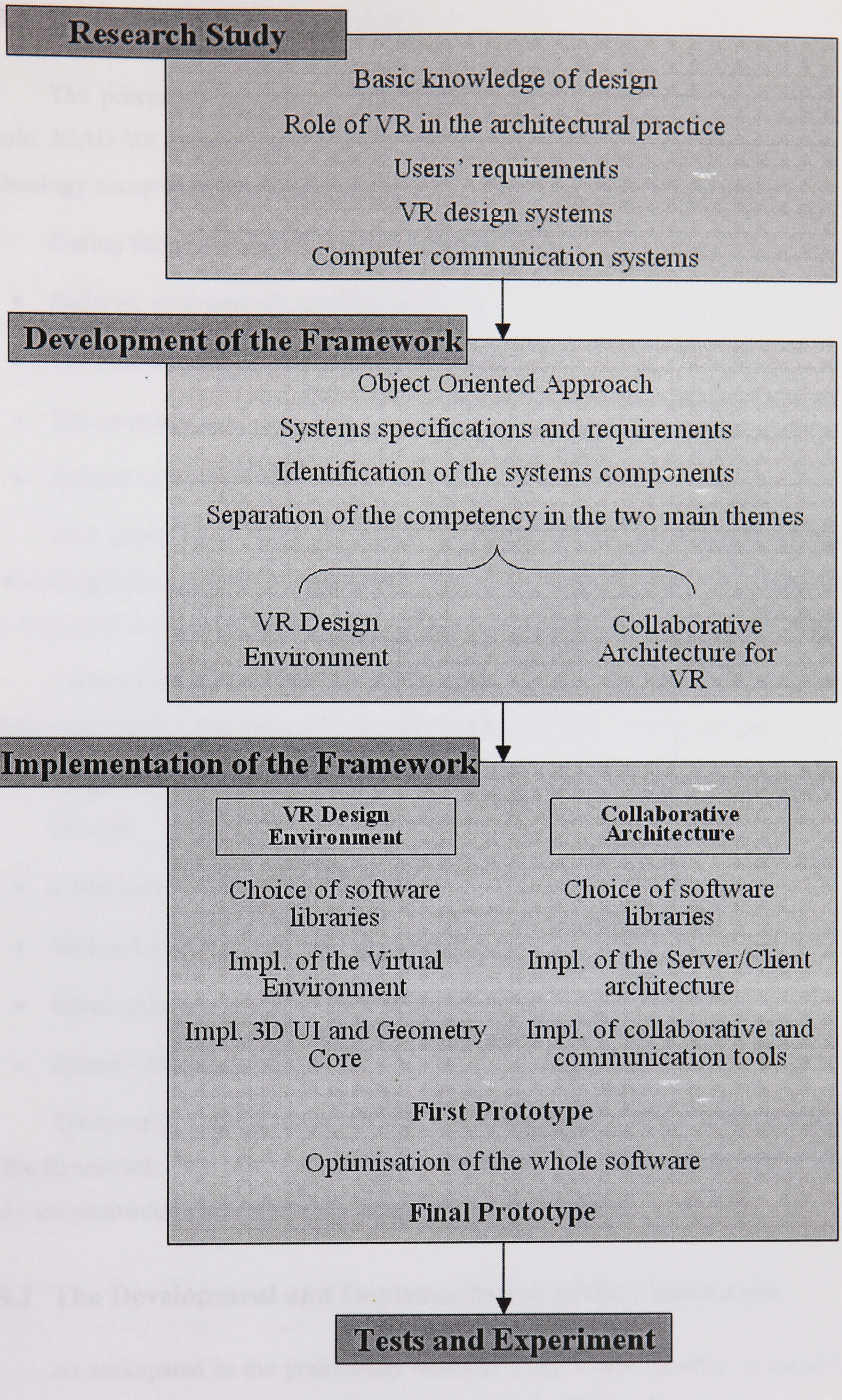


Figure 1.2: Main phases of the project

1.3.1 Preliminary Research Study

The previously mentioned research study provided the foundations of the whole JCAD-VR project and was the opportunity to clarify the role that VR technology occupies in the design process.

During this phase a study was also carried out on:

- Software programming approaches for VR
- The characteristics of VR software packages
- The computer platforms used
- Related technical issues

This preliminary study provided the base for the development of the methodology for the implementation of the first part of the framework. This supports the creation of a virtual design environment for conceptual modelling in architecture.

Furthermore an overview on collaborative applications and their use in the architecture practice was necessary, thus the following topics were investigated:

- Collaboration in the architectural practice and its impact on the organization of work
- Collaborative software architectures
- Software packages and their characteristics
- Communication tools and their implementation
- Related technical issues

These topics address issues relevant to the implementation of the second part of the framework. This part supports the development of a collaborative architecture and communication platform for the virtual design environment.

1.3.2 The Development and Implementation of the Framework

As anticipated in the preliminary research study it was possible to identify two independent, but related, research themes within the JCAD-VR project:

1. The creation of a Virtual Design Environment
2. The implementation of its collaborative architecture

Each theme required addressing with respect to a multitude of theoretical and technical issues and a great deal of attention was paid to the creation of a coherent unique application using independent compatible software modules. Figure 1.3 shows the five main sectors that form the framework of the project: 3D Interface, 3D Geometry Core, Computer Supported Cooperative Work (CSCW) tools, Collaborative Architecture and Database Management. Those sectors marked in red refer to the development of the Virtual Design Environment, those in yellow to the implementation of Collaborative Architecture, including the necessary communication tools and a module for the management of shared databases.

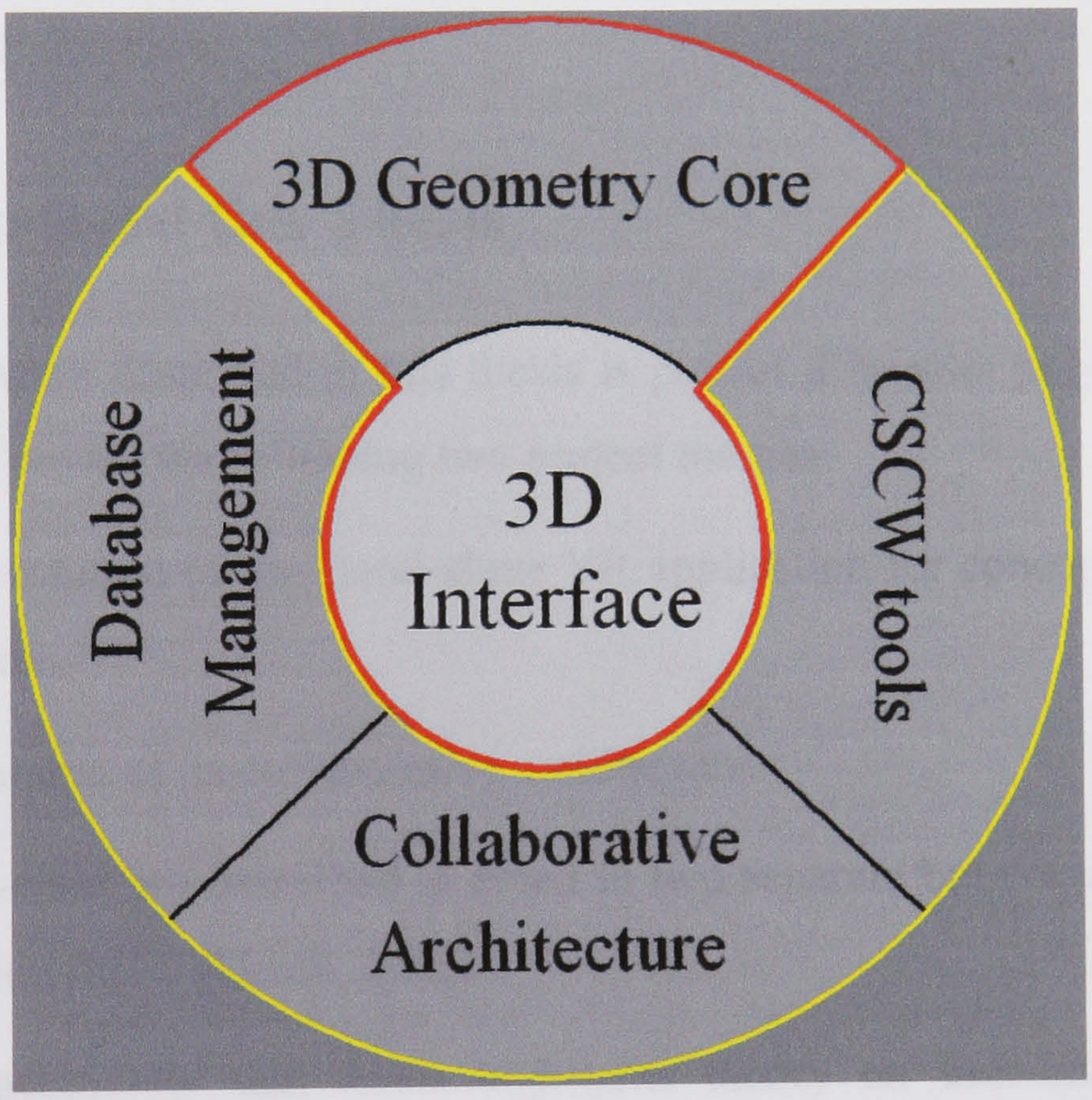


Figure 1.3: Overview of the whole software framework with the two parts marked in yellow and red

Once the general framework of the project was established, each area of interest was independently developed, although every sector was checked against the general framework to ensure compatibility.

The implementation of the two sectors proceeded in parallel as shown in Figure 1.2 until the completion of the first fully working prototype. Then after further software optimisations, the final prototype was developed.

1.3.3 Tests and Experiments

Many unreported tests were carried out during the implementation of the prototype in order to check for software mistakes and bugs in the code. Since the entire project focuses on the early stages of the architectural design process the final prototype was tested in a real working situation involving the creation of an initial conceptual model for a design project.

The experiment that involved students of architecture will be discussed in detail in Chapter 8. It gave the opportunity to test the stability, ease of use and efficacy of the application.

1.4 How to Read this Thesis

The research described in this thesis is part of a broader project outlined in Section 1.3 that covers the following two project themes:

1. The implementation of the stand-alone VR application for conceptual modelling in architecture.
2. The development of its collaborative architecture.

These themes are described in detail in two separate but coupled theses, this and its companion thesis (Ucelli, 2002).

More specifically this thesis highlights theory on user-interfaces in VR environments and it describes the methodology followed to implement the stand-alone Virtual Reality application, called JCAD-VR, specifically designed for the early stages of architectural design. The companion thesis (Ucelli, 2002) addresses issues related to communication and collaboration activities in architectural practice and it outlines the most important phases of the development of the collaborative architecture of JCAD-VR.

Each thesis can be read separately, since each one covers different aspects of Virtual Reality applications, or read together as part of a broader project. It is nevertheless highly recommendable to read them together in order to gain a comprehensive view of the whole project.

Both theses are structured symmetrically (See Figure 1.4), with four shared chapters.

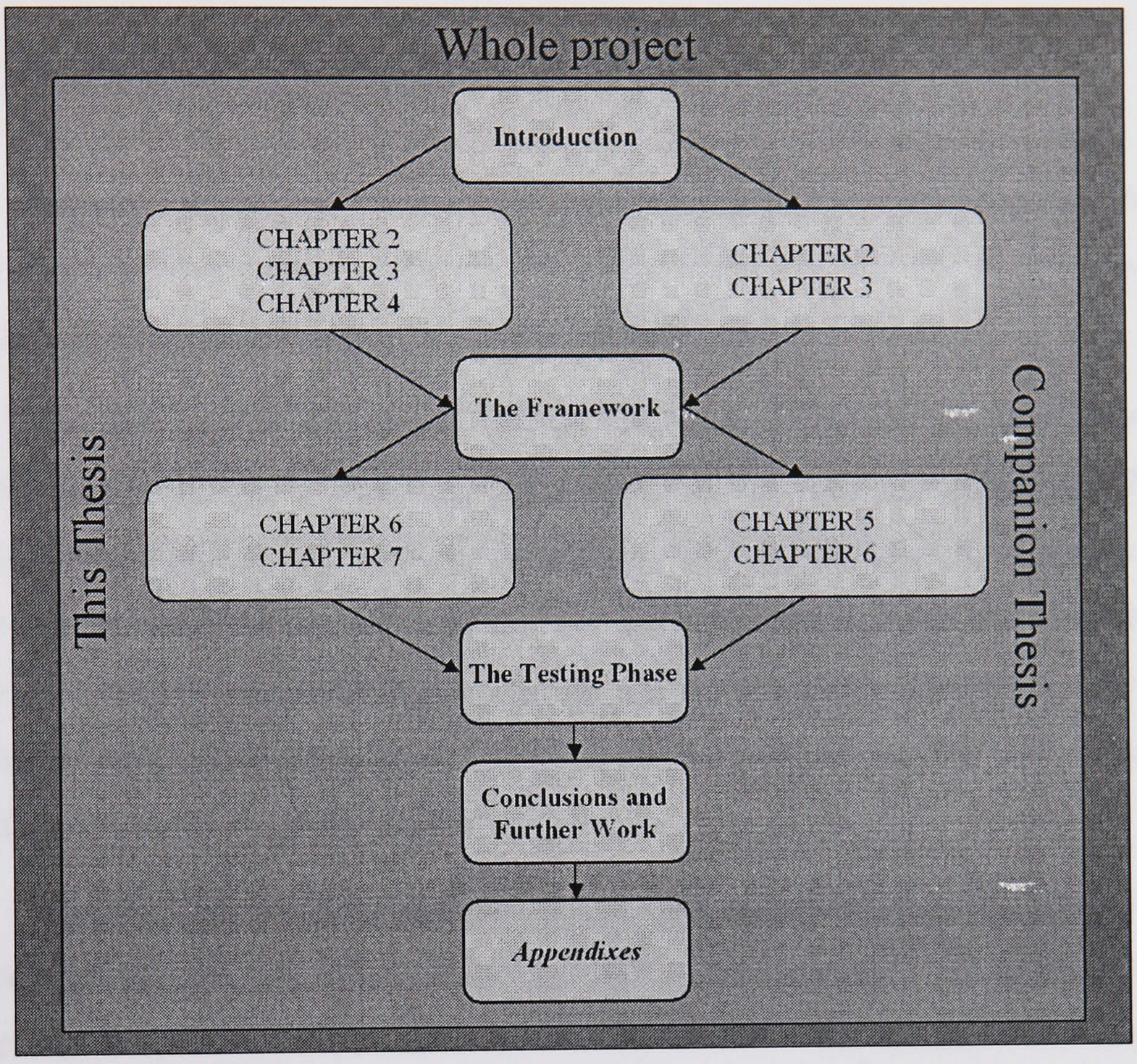


Figure 1.4: The symmetrical structure of the two theses with the shared chapters placed at the centre of the structure

The Chapters number *one*, *five* and *eight, nine* are common and therefore they are included in both theses, since they refer to the general topics that were shared by the two theses. Specifically:

- *Chapter One* provides the overall introduction to the project
- *Chapter Five* describes the general structure of the framework and of the JCAD-VR prototype

- *Chapter Eight* highlights the exercises set up for the testing phase of the prototype
- *Chapter Nine* draws the conclusions and proposes further developments

However, Chapters number *two*, *three, four, six* and *seven* are specific to this thesis and they describe in detail issues and methodologies relevant to their individual work.

1.5 Outline of the thesis

Section 1.4 clarified the symmetrical structure of the two theses covering the whole project. This section will provide a more detailed overview of the content of this particular thesis describing its chapters individually.

The first part of this work outlines the body of knowledge upon which the research was carried out. The second part presents the theory proposed and from a technical side it will show the framework and a working prototype tackling the issue of the use of VR in the early stages of the design process. After the first chapter, where an introduction to the whole project and individual research is provided, Chapter 2 will present an historical overview of Virtual Reality and propose a classification of the systems available.

Chapter 3 proposes the idea of VR as a new means of designing rather than as a mere visualisation tool. This chapter proposes a taxonomy of VR applications for design and offers an overview of the systems available in the scientific literature.

In Chapter 4 the metaphor of the interface is studied: this provides an introduction to the field of Human-Computer Interfaces (HCI) and reports on issues related to the development of a 3D HCI. This chapter gave the opportunity to study in detail issues such as space, way finding, navigation, control, manipulation and feedback.

Chapter 5 is an introduction to the second and more technical section of the thesis: it outlines the idea on which the proposed framework is based, it also justifies the methodology used and the techniques chosen. This chapter introduces the JCAD-VR software prototype as a tool for designing within the Virtual Environment (VE).

Its many advantages, such as its two main units, its modular structure and its Object Oriented nature, are presented and justified. This chapter introduces the two following ones, which describe in detail the technical implementation of the stand-alone JCAD-VR prototype and its 3D-Unit.

Chapter 6 presents the 3D-Unit, the part of the client application that deals with all the issues related to the creation and visualisation of the virtual world. This chapter presents the proposed metaphor of the 3D-GUI, where the interface becomes part of the virtual world and so can be manipulated like any other object in the virtual scene. The second part of Chapter 6 outlines an overview of the HCI unit, it gives the approach followed to implement the chosen metaphor, and describes the results obtained.

Chapter 7 presents in detail the implementation of the section of JCAD-VR designed to provide the user with a tool to create simple geometries in a rapid and intuitive way. This chapter describes how this section works, outlining an overview of the package together with the design patterns followed.

Chapter 8 is a common chapter and describes the working scenario that has been set as the benchmark for evaluating the efficacy, ease of use and stability of the software. The experiment involved multiple-users interacting concurrently on a proposed design task. Reports on the outcome of the experiment will be presented.

Finally Chapter 9 draws the conclusions reached during the implementation of the JCAD-VR prototype and it proposes further developments inside and outside the context of the proposed framework.

1.6 Conclusions

This first chapter gave a general introduction to the aspects to be covered in the rest of the thesis. It highlighted the structure of the JCAD-VR project especially, showing how this was developed by the combined effort of our research.

As shown the coupled theses have some common chapters in order to make them easier to read. The introductory and concluding chapters are common ones, while the other chapters will address only the research literature relevant to their

aspects of the project, specific issues and different parts of the implementation of the JCAD-VR system.

Chapter 2 will introduce more specific issues relative to Virtual Reality. It will present an historical overview of Virtual Reality and it will propose a classification of the systems available.

2 An Historical Overview of the Role of Virtual Reality within the Design Process

2.1 Introduction

In recent years the term Virtual Reality (VR) has become an extremely popular idiom to describe sophisticated computer-based simulation set-ups, probably due to the success of a number of science fiction titles. Today VR is a complete and mature technology which is used across many fields ranging from entertainment to engineering. In particular, VR is appreciated as a safe and relatively inexpensive technique which can be used to train people in situations which are hard to reproduce or are potentially dangerous.

This chapter will present an overview of the technologies used in most VR systems. The chapter will first outline a number of definitions to be found in scientific literature then in the second part will present an historical overview of Virtual Reality. This second part will first introduce two of the mechanical simulation set-ups which are considered the precursors of VR: the architectural endoscope and the pioneering immersive cinema which first used stereoscopic images and localised sound.

The remaining part of the chapter will focus on computer-based systems. Starting from Sutherland's seminal work the chapter will show the development of the hardware and software over four decades, from the mid-sixties to the present day. Particular emphasis will be given to the research at NASA which inspired generations of researchers over the following years. This final part will highlight the development of the first PC-based VR set-ups and their role in the spread of the technology.

Finally, the last part of the chapter will present a classification of modern VR systems. The taxonomy will be done according to the level of "immersion" of the VR system and three main categories will be presented: non-immersive, semi-immersive

and fully immersive. The classification will highlight the advantages and weaknesses of each type of technology.

2.2 What is Virtual Reality?

Myron W. Krueger who is generally considered one of the fathers of Virtual Reality (VR) points out (1994, p. xi) that in recent years, mainly due to the heavy impact of science-fiction titles, the term *Virtual Reality* has become the most popular way to describe the ultimate simulation experience. Other synonyms like Artificial Reality, Simulator Technology or Synthetic Environments have not caught the attention of the media to the extent of the idiom coined by Jaron Lainer (Pimentel et al., 1995).

In fact the media industry, as Burdea et al. observe (1994, p.1), has shown a great deal of interest in VR: magazines such as Business Week (Hamilton, 1992), Time (Elmer-Dewitt, 1993), Newsweek (Kaplan, 1993) and TV channels like CNN, NBC, CBS have all given "prime-time coverage to Virtual Reality reflecting a surge in interest by the general public" (Burdea et al., 1994, p. 1).

The success of the expression *virtual reality* is to be found, according to Negroponte (1993) within the semantic nature of its idiom: "if prizes were awarded for the best oxymoron, *virtual reality* would be a winner". Negroponte notes that paradoxically the term, is being turned from an *oxymoron* into a kind of *pleonasm*. In fact, if the term "virtual reality" is read not as a noun and an adjective but as "equal halves" its meaning suddenly becomes, according to the author, something that "makes the artificial as realistic as the real" showing its undoubted *pleonastic* nature.

Several other authors have attempted to provide a convincing definition of VR (Burdea et al., 1994; Loeffler et al., 1994; Aukstakalnis et al., 1992). Although it can be certainly stated that what VR is actually about is far from what Gibson's main character in Neuromancer experiences (Gibson, 1984), it is still complicated to explain what VR has become today. Unquestionably the term *cyberspace*, coined by Gibson in the science fiction novel has something of the visionary, the "alternative computer universe in which data exists like a city of light" (Pimentel et al., 1995, p. xix) but it obviously does not represent a comprehensive scientific definition.

A more scientific approach is proposed by Sherman et al. (1992, p. 17) who write, "Virtual Reality allows you to explore a computer-generated world by actually being in it".

Some people consider VR an assortment of technologies: as Burdea et al. (1994, p. 2) observe, "the general public now tends to associate Virtual Reality simulations with head-mounted displays and sensing gloves". But as Sherman et al. (1992, p. 24) point out, VR is neither a standardised industry nor a mere collection of technologies. Of the same point of view is Krueger (1994, p. xi) who states: "Virtual Reality is more than a sum of its components, it is fundamentally a *system* technology". In fact, as Burdea et al. (1994, p. 2) remark "Virtual Reality can be done without head-mounted displays by using large projection screens or even desk-top workstations [...]. Similarly, gloves can be replaced with much simpler trackballs or joysticks [...]. Therefore describing Virtual Reality in terms of the tools it uses is also not an adequate definition".

Burdea et al. (1994) expands the notion of VR including the possibility of involving all the senses, "Virtual Reality is a high-end interface that involves real-time simulation and interactions through multiple sensorial channels. These sensorial modalities are visual, auditory, tactile, smell, etc." (Burdea et al., 1994, p. 4). Likewise Chan et al. (1999a, p.44) state that VR "provides diversified media for visually, aurally and interactively experiencing activities and behaviours conducted in the cyberspace". In this regard, Loeffler et al. (1994, p. xx) point out how cognitive sciences have proved how the learning experience benefits from the implementation of multi-sensorial, or *multimodal* (Reeves et al., 2000) input such as that coming from an immersive VE. Finally, the approach followed by Burdea et al. (1994) also represents an important shift in the VR paradigm from previously considering it as an *interface*, a new way of interacting with computers, rather than simply as an *environment* or a mere collection of technologies.

Other authors shift the focus onto the experiencing of a simulated environment like Pimentel et al. (1995, p. xix) who suggest that VR is "an immersive experience in which participants wear visually-coupled displays, view stereoscopic or biocular images, listen to 3-D sounds, and are free to explore and interact within a

3-D world". Of similar opinion are Loeffler et al. (1994, p. xiv) who write that VR "is a three-dimensional, computer-generated, simulated environment that is rendered in real time according to the behavior of the user".

This last definition in particular, stresses two major features: the three-dimensionality of the simulation and its *real-time* nature. Regarding the former it is certainly true that VR has brought the user far beyond the one-dimension command line of the early Operatiing Systems (OS) or the two-dimension Graphic User Interfaces (GUI) of modern OS. As the authors note (Loeffler et al., 1994, p. xiv) "while a graphical user interface is like a window on a computer application, virtual reality interfaces are like portals that allow the user to step into the application".

But the definitions mentioned so far have not described one of the major features of modern VR systems, the *interactivity*, that is emphasised by one of the most frequently occurring definitions in the literature: "Virtual Reality is a way for humans to visualize, manipulate and interact with computers and extremely complex data" (Aukstakalnis et al., 1992, p. 7). This definition, which transcends technical solutions and gives a more comprehensive description of the idiom, will be used throughout this thesis.

Aukstakalnis and Blatner focus attention on the three activities that take place in modern VR systems: *visualisation*, *manipulation* and *interaction*.

The first takes place when an environment, either similar to the real world or abstract, like a representation of a database or a scientific simulation, is shown by visual devices of several types. The capacity to visualise seriously affects the level of *immersion* of the system and an overview of the different levels of *immersion* achieved by different visualisation devices will be presented in one of the following paragraphs (See Section 2.4). As Pimentel et al. (1995, p. 19) note, "*immersion* means to block out distractions and focus selectively on just the information with which you want to work".

The second activity, the *manipulation*, takes place when the user controls the state of the environment by means of some input device and changes the state of the system according to their wish. This feature, which marks a fundamental difference between VR and other visualisation media, gives the user the power to *manipulate*

the data in real time without necessarily following a pre-recorded path or animation script.

The last and probably most important activity that differentiates modern VR systems from those of the first generation, is the *interaction*. As Aukstakalnis et al. (1992) state, the virtual world can be static or interactive. Furthermore interaction can be generated either in real time, as an effect of some user's action, or it can be the result of a scripted command.

Pimentel et al. (1995) describe two types of interaction: *navigation*, the ability to freely move within the environment and the *dynamics of the environment*. The latter, according to the authors (Pimentel et al., 1995, p. 21), are "the rules for how its contents [...] interact in order to exchange energy or information". This set of rules might change according to the environment being simulated and it could be made of a set of physical forces, for instance in a flight simulator, or the rules could be based on the physiology of the human body, in the case of a training application for medical purposes.

2.2.1 A Real-Time Visualisation Tool

The prime factor on which the majority of the authors agree that they consider the key element for the success of any VR system, is the *real-time* capability: the capacity of a system to show images at a speed sufficient to provide the user with a convincing engagement with the scene being shown. As Sherman et al. (1992, p. 17) state, "you become part of that world, you change it, and the changes occur as you make them".

The absence of a delay between the command given by the user and the consequent response of the system is the crucial element in making the simulation appear a natural experience.

Although the physiological implications behind perception will not be the subject of research in this thesis, understanding its effect is essential in order to properly understand the suitability of some visualisation systems we shall be referring to, to remember that several authors (Gibson, 1950; Hart et al., 1973;

Hubona et al., 1999; Hinckley et al., 1997; Ware et al., 1996) have comprehensively investigated the topic in the last fifty years.

More recently, a field of research has emerged to address the specific topic of interaction between machine and users: Computer-Human Interaction (CHI). Several studies in the field of CHI have shown that the user's perception is related to the cues that a simulation system can provide. Although some authors (Hubona et al., 1999, p. 215) think that "the use of these cues, and particularly, the combinations of these cues, to effectively convey depth information in computer-generated scenes remains an ongoing topic of research" it is acknowledged that "if the cues available to our perceptual systems are close enough to those we are evolved to notice, then all of the same evaluations that we make in real life will be true with media as well" (Reeves et al., 2000, p. 68).

More precisely, it is widely accepted (Weber, 1995) that our perception of reality is deeply affected by the close coupling between the making of a change to the point of view and the corresponding image that is therefore perceived by the human eye. As Weber (1995, p. 77) states perception itself can be considered "a sensory-motor process by which an array of environmental stimuli is transformed into a perceptual image".

Likewise Ware et al. (1996) state that the most important factor affecting the perception of a three-dimensional scene is the relative movements between objects: the so-called *motion parallax*. The authors also state that *motion parallax* "seems to be considerably more important that *stereopsis* in helping us interpret spatial layout [...] This movement allows the brain to integrate spatial information over time, and whereas a stereo display only gives two views to help understand a scene, a scene in which there is relative movement of the head and objects provides a whole continuum" (Ware et al., 1996, p. 124).

Therefore, the speed at which the visualisation takes place and in particular the immediateness of the response by the system to the user's commands, are the key factors to convey the cues necessary for the consequent *illusion* of reality.

Myers et al. (1996, p. 805) state that the *real-time* capability, "the ability for the system to respond quickly enough to the effect of direct manipulation", is a major

factor when creating 3D interactive applications. Also Negroponte (1993) affirms that the speed at which a VR system is capable of reacting to the user's commands is crucial: "the real issue and challenge in VR today is not the display, but how to reconcile a person's expectations of reality with what current systems deliver in terms of response time". Krueger (1994, p. xii) pursues the same idea when he says that "if a Virtual Reality system does not respond to the participant's movements instantaneously, it does not work".

In fact, the *real-time* requirement implies that if the user decides to turn or to look in a particular direction the system must render the new scene at a frequency close to or higher than 25 fps, the threshold commonly accepted to result in a convincing illusion of movement.

Although the feeling of being real can be enhanced involving other senses, vision and hearing remain the two most important ones. Regarding this, Negroponte (1993) notes that the experience can be enhanced using other means like spatial sound and haptic devices, but since "we grow up in a world that fosters immediacy in action and reaction [...] in VR, the frequency response of the system will be almost all that counts". What he calls *spatial* and *temporal aliasing* are vital for a natural experience of a simulated environment. In other words "aliased VR is the oxymoron while VR itself will be the pleonasm" (Negroponte, 1993).

The importance of vision to perception, and the consequent importance that the *real-time* behaviour acquires, is evidenced by Reeves et al. (2000, p. 67). They state that however sophisticated computer simulated systems may become they "will still be primarily about the senses of sight and sound, not because other senses won't be added to machines, but because sight and sound dominate human perception".

At this point it is perhaps worth noting that the concept of *real-time* technology is, strictly speaking, technically not achievable: there is simply no system capable of responding without delay to the user's commands. This *latency*, which is the time needed by the system to respond to the user's command, will always have a value greater than zero. More realistically then, the ideal system should have a *latency* shorter than the time necessary to the system to render the following frame. But since the *latency* is a function of the amount of data that has to be calculated by

the system every second, its value is not a constant and therefore every system has a certain threshold representing the amount of information that it can handle giving a satisfactorily prompt response. This is why some authors like Myers et al. (1996, p. 805) prefer to refer to this feature as the "near-real-time" capability.

Nevertheless, it is commonly accepted within the research community working in the broad field of computer graphics that, if a system is capable of responding to the user's command within an interval that is short enough to provide a feeling of natural, smooth movement, that system is working in *real-time* and this necessarily rather flexible definition will be adopted throughout this thesis.

In other words, as Krueger clearly says, "if the responses follow your action instantaneously, you feel a real sense of cause and effect. The experience is real. That speed requirement is a cognitive imperative. If you have to wait for the response, you're distanced from the experience. Either you make it, or you are not doing it. Period. You don't show slides and say you are showing a movie" (Morgan, 1994, p. 174).

2.3 A 40 Years Long History

The desire to re-create an experience through some kind of artificial environment has always been part of the human wish to simulate the world around us. All the figurative arts for instance, could be considered as attempts to re-create reality through artificial means.

As Pimentel et al. write (Pimentel et al., 1995, p. 25) the Greeks theatre of the 4th and 5th centuries B.C. was a *simulation* that "gave spectators new points of view by putting them in the shoes of people not unlike themselves". A relevant example of *surrounding experience* was Robert Barker's 360-degree painting of the city of Edinburgh (Pimentel et al., 1995). This 18th century first example of a *panorama*, as the Scottish painter named it, was a successful effort to heighten the feeling of realism. One century later Wheatstone invented the stereoscopic display: two images, which were slightly shifted, were mounted in a viewer to give the user true sense of depth (Pimentel et al., 1995).

2.3.1 Precursors of VR

If we restrict our analysis of the precursors of VR to modern times and to electro-mechanic devices, two prime examples emerge. The first is in the field of architecture and is the endoscope, a visualisation tool used to navigate models of urban environments. The second is *immersive* cinema, which was first developed in 1946 in the US.

Both devices aim to create a system that can enhance the experience of an environment and to re-create the illusion of that environment which is as close as possible to the natural way we perceive it.

2.3.2 Architectural Endoscopy

The need for simulation of urban spaces has always been present among people working within the built environment. Before the arrival of computer technology visual simulations were often achieved with complex opto-mechanical devices called architectural endoscopes.

Although a comprehensive insight into architectural endoscopy is outside the scope of this thesis, it is worth presenting a short overview of its origins and the influence it had on modern visualisation systems for architecture.

Architectural endoscopy is considered a "unique medium for exploration and representation of architecture and space [...] a platform for experimentation, research, communication development, user participation" (EAEA, 1995). This definition, that might well suit modern VR systems that are being used in architecture, shows how the two approaches share the same ultimate goal: the visualisation and experience of architectural spaces.

One of the first prototype endoscopes was the Urbanoscope installed in 1976 at the Landbouwhogeschool in Wageningen, Holland (Smardon et al., 1986). To give the observer the illusion of walking through the environment a servo-controlled periscope transmitted the image of a physical model onto a TV screen. The visualisation system was simply based on a black and white micro camera mounted on a mechanic arm. The control system was set in a second room where the user,

with the help of pedals and a steering wheel, could *drive* the micro camera through the model. The feeling of immersion in, and the simulation of the environment was ensured by a screen placed in front of the user where the images captured by the camera were projected.

After that first experimental device other universities, like Berkley, Vienna and Lund, started to analyse the advantages of such simulations in providing the feeling of a built environment.

The results of the research proved (Smardon et al., 1986) that these first examples of simulators were more effective in terms of the level of comprehension of the environment achieved if compared with the use of traditional visualisation techniques such as plans, sections, perspective views and physical models.

The study revealed that the *naturalistic model* (Smardon et al., 1986) was the most effective approach with the best results achieved when filming from eye what would be level. Moreover the system proved how familiarity with the user interface, which followed the metaphor of a car, made the relationship with the simulator extremely easy.

Some authors even compare the use of this technology to modern VR systems, like Stellingwerff (1999, p. 491) who states: "by means of an endoscope you can get an eye-level image of an architectural scale model. New medical optic endoscope tubes and small video cameras, good scale-models and direct interaction of movements can make architectural endoscopy comparable to an advanced Virtual Reality (VR) system". This point of view can be easily questioned if we do not limit our scope to simulations of urban landscapes, and we take into account all the physical restrictions endoscopy has to deal with. In fact, it is impossible to get the same interaction and ability to manipulate the environment as can be achieved by computer-based simulations.

Nevertheless, today architectural endoscopy is quite a mature research area and researchers have started speculating on the convergence of *optical endoscopy* and computer-based simulation or, in Stellingwerff et al.'s words (1995), *digital endoscopy*.

2.3.3 Immersive Cinema

During the 1950s the American cinema industry started to explore the possibility of improving the traditional way films were experienced. Cinematographers who were aware that our perception could be greatly enhanced if the a wider part of the viewer's visual field was engaged, tried to developed new forms of *immersive* cinema. In fact, as Gibson writes (1950, p. 27) the visual field "extends about 180 degrees laterally and 150 degrees up and down" and traditional cinema or TV sets only cover a small part of it.

During the early 1950s, Fred Waller tried to address this issue by developing a system called Cinerama, a precursor of today's IMAX systems and other modern semi immersive visualisation environments. The system used three synchronised 35mm cameras to record the scene, and then it had to be projected in special theatres with a 180-degree horizontal view. Unfortunately, the high cost of filming and synchronised playback together with the cost of such specially outfitted theatres doomed the spread of this system.

At the end of 1960 a cinematographer called Morton Heilig patented the idea of a device for simulating 3D environments, a predecessor of the modern Head-Mounted Display. The system would use 3D-slides, stereo sound and a capability to include smell but the revolutionary idea did not meet the enthusiasm of investors and the project was left undeveloped.

Two years later, 1962, Heilig patented another device that could exploit the full field of view of the spectators. Heilig himself stated "why stop at a picture that fills only 18 percent of the spectator's visual field and a two-dimensional picture at that? Why not make it a three-dimensional image that fills 100 percent of the spectator's visual field accompanied by stereophonic sound?" (Heilig, 1992. In Burdea et al., 1994, p. 6). The idea, which was certainly visionary for that time, did not attract any financial help and the only prototype he realised was the "Sensorama Simulator" (See Figure 2.1).

In this early 60s device, the user could experience a ride on a motorbike through a 3D video created by three 35mm video-cameras, stereo sound, a wind

effect simulated by a fan and potholes in the road felt through a moving seat. As Burdea et al. (1994, p. 7) recall, the user "could even smell food when passing by a store". Certainly the machine was rather primitive if compared with modern devices but it opened the door to a brand new field of research.

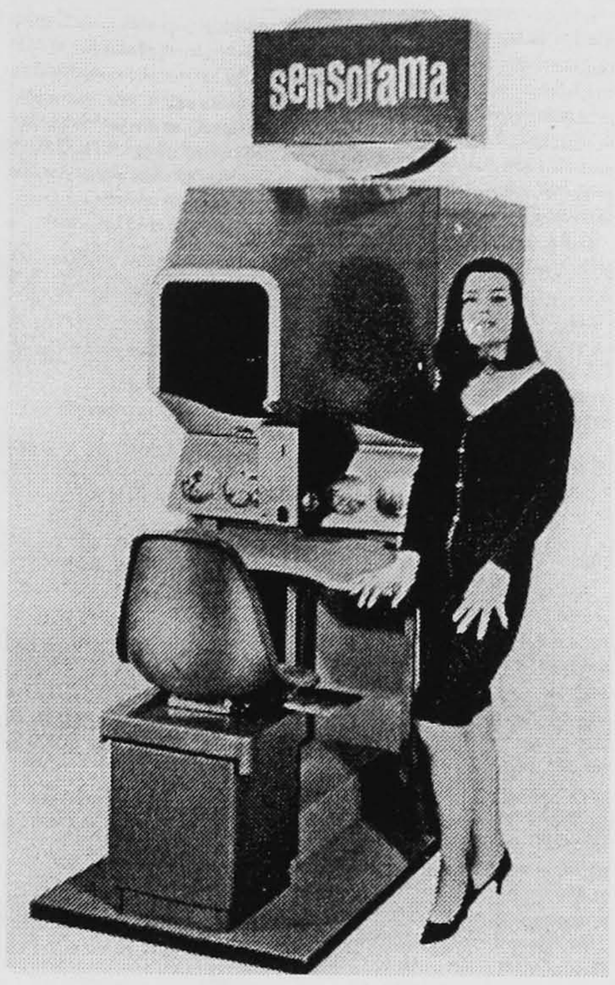


Figure 2.1: The "Sensorama" machine (Burdea et al, 1994, p. 8)

2.3.4 The Advent of Computer Technology

The history of VR, if we restrict the analysis to computer-based media, is about 40 years long, which is quite a long time when compared to the history of computer technology itself.

The arrival of computer technology, as Loeffler et al. noted (1994, p. xv), has facilitated a "necessary leap of imagination" and it has eventually provided society with a more effective tool.

A few years after Heilig's revolutionary system, Ivan Sutherland, commonly considered one of the fathers of computer graphics, published a paper called "The Ultimate Display" (Sutherland, 1965), a milestone in VR system technology. In his famous paper, Sutherland outlined a visionary future for computer visualisation where the user could look at a screen connected to a computer to see a visual representation of a mathematical model. Sutherland wrote: "the ultimate display

would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked" (Sutherland, 1965, p. 508).

2.3.4.1 Sutherland's Sword of Damocles

During the following years Sutherland, while at Harvard and then at the University of Utah, designed the first *Head-Mounted Computer Display* (HMD). Two Cathode Ray Tube (CRT) screens projected a simple wireframe representation of a cube onto half-silvered mirrors placed in front of the user's eyes. That allowed the user to see the computer-generated images overlaying the real world. The system was not truly *wearable* due to the absence of any degree of miniaturisation in the CRT technology. In fact the entire HMD had to be supported by a mechanical arm (See Figure 2.2) and for this reason it quickly earned the nickname *the Sword of Damocles* (Pimentel et al., 1995). The mechanical arm "had potentiometers that measured the user's view direction" (Burdea et al., 1994, p. 7) and the user could move their head to see different sides of the cube.

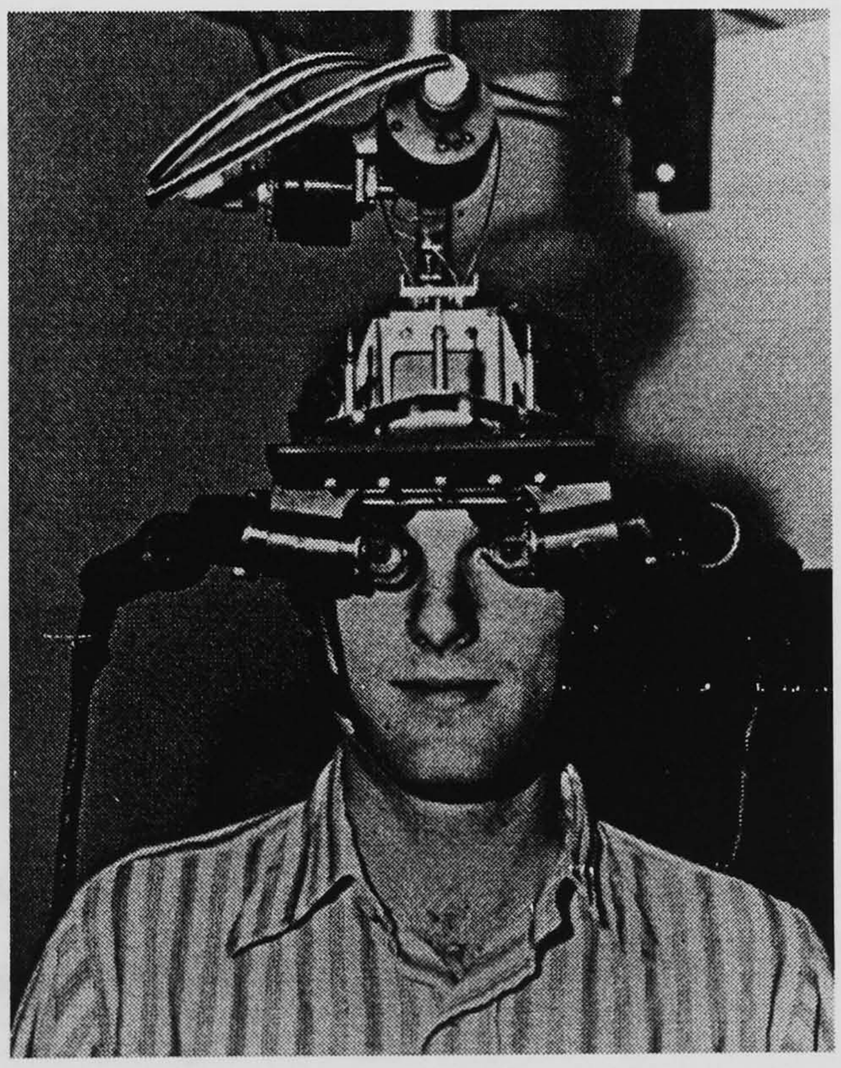


Figure 2.2: Ivan Sutherland's *Sword of Damocles* (Zampi et al., 1995, p. 20)

Sutherland's *Sword of Damocles* was the first example of a three-dimensional experience without analogue cameras and as Pimentel et al. (1995) wrote "in general terms, there's no difference between this system and the system that was to emerge from NASA Ames almost 20 years later. The long search for a synthesized reality had finally led to a system capable of generating virtual objects. [...] The barrier between the computer and the user had finally been eliminated; the user was now inside the computer" (Pimentel et al., 1995, p. 44).

2.3.4.2 Krueger's Metaplay

A few years later Myron Krueger coined the term Artificial Reality. He was a scientist interested in the field of human-machine interaction who happened to become the first computer artist. Krueger wanted to create a system to let people interact with a computer without needing to be programmers. He wanted "people to walk into a room, have a brief pleasant experience, and leave knowing that computers could be playful, creative, vested with the humanity of human programmers" (Morgan, 1994, p. 172).

After finishing his PhD thesis with the title "Computer-generated responsive environment" he prepared an installation called Metaplay. When the user entered the room where Metaplay was installed they had the feeling of being in a computer-generated environment. The effect was faked by having a camera filming a computer screen and projecting the image onto a wall together with the silhouette of the person participating.

As Krueger reported in an interview to Morgan (1994, p. 173) "when people entered the gallery, they saw their images projected life-size in front of them. They saw computer-graphic graffiti appearing on their images: I was drawing on their images with a data tablet a mile away. When they got the idea, they would duck when they saw the cursor coming toward them. Or bat it away. When I put a graphic ball at the top of the screen, they would reach and hit it, and I would move it across the screen. Sometimes I would put my own image on the screen to interact with them".

As Pimentel et al. (1995) noted, in Metaplay the users could watch themselves projected into the virtual environment as opposed to traditional VR systems where the user sees the environment through the "eyes of a computer". For this reason they called the environment created by Krueger "mirrorworld" (Pimentel et al., 1995, p. 12).

But Metaplay and a few other examples of *projected reality* like the Mandala Machine developed by the Vivid Group (Pimentel et al., 1995) were, as Sherman et al. (1992) noted, quite ahead of their time and Krueger had to spend a few years in academic research before a commercial application of VR would become available.

2.3.4.3 The Role of NASA

The following years were characterised by intense research activity. People at the Media Lab at the Massachusetts Institute of Technology investigated possible new interfaces, feedback and graphics. The Aspen project (Negroponte, 1996) was the result of this effort. The system, initially developed for military purposes, was based on a complex set of videodiscs that allowed interactive navigation of the town of Aspen. Military institutions and the flight simulator industry understood the potential of these first attempts and they became the two major sponsors of the first research into VR.

In 1982 Thomas Furness III developed a control system for the US army called Visually Coupled Airborne System Simulator (VCASS), which was used to train the pilots of jet fighters. Through a helmet the pilot would see a high-resolution wireframe environment (Pimentel et al., 1995).

At the same time Frederick Brooks at the University of North Carolina (UNC) started exploring the field of haptic interaction. He wanted to create a system that would allow chemists to handle and manipulate molecules. The first control device called GROPE-II was ready by 1971 but, due to the limits of computer technology available, it was only in 1986 that very basic computer generated images could be used to visualise the result of the interaction.

This system called GROPE-III could be used to feel the docking forces between molecules. As Pimentel et al. (1995, p. 59) recall, "chemists weren't the

only ones to have their minds amplified at UNC. Architects were also among the chosen few. For centuries, they had struggled with communicating their vision to those around them [...] Architects and their clients would benefit tremendously from a tool that allowed them to visit the design and walk around it before it was even constructed".

Understanding the potential of this new technology, people at UNC modelled a VR environment of the new premises of Brook's lab. The experiment, which was probably the first example of the application of VR to architecture, led architects to change some of their original design choices (Pimentel et al., 1995).

During these years NASA, (Burdea et al., 1994), showed its interest in using this technology for space flight simulation. These projects played a decisive factor in the development of the first low cost VR system. In 1981, Michael McGreevy and Dr. Stephen Ellis respectively a cognitive science PhD and researcher at the Aerospace Human Factor Research Division of NASA, were engaged in research into the interpretation of 3D displays.

Aware of previous work carried on first by Sutherland and later by Furness they wanted to create a cheaper device that could replace the one-million dollar VCASS system. They successfully assembled the first Liquid Crystal Display (LCD) – based HMD, called *Virtual Visual Environment Display* (VIVED), using off-the shelf commercial LCD portable Sony TV sets, mounted onto a scuba diving mask fitted with special optics (Fisher et al., 1987). As Burdea et al. report (1994, p. 10) the NASA researchers were later joined by Amy Wu, a programmer, who coupled the VIVED system to a "DEC PDP 11/40 host computer, a Picture System 2 graphics computer (from Evans and Sutherland) with a Polhemus non-contact tracker". To convert the video signals from the computer to the LCD screens the research team used two cameras pointing at two 19 inch CRT screens, which were connected to the Evans and Sutherland system.

The set-up, although quite technically crude, was an incredible achievement. Pimentel et al. (1995, p. 62) noted, "It was the only $ 2,000 head-mounted display on the planet" connected to a computer that could render real-time graphics. The most important issue at that point was the quality of the system. Regarding this Pimentel et

al. (1995, p. 66) noted: "as Myron Krueger has so aptly pointed out, the image quality was so poor that you would be declared legally blind in most states if you had similar vision". But the VIVED system was to become a milestone in VR history. In fact unlike its predecessors, due to its limited cost it captured the attention of the public, triggered the interest of several non-military research institutions and established the basis for the research of the following years.

Meanwhile Thomas Zimmerman had developed a glove that could measure the bend of each finger. Zimmerman had originally developed the glove for musical applications, to allow a user to play music through a computer synthesiser. Meanwhile Scott Fisher, who had joined the VIVED group, suddenly realised the potential of Zimmerman's glove as an input device. He therefore interfaced the *glove* with the VIVED system using a computer representation of the hand within the virtual world (See Figure 2.3). As Pimentel et al. (1995, p. 65) noted, "for the first time, a representation of a person's physical body could became part of the simulation".

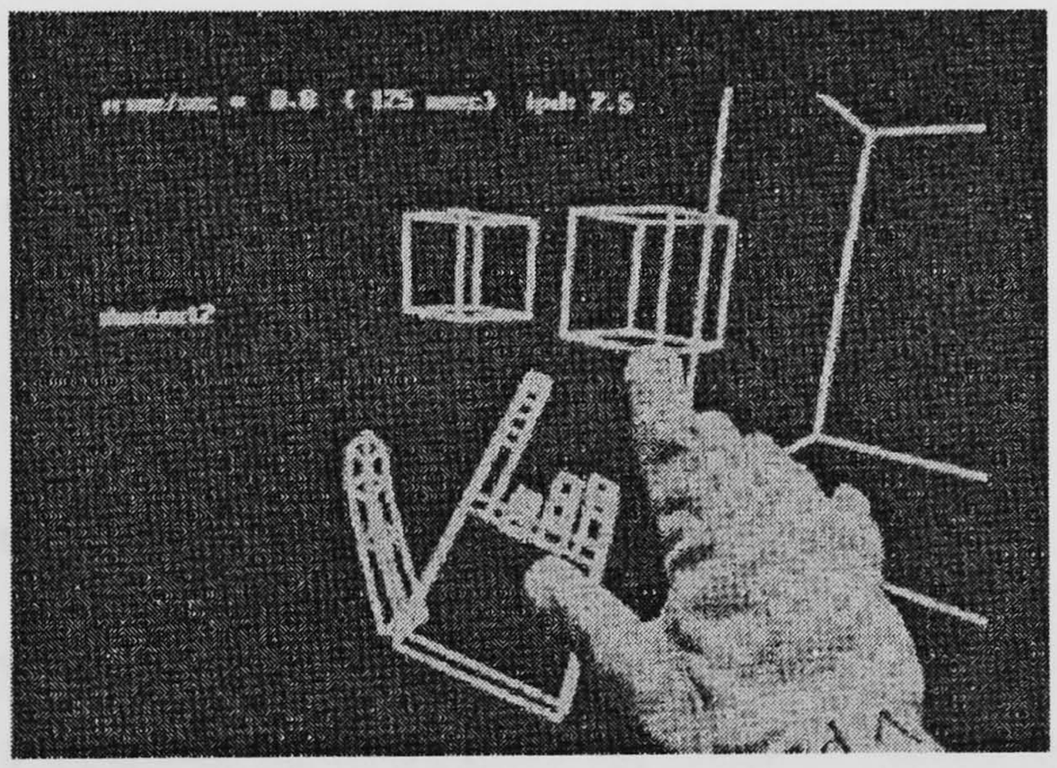


Figure 2.3: The DataGlove used by Zimmerman to interface the VIVED system (Fisher et al., 1987, p. 82)

Two years later Jaron Lanier, the father of the term "Virtual Reality" (Pimentel et al., 1995), joined Zimmerman and they founded a company called VPL Research (Burdea et al., 1994). In 1986 VPL developed the first VR system that could interface both with the user's movements through a glove and with their voice using a commercial voice-recognition system. The system could also reply by means

of a commercial voice synthesiser package. "No longer was the computer this separate thing you sat in front of and stared at; you were now completely inside. You communicated with it by talking and gesturing instead of typing and swearing." (Pimentel et al., 1995, p. 65).

The evolution of the hardware led, in 1988, to the Virtual Interface Environment Workstation (VIEW) system, the successor of VIVED. An HP 9000 system could render, for the first time, shaded images instead of simple wireframe ones (Burdea et al., 1994). At the same time Wenzer and Foster developed *Convolvotron*, a system that was able to calculate *binaural* sound from up to four sources at the same time. The characteristic of *binaural* sound is that it can be considered truly 3D sound. In fact, as Pimentel et al. (1995, p. 69) noted, "listening to a 3D recording allows the musical sources to appear anywhere in a sphere surrounding your head" instead of the effect obtained with traditional stereo systems where the sound seems to be coming from a flat plane in front of the user.

In 1988, three years after its foundation, VPL Inc. sold the first commercial VR products. The company started a successful business selling DataGloves and EyePhones and it released the first system to create a 3D world called Swivel 3-D. A few years later the company went out of business due to debts in the form of loans. By this time they were contracted to the French company Thomson CSF who eventually bought most of their patents (Pimentel et al., 1995).

2.3.4.4 The Development of PC-Based VR systems

The last major player in the short history of VR was Autodesk. John Walker, one of its founders, aware of the potential of the research being carried out at NASA, started to create the *Cyberia* project. As Pimentel et al. (1995) reported, the research team who were inspired by William Gibson's novel Neuromancer (1984), aimed to create a Gibsonian "deck" that could be used with normal PCs.

Generation	Description	Barrier
First	Plugboards, dedicated set-up	Front panel
Second	Punched card batch, RJE	Countertop
Third	Teletype timesharing	Terminal

Fourth	Menu system	Menu hierarchy
Fifth	Graphical controls, windows	Screen
Sixth	Cyberspace	?

Table 2.1: User Interface generation classification according to John Walker as quoted by Pimentel et al. (1995, p. 78)

Walker, in a visionary description of computer interface evolution, identified six phases of that evolution (See Table 2.1). Each phase would be characterised by a different metaphor and would make computers easier to understand. According to Walker (Pimentel et al., 1995) we are now at the start of the sixth generation and the *Cyberia* project intended to exploit its potential.

In 1989 the first PC-based prototype of a VR system was ready for demonstration, but a few months later Autodesk abandoned the project and two of the researchers involved in the project left to found another company, Sense8. Although the role of Autodesk might seem limited it had been of crucial importance to show that it was possible to develop a VR system that could run on a common PC.

Four years later Sense8 would release the first Cyberspace Development Toolkit (CDK) and in 1990 the company, together with Intel, developed the first texture-based PC system where photography could be imported into the 3D world to make the environment look more realistic. One year later they released the WorldKitTool the first C-based Application Programming Interface (API) that allowed the creation of customised 3D based applications (Pimentel et al., 1995).

Since then IMB, Sun, HP and especially Sgi have started to sell dedicated graphic engines for VR and a number of vendors are now selling VR interfaces or displays. The last few years in particular, have seen VR entering the home-entertainment market with products being sold as cheaply as a few hundred dollars each. The major player in the present development of VR applications is, as Knight et al. (1999) pointed out, the video games industry. "Manufacturers spend large amounts on developing both hardware and software to satisfy this burgeoning market for first person view 3D games that run (in current terms) on a standard PC" (Knight et al., 1999, p. 432).

2.4 Classification by Level of Immersion

Several authors have proposed taxonomies for the number of systems available nowadays. Pimentel et al. (1995) for example, suggest that VR systems fall into six categories:

- Cab simulators. Such as those used as a flight simulator, where the user is controlling the computer through an interface that is a replica of the actual device that the system is simulating. The windows of the cab are replaced by screens and the cab can be placed over a moving platform to simulate the effects of motion. The level of realism achievable is so high that pilots are now fully trained on this type of flight simulator.

- Projected reality. The first system in history falling to this category is Krueger's Metaplay (See Section 2.3.4.2) where two people in two separate rooms see each other's silhouette projected onto wall-sized screens and interact with them.

- Augmented Reality. This system uses a special transparent Head-Mounted Display where computer images are superimposed onto the image of the real world to *augment* it. These displays are also called Heads-Up-Displays (HUDs) "because the outside world is visible along with the computer-generated graphics" (Pimentel et al., 1995, p. 14).

- Telepresence. Pimentel et al. (1995, p. 14) suggest the following definition: a system that "uses video cameras and remote microphones to immerse users so deeply as to project them into a different place". One of the most important applications is for medical purposes where the doctor can operate on the patient's body from a remote location through a mechanical arm. Loeffler et al. (1994, p. xxi) consider *telepresence* as a "networked extension" of VR. The synthetic world is created in relation to the real one, and at the same time it affects the real world through an interface at the end of the network line. The entire system acts as a real-virtual-real converter.

- Desktop VR. This system uses a normal screen as the visualisation device. It is considered the simplest VR setting but nevertheless it can be very effective

in particular circumstances as will be discussed in the following pages (See Section 2.4.1.1).

- Visually Coupled Display. This is what Pimentel et al. (1995, p. 14) refer to as the "immersive system most often associated with virtual reality". The user wears a tracked HMD whose position affects the images projected on the screens (See Section 2.4.2.1).

However, in most of the literature the classification of VR systems is done on the basis of the *level of immersion* provided by the visualisation sub-system. The definition of *level of immersion* is rather vague: Pausch et al. (1997, p. 266) state that *immersion* is the sense of "being there" while Loeffler et al. (1994, p. xiv) consider it as a *sense of presence*, in other words, "the degree to which the user's senses are limited to the simulation and screened from the real world".

The *sense of presence* is used as a parameter to assess the effectiveness of the device in terms of simulation of the experience within an environment. Unfortunately though, there is no measurement directly describing the sense of presence. This could be calculated as a result of several factors such as the level of interactivity, the resolution of the system, the accuracy of the modelled world and the *latency* of the system.

Some authors like Pausch et al. (1997) and Arthur et al. (1993) have proposed some forms of experiment to measure the parameters associated with the level of immersion. Unfortunately though the results are often vague or are the consequence of highly constrained conditions or very specific hardware settings. Reeves et al. (2000, p. 70) state, "Unfortunately, all these studies do not allow a conclusion that increases in perceptual bandwidth will be universally good or bad. The most import conclusion is that more and different perceptual experiences turn up the volume on perceptual responses, an outcome that increases the importance of successes and failures in interface design".

The issue of the *immersiveness* of a VR system has led the research community towards two antithetical approaches, divided between those who back the advantages of the so-called *immersive* systems and those who support *non-immersive* ones. The first approach is typically characterised by the user wearing a helmet that

completely occludes their vision. The second approach is based on some kind of 2D projection display. The latter can support both monoscopic and stereoscopic images through the use of special glasses.

Between these two approaches there are several hybrid ones. Therefore within the scientific community it is generally acknowledged (Bowman et al., 2001) that most of the systems available fall into three main categories: *non-immersive*, *fully immersive* and *semi-immersive* projection systems.

2.4.1 Non-Immersive Systems

Non-immersive systems, as the name suggests, are based on traditional screens or projection surfaces. In these systems the feeling of being immersed merely relies on the user looking at the screen and therefore it immediately vanishes as soon as the user looks somewhere else.

There are several advantages to their use such as high resolution and low cost. One of the other main advantages of this technology is that it can also be used without special hardware and therefore it can be implemented on PC clones. Typical examples of desktop-VR systems, which are now part of our daily lives, are 3D computer games. Some authors like Knight et al. (Knight et al., 1999) have shown the effectiveness of PC-based game engines to navigate through virtual worlds.

2.4.1.1 Desktop-VR

The basic non-immersive or *desktop-VR* systems, as the name suggests, are based on traditional screens or projection surfaces. Typically, in a *desktop-VR* environment, the user can see the virtual world through a browser and interact with it by means of a traditional keyboard and mouse as well as more sophisticated 3D-interaction devices. Sherman et al. (1992, p. 24) comment on desktop-VR stating that "it is however proper Virtual Reality in that the images are generated in response to commands in real time (they are not just pulled from the memory)".

This simple visualisation device makes it possible to implement such systems with PC clones. One of the first examples of a non-immersive system based on low-cost PC clones was the Rend386 project started in 1991 at the University of

Waterloo. As Stampe et al. reported (1994, p. 56) the idea behind the project was to create a non-immersive low-cost VR system that could be used with the normal PCs available at that time, mostly 386 and 486-based PC clones, hence the name. The interface was developed starting from a Nintendo PowerGlove™ while the visualisation core was based on a standard VGA card. Support for stereoscopic views and for HMD and Sega LCD shutters was provided. Optimising the code for high speed but without any trade-off in functionality or usability operations, the system was capable of rendering 50,000 polygons per second. After several releases by 1993 the design team decided to launch a new package of VR-386 written on the basis of the experience gained from Rend386, but with an open architecture so that every programmer could improve the core. One year later Roehl (Stampe et al., 1994, p. 56) released AVRIL, a rendering library designed to be portable by the elimination of part of the assembly code present in Rend386.

The popularity of PC-based desktop-VR systems was partially encouraged by the use of *panoramas* which some authors (Gatermann et al., 2000; Gatermann et al., 2001) considered a basic form of VR, and partially by the advent of the Virtual Reality Modelling Language (VRML). VRML, which was introduced in 1995 by Silicon Graphics Inc. at the Third International Word Wide Web Conference in Darmstadt, had gradually become the standard for 3D virtual environments. In 1997 it was accredited with ISO Status (Web3D Consortium, 1997). Although VRML is now being replaced by other technologies such as Java3D™ (Sun Microsystems, Inc., 1999) or X3D (Web3D Consortium, 2002), it has still retained a relevant role in the success of many VR applications.

Several projects have tried to interface desktop-VR systems with different input devices. One of the most interesting cases was Moloney's Bike-R (1999), an example of a low cost VR environment where a low-tech exercise bicycle was connected to the computer. The computer, instead of rendering real-time animation, rendered pre-recorded sequences of videos. The user could interact with the system by pedalling and choosing different paths at the crossroads. To a certain extent, the system proposed the same approach demonstrated a few years earlier by Negroponte (1996) at Media Lab with the Aspen project (See Section 2.3.4.3), but Bike-R proved that a PC connected to a simple interface could be used successfully. Knight et al.,

(1999) proposed a similar approach where a bicycle was connected to a desktop-VR system. This, as the authors noted, was also inspired by Jeffrey Shaw's work at Autodesk a few years before (See Section 2.3.4.4).

The desktop-VR paradigm, taken to an extreme stage, saw the development by the present day of the first commercially successful attempts to port desktop-VR systems into so-called *embedded* devices, such as mobile phones, Personal Digital Assistants (PDAs) and PocketPCs. Commercial examples of this idea are available on the market, for example ParallelGraphics' Cortona (ParallelGraphics, 2002) or Swerve by Superscape, a technology that allows "3D games, innovative user interfaces and 3D Interactive messaging" (Superscape plc, 2001).

2.4.1.2 FishTank-VR

An improved version of basic desktop-VR is the so-called *FishTank-VR*. The technology, which has been adopted by several authors (Deering, 1995; Arthur et al., 1993), is based on a rendering engine attached to a monitor where stereoscopic images are rendered. The user wears a pair of tracked LCD shutter-glasses that transmit the position of the user to the computer, which then creates the relevant images according to the user's point of view.

As Pimentel et al. (1995) note, although a basic desktop-VR system can be implemented without a fully stereoscopic visual device the author recommends "a stereoscopic system because it provides important depth cues when manipulating objects in the virtual world. Though there's much debate over the value of stereoscopic visualisation, in certain situations it's an important feature" (Pimentel et al., 1995, p. 131).

In his work titled "High-resolution virtual reality", Deering (1992) has proved the precision of his FishTank-VR system. According to him it could achieve sub-millimetre accuracy. Some authors (Wesche et al., 2000) also proposed the combined use of haptic feedback devices like the PHANTOM™ from SensAble Technologies (SensAble Technologies, 2002).

All these elements combined with the rising increased amount of visual information provided by the motion parallax make this system extremely popular in industrial design applications.

2.4.1.3 Virtual Workbench

One evolution of simple FishTank-VR systems is the *Virtual Workbench* or *Virtual Table* (Schmalstieg et al., 1999). In a typical system a projection table replaces the screen. Commercial products like ImmersaDesk®R2 (Fakespace Systems, 2002) and Baron (Barco, 2002) have been successfully marketed. Here the metaphor is a table or a wall showing stereoscopic images and the user interacts through 3D-input devices.

As Wesche et al. (2001, p. 170) noted "When dealing with curves and surfaces in a workbench environment, the working situation is completely different from that on a desktop system. Curves and surfaces appear perspectively correct, shown as stereo objects in reach of the user's hand and represented on a scale corresponding to the work area". Several authors have achieved very interesting results using this technology (De Amicis et al., 2002; Fiorentino et al., 2002; Igarashi et al., 2000; Wesche et al., 2001; Schmalstieg et al., 1999).

An interesting precursor of the Virtual Workbench was Park's Electronic Drawing Board (EDB) (Park, 1996). EDB and the later EDB2 were implemented using a desk, a video-projector, a video camera and a computer. In this system mixed analogue-digital tool images filmed by a camera were projected onto a real desktop that became a real place for work: the user would use the desk as a real desk and as a computer desktop at the same time.

2.4.2 Fully Immersive Systems

The second category of VR system is the *fully immersive* type, which is, thanks to the media industry, the most famous type of VR system. This class includes two different categories: the Head-Mounted Display (HMD) and the Cave Automatic

Virtual Environment (CAVE). Both systems are characterised by providing the user with a 360-degree view of the environment.

It is obvious, from what has already been said about *non-immersive* systems, that the main advantage of this system is the great lever of presence. As Green et al. (1996, p. 48) note, "the issue of immersion versus non-immersion is hotly debated within the VR community. The supporters of the immersive side claim that HMDs will improve over time and will eventually have the same resolution as regular workstation screens. [...] The supporters of the non-immersive side claim that HMDs prevent easy access to standard design tools, such as paper documents, telephones, coffee and other people in the room". As Singh et al. (1996, p. 35) note, "there are significant usability and functional trade-offs between the two approaches. While the HMD-based systems are appealing for walkthrough and entertainment applications, they do not lend themselves well to precision work and extended use". Green et al. (1996, p. 48) also note that "immersive systems suffer from the navigation problem, since the user must continually move through the environment, or scale the object, to reach different parts of it. For example, in building design, if the designer is the same scale as the building, it can take a considerable amount of time to get from one place in the building to another".

2.4.2.1 The Head-Mounted Display (HMD)

These systems, typically based on a tracked Head-Mounted Display (HMD), are the evolution of Sutherland's first *Sword of Damocles* (See Section 2.3.4.1) and the later NASA's VIVED system (See Section 2.3.4.3.).

The HMD is typically composed of two screens where stereoscopic images are projected. A helmet is fitted with special optics that move the focus point from 50 to 70 mm, which is the average distance between the user's eyes and the screens in order for them to be able to see the objects in the virtual world. Often the screens used are based on LCD technology although some systems have adopted Cathode Ray Tubes (CRT). The latter can produce higher resolution images but are usually more expensive. The HMD is usually coupled with headphones to provide 3D-sound and the pointer device is normally a glove-based tool or a *wand*.

Several authors report on their use giving interesting results (Butterworth, 1992; Bolas, 1994; Chan et al., 1999a). According to these authors the user perceives a great sense of being immersed into the virtual world, as their head's position is tracked and the images shown are calculated according to their point of view, giving a 360-degree view of the environment. Butterworth et al. (1992, p. 135) wrote that "an HMD system gives the ability to understand the complex spatial relationship of models by placing the user in the model's world [...] as a result, the user can build the virtual world from within the virtual world".

However these systems tended to be very expensive, although the cost is rapidly dropping due to the necessary dedicated hardware coming down in price. Moreover, as several authors pointed, if compared with other types of VR systems, HMD's had the twin drawbacks of lower resolution and poor ergonomics due to the size and the weight of the helmet. However, regarding this, Negroponte (1993) wrote: "the argument will be made that head-mounted displays are not acceptable because people feel silly wearing them. The same was once said about stereo headphones".

2.4.2.2 Cave Automatic Virtual Environment (CAVE)

An interesting *immersive* alternative to traditional HMDs is the Cave Automatic Virtual Environment. The CAVE, as researchers referred to it, was developed a decade ago at the Electronic Visualisation Lab (EVL), University of Illinois, by Carolina Cruz-Neira, Dan Sandin, and Tom DeFanti and it was presented at the SIGGRAPH'92 conference.

As the authors declared (Cruz-Neira et al., 1993, p. 135) the name "is both a recursive acronym (CAVE Automatic Virtual Environment) and a reference to 'The Simile of the Cave' found in Plato's *Republic*, in which the philosopher discusses inferring reality (ideal forms) from projections (shadows) on the cave wall".

The system is based on a variable number of projection walls, from four to six, that limit the physical space where the user can stand. Computer-rendered stereoscopic images are projected on the walls giving the impression on the user that they are fully immersed into the virtual environment. The space, usually a 3x3x3

meter cube, is big enough to allow more than one user to experience the same virtual world.

2.4.3 Semi-Immersive Systems

As already mentioned between the two previously mentioned systems a third one has emerged: the *semi-immersive* type. This type borrows most of its technology from either the CAVE systems or from the simulation industry.

In these systems the rendering engine is typically made by multiple graphic subsystems that project synchronised images on different configurations of screens. For this reason they are also called by some authors (Wesche et al., 2000) Projection-Based Virtual Environments (PBVE). The visualisation device can be either a tessellated screen or the windows of what Pimentel et al. (1995) call a "cab-simulator" (See Section 2.4).

The tessellated screen type can be set up with different arrangements, depending on the requirements, from a semi-cylindrical, to hemispherical to spherical configuration (See Figure 2.4).

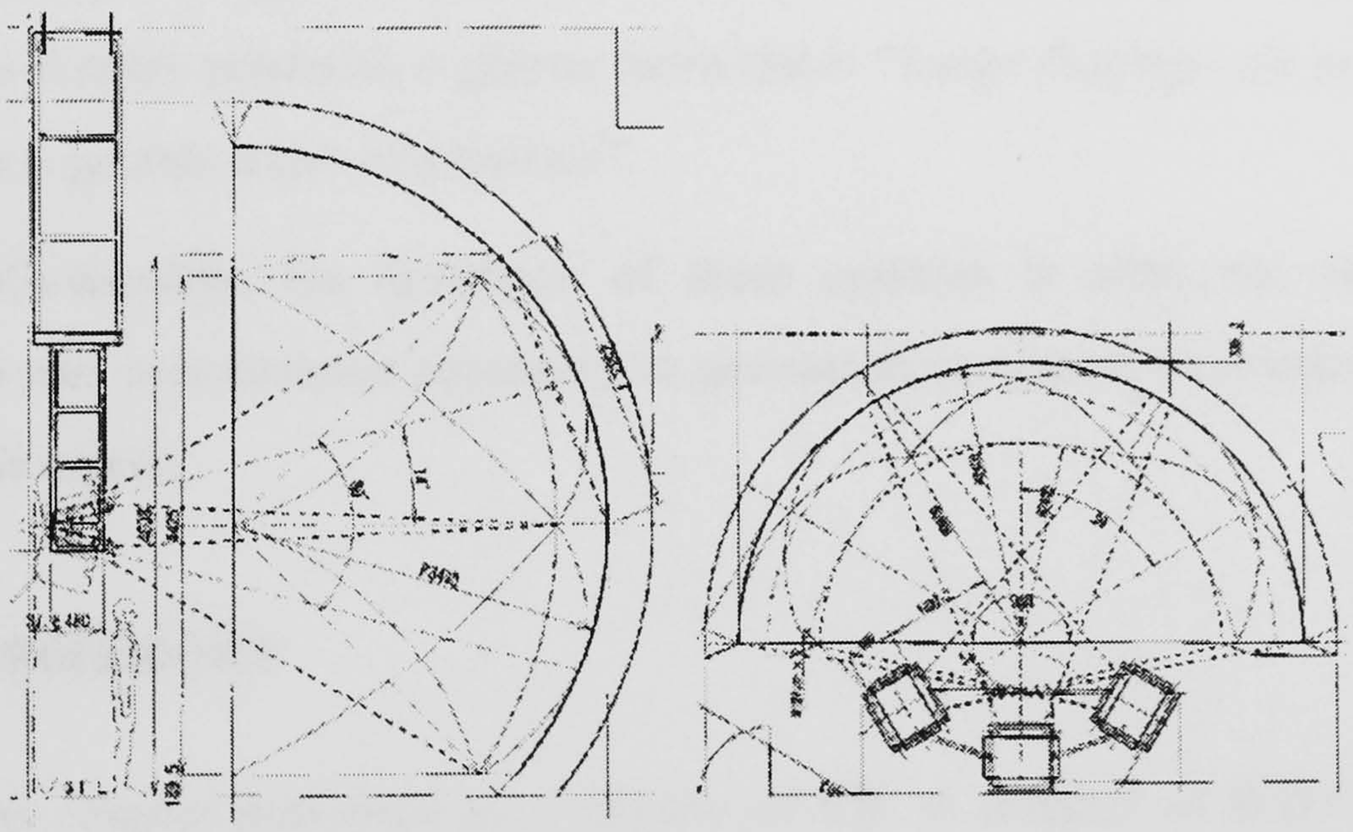


Figure 2.4: Two sections of semi-spherical VR room (From Fukuda et al., 2000, p. 491)

The cab type is typically a simulator where the inside of a cockpit of an airplane, a tank or the like, is replicated to create a training environment for pilots,

and whose windows have been replaced by screens. These applications were originally developed for military or training support but have also been recently borrowed by the video-game industry.

Several authors use these systems (Fukuda et al., 2000) and as most authors explain their advantages are to be found in the possibility of allowing a greater number of people to be involved in the simulation experience. For this reason they can be particularly appreciated where collaboration between several users is required or in a teaching context. Regarding this, Knight et al. (1999, p. 432) recommend that the ideal VR system used in an architecture studio environment should make use "of a large-scale projection system rather than other alternatives such as a head-mounted display or conventional monitor".

Stereoscopic effect can then be obtained with the use of additional shutter glasses although the implementation of simultaneous multiple tracked points of view is only feasible for a very limited number of users. The resolution achieved with these multiple projection systems can consequently be very high especially when compared to HMDs.

Additionally the sense of scale can be felt in a more dramatic way than with simple Desktop-VR systems. In fact, as Reeves et al. (2000, p. 69) show, the larger size of this display produces a greater *immersion*: "Large displays are preferred and they create a greater sense of presence".

Unfortunately, the drawback of these systems is often the very delicate calibration and maintenance necessary to prevent distortion or overlapping between areas of the screen.

2.5 Conclusions

This chapter presented an overview of VR. A number of definitions of the idiom *Virtual Reality* which recur in the scientific literature were given. The central part of the chapter presented the evolution of the hardware and software that over the last forty years has led to modern VR systems.

The last part of the chapter presented a classification of existing VR systems and highlighted their advantages and weaknesses. A classification of VR systems according to their level of "immersiveness" was given.

This chapter presented VR as the ultimate visualisation and simulation tool. The following chapter however will present VR as an innovative design tool. It will demonstrate how the superior visualisation capabilities and the interactivity of modern VR systems can be exploited to make the design process much more efficient.

The following chapter will particularly focus on the use of VR technologies in the early stages of the design process and examples of research, past and present will be reported.

3 VR as a design tool

3.1 Introduction

The previous chapter has shown the use of Virtual Reality as a visualisation tool. However, if limited to being a mere visualisation tool VR loses its power and becomes nothing more than an effective presentation technology. Such virtual environments are static worlds which the user can navigate freely, but as Green et al. (1996, p. 46) points out, "an environment where the user can merely move amongst the objects, like a ghost, and not interact with them in any form is not very interesting". To address this issue, this chapter will introduce the concept of using VR as a *design tool* as opposed to a simple visualisation vehicle and it will show the potential of this technology for the use in the early stages of the design process.

In the following section an overview of the issues related to the use of Computer Aided Conceptual Design (CACD) tools (Sener et al., 2002) will be presented. In particular the focus will be on the tools which are usually called *sketching* tools.

The relevant scientific literature tends to refer to *sketching* or *3D sketching* tools (Achten et al., 2000; De Vries et al., 2000; Deering, 1995; Deering, 1996; Do, 2001; Igarashi et al., 1999) although often only a few of these tools are actually based on traditional sketching. Most of them are closer to modelling packages but in these cases their modelling capability is usually just limited to a few basic tools and the focus is on providing a quick and easy interface to create shapes and to support the creative requirements of the initial design stage. This broader definition of *sketching* will be used here in the analysis of the problems relating to the early stages of the design process.

A number of previously developed or ongoing projects into the use of VR-based tools in the early stages of the design process will be presented at the end of the chapter. The focus of this thesis will be on the sub-set of VR systems made by Virtual Reality Aided Design tools (VRAD) (Turner et al., 1999) and a metaphor-based taxonomy of their use across different areas of design will be presented.

In particular the focus of this chapter will be on real-time 3D-modelling tools that require a "creative" role of the user. Therefore the proposed taxonomy will not include other studies that have proposed automated systems to create shapes, for instance Liu et al.'s (2000), where the computer generated 3D-shapes from 2D-images in real-time.

3.2 A Design Tool rather than a Mere Representation Medium

As noted in previous research (Ucelli et al., 2000) VR is quite often left to the last stages of the design process, as a powerful tool to show the clients to impress them with the design. Also Achten et al. (2000) point out that VR has been used mainly in the last stages of the design project as a potent visualisation tool but this is not exploiting a great deal of power early in the design stage. Similarly Alvarado et al. (2001, p. 423) observe that "computers have reached mainly an utilitarian role in architectural work, focused on quantitative production of plans and images, and not related to the quality of design [...] They are used more at the end of the process, than at the beginning, when the design is conceived".

Kurmann (1995 and 1998) is of the same point of view and notes that "architects face a significant lack of computer tools that truly support them in the early, conceptual stages of design" (Kurmann, 1998, p. 317). He also outlines that the majority of CAD packages are not designed specifically for architects but they are adapted versions of engineering programs. Moreover CAD systems are designed to fulfil the widest range of engineering needs and they mainly focus on the construction part of the design process. The designing phase will require different tools since the architect has to face more intuition based problems and less mathematical ones. Kurmann (1998) states that there are three questions regarding designing with computers: *how, what* and *when* to design, and also that "different interfaces are needed in the construction and the design phase of a project" (Kurmann, 1998, p. 318). He continues saying that "an architectural design tool has to both offer the ability to build space and to experience it" (Kurmann, 1998, p. 318).

An attempt to use VR earlier in the design stage has made in the research carried out by Shih et al. (1999). Here a VR system was used to study the behaviour of people during the emergency of being in a building on fire. Through the study of the routes followed by people escaping a fire the research, aimed to find the occupants' choice of exit. The results of the simulation revealed to the research team new safety issues not previously taken into account and also suggested some new design solutions.

But although this approach affected the design choices, it used VR as an enhanced investigation tool, and did not allow the user to manipulate the result of the design directly within the virtual environment. In applications like this it appears that the present use of VR "seems limited to presenting relatively static models of more or less finished designs without much support to change elements of the design" (Achten et al., 2000, p. 459).

Apart from simple browser-functioning applications, interactive VR technology has existed for some time in the form of programming libraries or toolkits. Traditionally it has required technical and programming skills that made it the sole domain of computer scientists. As Singh et al. (1996, p. 36) wrote in 1996, "given today's technology, it is almost impossible for a non-programmer to develop a good virtual environment". Also Green et al. (1996) pointed out that VR as a design tool should not only be available to computer scientists but should be accessible to designers, engineers and teachers.

In addition, according to Wang (1999, p. 71), "what the designer sees and understands is limited by the design world constructed through the design medium he has chosen to adopt". Similarly Ataman (2000) studied the effects of different media upon architectural design. To assess how the media used can affect the design process he developed an empirical study where two groups of students were asked to use digital media and manual media respectively. The study revealed that "unclear conceptualisations were more likely to appear in the manual media group" (Ataman, 2000, p. 94) and that the group using manual media "seemed to produce some easy-to-build concepts and created fewer categories". The experiment proved that "students who used digital media developed designs that suggested more

understanding of architectonic space and clear distinctions between the conceptual and perceptual spaces" (Ataman, 2000, p. 94). The author thought that the limited manipulation capability of manual media was the main reason for the lower quality and he pointed out (Ataman, 2000, p. 94) that "exploration and efficient conceptual representation of content is essential for effective concept development".

Similarly Chan et al. (1999a, p.43) stated that "if a medium provides designers with an opportunity to see a design interactively, a sense of projection into the design will be generated". This kind of projection could lead to a different outcome in the design and will be likely to affect the design process and ultimately to improve the design quality. This process should also allow enhanced creativity and multidirectional thinking.

Stating that the medium chosen by the designer can be of primary importance when generating ideas Ataman (2000, p. 93) concludes that "it would be appropriate to assert that the nature and power of the available media facilitates what is conceived and accomplished. Conversely, limitations in the design can result from the limitations of the media".

According to these results the use of Virtual Reality as a creation tool allows designers to have much greater control over design choices earlier in the process giving a more comprehensive view of the design.

Research in recent years has explored the implementation of VR-based tools to support the early stages of the design process. The goal of these studies is to develop new tools that allow the designer to manipulate the environment interactively, and to simultaneously create shapes in a simple and intuitive way.

These systems are now called Virtual Reality Aided Design tools (VRAD) (Turner et al., 1999) and they have originated from different areas of design. They are the results of research carried out in the fields of industrial design, car manufacturing (Fiorentino et al., 2002; De Amicis et al., 2002), architecture (Camarata et al., 2002; Do et al., 2000; Jung et al, 2002) and also in the figurative arts (Keefe et al., 2001).

3.3 Sketching with Computers: a Tool for the Early Stages in Design

Since the development of Ivan Sutherland's milestone application in computer graphics called Sketchpad (Sutherland, 1963), sketching with a computer has been considered as a "highly effective means of constructing geometric shapes" (Arvo et al., 2000, p. 73).

As Stellingwerff (1999, p. 492) noted "Sketching is interaction in optima forma. The medium is filled with represented knowledge and key information for a specific problem. While the sketch is produced, the information on the medium is re-read and interpreted, the designer can take creative steps and re-draw, reflect and refine the 'sketchy' ideas. Through a series of iterative loops the sketching designer might be able to create an innovative design from scratch". Sener et al. (2002, p. 539), quoting Temple (1994), stated that "the activity of 'sketching' in its widest sense is an essential tool employed [...] in any creative enterprise as part of the conceptualising process involved in 'working up' ideas to a final and formal condition".

Several authors agreed on the unsuitability of current CAD packages when it came to using them in the early stages of design. Turner et al. (1999, p. 155) stated that "current CAD technology is far too restrictive to use as a tool for thinking with". Knight et al. (1999, p. 432) noted that "there is a need for better computer based conceptual sketchers and modelers". Finally, Roberts (1999, p. 445) points out that "CAD packages are designed to allow one to draw accurately, and therefore for every operation [...] the computer will ask for a certain number of parameters. [...] In a Virtual Reality environment, one could simply pick up an object and rotate it around as would happen with a real life model".

Regarding the use of VR in the early design stages, Chan et al. (1999a) noted that most VR systems rely on external modelling packages used before the model is exported to the VR system to create or manipulate geometries. This approach certainly "limits the flexibility and plasticity of VR's application [...] and allows VR technology to be used only as a production tool instead of a design tool" (Chan et al.,

1999a, p. 44). The inadequacy of using a simple VR-based visualisation system is evident when a change in the model is required: the designer has to return to the modeller and eventually back to the VR system. "The lack of immediate interaction and feedback may cause visual perception errors while modifying the design" (Chan et al., 1999a, p. 45).

Nevertheless, as Sener et al. (2002) noted there are three main reasons promote the use of digital media in the early phase of the design process:

- Every method or tool that can enhance the power of the conceptualisation phase should be used.
- Since the design process will ultimately require a digital support for engineering or planning purposes the ideas presented in the sketch will have to be converted to digital format.
- The computer data can also be used from the design phase to quickly produce physical models through rapid manufacturing processes and thus further enhances the power of creativity at this stage (Ucelli et al., 2000).

Moreover, as Achten et al. (2000, p. 459) wrote, VR is "inherently spatial by nature, it involves the architect in immersive environment, and it enables interactive manipulation of the design". As Kurmann (1998, p. 319) stated "what has to be offered to a designer is the possibility of interacting with a computer model in a convincing way".

For these reasons several authors have focused their attention on new tools that allow the creation of shapes with ease and with few constraints and they have used the terms *sketching* or *3D Sketching* tools to describe them (Achten et al., 2000; De Vries et al., 2000; Deering, 1995; Deering, 1996; Do, 2001; Igarashi et al., 1999).

Most of these systems are not strictly based on proper sketching but they are often very interactive basic modellers providing the user with a basic set of functions and a very flexible interface. This broader definition of *sketching* tool will be adopted in the future and consequently we will refer to the act of *sketching* with computers as the act of quickly creating, manipulating and editing shapes during the

early stages of design. Deisinger et al. (2000) have identified some key features for the success of such tools:

- Absence of reference to the mathematical representation behind the system.
- Real-time interaction.
- Full scale modelling.
- Intuitive interface.

The general principle behind these systems is therefore to provide the user with the few essential commands which are required to create basic shapes rather than providing them with a comprehensive set of modelling tools that would clutter the interface. In fact as Szeliski et al. (1992, p. 185) noted, "for shape design and rapid prototyping application, we require a highly interactive system which does not force the designer to think about the underlying representation or to be limited by its choice". By using this new family of systems, as Chan et al. (1999a, p.43) observed, designing with VR becomes "a new and unconventional mode that will heavily influence design thinking".

3.4 A taxonomy of Virtual Reality Aided Design systems

The next sections propose an overview of the previous and ongoing research into VRAD systems. The following sections will present a metaphor-based taxonomy of the existing VRAD systems. For each system, the interface and the approach followed will be described. No mention will be made on the topic of the mathematical implications of each system. Nevertheless references to the authors' research will be provided to give further details on each system's algorithms.

3.4.1 Non-Photorealistic Systems

Non-photorealistic rendering is a technique developed to preserve or sometimes even simulate, the imprecision and natural feeling of a hand made drawing. As Durand (2002, p. 112) wrote "*non-photorealistic* is a loosely-defined term. It should be used only to qualify a pictorial style. The only meaning of non-

photorealistic is that the picture does not attempt to imitate photography and reach optical accuracy". According to Cohen et al. (Cohen et al., 2000, p. 84) the drawing maintains "the distinct stylistic appearance and subtleties imparted by the user". As Markosian et al. (1997, p. 415) noted "Nonphotorealistic rendering (NPR) can help make *comprehensible* but simple pictures of complicated objects by employing an economy of line. Graphic designers have long understood that photographs are not always the best choice for presenting visual information". Durand (2002, p. 111) at the Laboratory for Computer Science at MIT, pointed out that "Non-photorealistic pictures can be more effective at conveying information, more expressive and more beautiful". Durand (2002) also noted that the division between photorealism and non-photorealism is not sharp but it is rather fuzzy. Both approaches also have to tackle the same *pictorial* issues like lighting etc.

Several authors have studied the use of Non-PhotoRealistic (NPR) algorithms to create images with qualities resembling the human drawing style. Until a few years ago the NPR technique was only possible for batch processes. Recently, the increased efficiency of the hardware and the optimisation of the rendering algorithm have made it possible to perform real-time non-photorealistic rendering (Northrup et al., 2000; Markosian et al., 1997; Raskar et al., 1999).

In their work Cohen et al. (2000) catalogued the non-photorealistic systems into two main types: *geometric* and *image-based* systems. The *geometric* group "attempts to create a geometric description of the 3D scene from the user's 2D input" (Cohen et al., 2000, p. 83) and the user can render the object from any point of view. Systems like *Sketch VRML* (See Section 3.4.1.3) follow this approach. However, the limitation of this system is that the geometry is often incorrect.

By contrast the *image-based* approach does not create a geometrical representation of the scene "but instead redisplays the original input image, modified to reflect the new camera parameters" (Cohen et al., 2000, p. 83). The approach followed by Cohen et al. (2000) in *Harold* (See Section 3.4.1.1) is an example of this second group.

3.4.1.1 Harold

The metaphor followed by Cohen et al. (2000) to create *Harold* is *drawing, i.e.*: the user draws 2D shapes through a 2D input device and then these shapes are rendered over a *billboard* in the virtual space. A *billboard* is a view dependent object that is always rendered facing the user's point of view: "a billboard is typically a plane with a image texture-mapped onto it that rotates about some point or axis to face the viewer as much as possible" (Cohen et al., 2000, p. 83). In this way the virtual world, as the authors have said (Cohen et al., 2000, p. 83), "is populated by drawings, not 3D objects".

This approach makes the entire system very simple since there is no need for interpretation of the strokes that are simply being rendered every frame. Moreover the shapes created in this way "maintain a hand-drawn appearance" (Cohen et al., 2000, p. 83).

The main drawback of using billboards becomes visible when the user moves through the scene and finds that the relative orientation between objects changes accordingly. For instance, if the user creates a simple house out of four walls and a roof, the entire object becomes inconsistent once the user starts to move.

To solve this problem the authors developed what they call *bridge billboard*, a "collection of planar strokes anchored to points on two billboards" (Cohen et al., 2000, p. 84).

The use of *Harold* is therefore limited to objects with approximately cylindrical symmetry, for instance trees, posts or even people, because the billboard effect becomes disturbing in the other cases.

3.4.1.2 SketchBoX

As Stellingwerff (1999) reported the SketchBoX system has been developed to exploit two powerful features that computers can offer when the metaphor of the pen and paper is being followed. In this system the user can place semi-transparent planes onto the faces of 3D models and use them as canvases to draw on. The user can then make annotation on top of the model in a similar way to using Redliner (See Section 3.4.2.5), the system proposed by Jung et al. (2002).

3.4.1.3 Sketch VRML

Sketch VRML is a tool to allow designers to transform sketches into 3D-entities. As Jozen et al. (1999, p. 561) noted that "nothing is easier than sketch on paper" but it is then difficult to implement these sketches into the design. Simply scanned drawings would not be useful to create 3D objects because they could only be used to texture a surface.

The approach followed by Sketch VRML to take sketches into the 3D world is similar to that developed by other authors (Cohen et al., 2000; Stellingwerff, 1999). However, unlike other systems, *Sketch VRML* does not automatically transform the sketched drawing into a texture and import it into the virtual world. Instead the authors have proposed a more original solution.

The software, makes a model of the drawing from the scanned paper-based sketch through an algorithm that turns black pixels into points in space. To solve the inevitable ambiguity of the third dimension, an algorithm written in Java™, takes into account the strength of the strokes in the drawing and it calculates a plausible value for their depth. Finally, all the relevant information is sent to the VRML browser through the VRML External Authoring Interface (EAI) and then it is shown in the virtual world.

The result is a plausible interpretation of the drawing in 3D space that can eventually be imported into a CAD package for further changes.

3.4.2 The semantic approach

During the process of designing a person "makes a drawing, which stimulates recall of similar forms, visual analogs [...] and the designer reacts in turn by making another drawing" (Gross et al., 1996a, p. 183). The more the drawing is ambiguous the wider the stimulation will be. Unfortunately CAD packages do not support ambiguity and therefore "the suggestive power of the sketch" (Gross et al., 1996a, p. 183) cannot be interpreted as an additional level of information carried within the semantic of the sketch. The result of this limitation is that the transfer to CAD

systems normally only happens when the idea is at its final conceptualisation and requires to be resolved.

The work of the research presented in the following sections focused on the retrieval of the semantic information stored inside a drawing. This is accomplished by interpreting the sketches through a new computer based system which was developed to assist the user in the early stages of the design process.

The outcome of research was to be systems that aimed to create shapes through the recognition of the semantic behind the hand-made strokes rather than focusing on the creation of complex surfaces. The result of these efforts is a number of intelligent computer aided systems, which through a knowledge based system supports users in the early stages of designing by providing flexibility of free hand sketching with computer editing and calculating power.

In the following paragraphs several systems will be presented. The first two systems introduced, the Cocktail Napkin and Flatland, are non exclusively related with 3D graphics. Nevertheless they are included in this taxonomy because they provided the basis for the 3D-based systems which are described later. Therefore these first two systems descriptions are to be read as the fundamental introduction to the ones following.

3.4.2.1 The Electronic Cocktail Napkin

Several research projects have tried to show how traditional sketches can convey design ideas. Gross et al. (1996a, p. 183) conjected "that designers prefer to use paper and pencil because it supports ambiguity, imprecision, and incremental formalization of ideas as well as rapid exploration of ideas". As Gross et al. (1996a) summarised there are three key features in characterising sketches:

- Abstraction. A symbol replaces more complex configurations "enabling a designer to work with components without specifying their internal structure" (Gross et al., 1996a, p. 184).
- Ambiguity and Vagueness. "Another way to postpone commitment yet retain a marker for a later decision" (Gross et al., 1996a, p. 184). Ambiguity and

Vagueness can be caused by the variety of forms of the design solution or simply by the simplicity of the idea.

- Imprecision. "Designers need only rough dimensions to decide on a basic layout" (Gross et al., 1996a, p. 184), more detailed drawings and exact calculations will follow once the design idea is established.

Traditionally computer based applications "force designers into premature commitment, demand inappropriate precision, and are tedious to use compared with pencil and paper" (Gross et al., 1996a, p. 183).

Nevertheless Gross et al. (1996a) also observed that computer based media indeed can offer many advantages over traditional paper-based techniques for example 3D modelling, distance collaboration, fast editing and easy storage and retrieval. The ideal system should therefore have the best aspects of both methods and the system proposed tries to address these issues.

The idea for the *Electronic Cocktail Napkin* or simply *Napkin* was inspired by the commercialisation of the Apple Newton Message Pad. The system, is a freehand drawing tool written in Macintosh Common Lisp on an Apple PDA. The system's aim "is to support the kind of informal drawing that designers do on the back of an envelope or a cocktail napkin during conceptual design" (Gross et al., 1996a, p. 185).

The user draws through a digitiser onto an area called the Drawing Board or Sketchbook window. *Napkin* supports three levels of recognition: "(1) the low level recognition of hand drawn glyphs (2) the higher level recognition of configurations, and (3) the maintenance of constraints and spatial relations among diagram elements" (Gross et al., 1996a, pp. 188).

On the first level the system tries to recognise and store the *glyphs* drawn by the users as shapes (boxes, circles, lines). If ambiguities are found the system treats the entity as unknown until further information is provided. Once recognised the shapes can be recalled for editing.

On the second level, the difficulty to interpret *diagrams*, the symbols sketched using strokes (Gross et al., 1996a), is not only related to the recognition of

the shape but also to the interpretation of the semantic behind it. For the semantic to be understood the program has to know about the context of the drawing, and so the system interprets the configuration according to this.

As the Gross et al. (1996a) report, *Napkin* has some re-configuration support embedded in it. This means, for instance that if there are four chairs sketched around a table this might be successfully recognised by the system as a dining table and be replaced by the appropriate symbol. Later the user can make a query and retrieve the old configuration if required.

The description of constraints among diagram elements requires the creation of "rules" or what the authors call a "configuration recogniser" (Gross et al., 1996a). The designer draws a sample where all the graphical relationships are present and then states the relevant conditions to create a rule, including the context of the drawing. This way the system can, for instance, interpret four boxes around another bigger one as four chairs around a table rather then four houses around a lake (Gross et al., 1996a).

3.4.2.2 Flatland

Similarly to *Napkin*, the research developed by Igarashi et al. (2000) and Mynatt et al. (1999) tried to create a system whose interface could answer the needs of the user. The outcome is a system called *Flatland* "an augmented whiteboard interface designed for informal office work" (Mynatt et al., 1999, p. 346).

Although the scope of this research was not only constrained to the design task, the project faced the problem of bringing an integrated interface to the work place and it tried to address it by proposing a computer-based assistant. The research team decided to focus on using a whiteboard tool for four reasons (Mynatt et al., 1999):

1. Because it deals with *pre-production tasks*. It acts as a working area and a "repository to support *thinking* tasks such as sketching out a paper as well as quick *capture* tasks such as jotting down a reminder" (Mynatt et al., 1999, p. 346).

2. For its *everyday content.* Information on whiteboards is upgraded daily. Its typical content is reminders, to-do lists, sketches and notes and it has quite an informal, incomplete and often *transient* format.

3. Its content is often organised into *clusters* where information is grouped together. Some critical data, like important telephone numbers, can remain on a whiteboard for a long time. Other pieces of data can be there for a very short time.

4. For its s*emi-public* and/or *personal* use. Due to the size and nature of the tool it can be used as a way of communicating things to others as well as being a personal tool.

The goal of the research team was to provide the user with an intelligent assistant with a very natural *look and feel*. For this reason *Flatland* follows the metaphor of the whiteboard through a computer connected to a LCD projector which projects images over a large SmartBoard™ (Smart Technologies, Inc, 2002) board hung on a wall. Although this configuration can take advantage of its large scale, as the authors note, it can also be implemented with PCs or PDAs since it is coded in Java™ and it is therefore hardware independent.

The user writes on the board with a stylus or a marker and the captured strokes are projected back onto the surface of the board. The input is based on the *strokes* drawn by the user and the output is always given in an analogous form through *hand-written style* content.

As a result, *Flatland* works like a normal whiteboard with some intelligence embedded in it, and widgets if any are limited to minimum. For example, when a user taps on the board a very informal pie menu appears (See Figure 3.1).

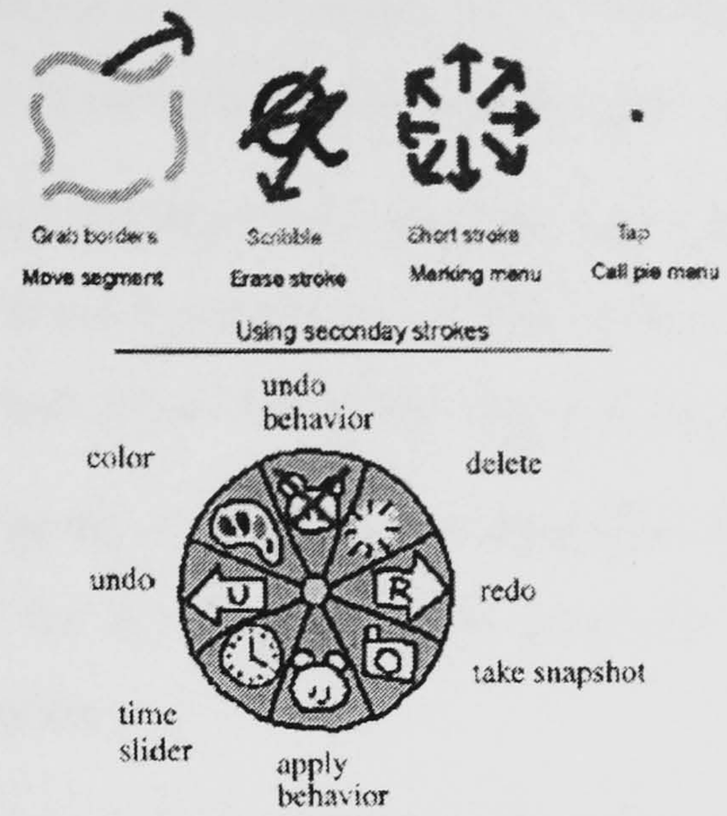


Figure 3.1: Gesture choices and menus in Flatland (Mynatt et al., 1999, p. 348)

The entire system is based on two important concepts: *dynamic segmenting* and *pluggable behaviors* (Igarashi et al., 2000). The former are the equivalent of windows in traditional GUI, they are regions that the system automatically identifies when the user draws a stroke. If the user starts writing on the board the system will group the strokes as needed, the user can then modify these areas by merging or splitting them.

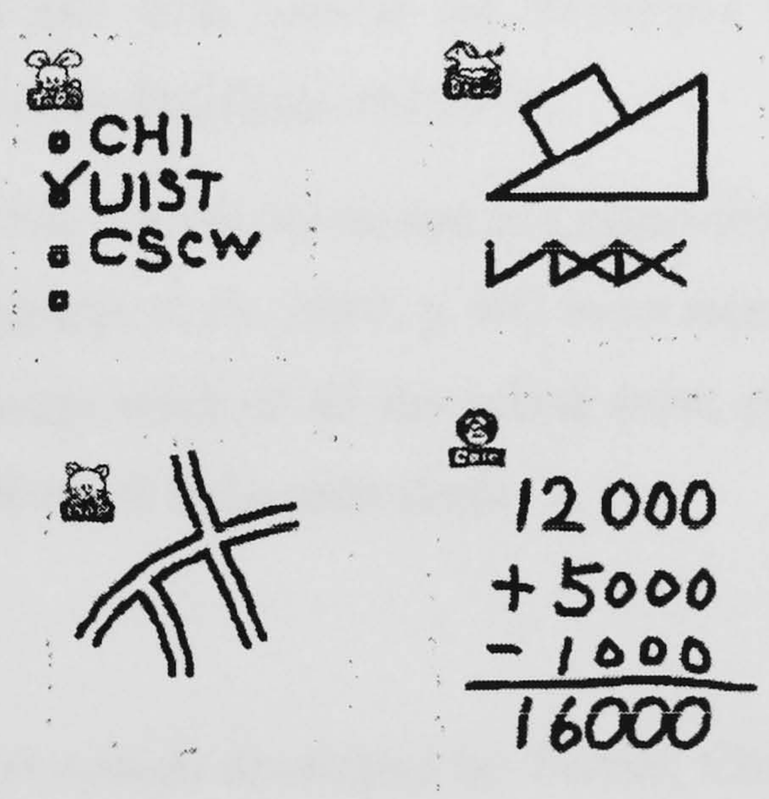


Figure 3.2: Behaviours in Flatland (Mynatt et al., 1999, p. 350)

Six different *behaviours* (See Figure 3.2) can then be dynamically attached to or removed from these regions:

1. To-do list: the system automatically prints hand-written sketchy checkboxes and it maintains the text aligned vertically.

2. Map drawing: the strokes are converted into a double line to represent a street and the intersections between streets are handled appropriately.

3. 2D Geometric Drawing: the drawing is interpreted or *beautified* and it is replaced by its rigorous geometrical representation. This system also predicts the next line based on the geometrical properties of the shape being drawn.

4. 3D Drawing: mainly based on the *Teddy* system (See Section 3.4.6.2) that allows 3D sketching based on 2D drawing. The user can also rotate and edit the geometry using the stylus.

5. Calculation: it recognises hand-written figures and it calculates the formula once a line is drawn below the formula.

6. A search engine: it allows retrieval of previous data, searching by size, time and then proposes a choice of thumbnails from which it is possible to retrieve previous configurations.

The difference between these behaviours and traditional applications is evident when considering the example proposed by the authors (Mynatt et al., 1999). Unlike traditional application, in this example the user can start with map behaviour and draw a few streets and then remove the behaviour and replace it with 2D drawing, where they can draw buildings and so on.

It was concluded that traditional saving and retrieving operations would cause "too much overhead" (Igarashi et al., 2000, p. 69) so an automatic saving system has been developed which keeps track of all the action done, providing the user with a timeline and an infinite level of undo-redo steps.

3.4.2.3 Stilton

Stilton is a VRAD system developed by Turner, Chapman and Penn. As the authors reported (Turner et al., 1999) the idea behind Stilton was to provide architects with a much more user-friendly tool that could be used during design.

Architects spend "much time sketching concepts on paper, and then construct models of their thinking in 3D using a multitude of materials" (Turner et al., 1999, p. 155). The authors believed that this should become part of the digital process and

proposed a system where architects could sketch 2D drawings, create 3D models and then maybe sketch some more on top of the model.

The research was inspired by the approach taken by Cohen et al. (1999 and 2000) where the user can draw freehand sketches that are automatically converted into textures and placed in the 3D world. This did not provide an interpretation of the underlying semantics however whilst the research team with *Stilton* proved the feasibility of such an approach.

The *Stilton* system is close to the one proposed by Jung et al. (2002) (See Section 3.4.2.5) but it allows further manipulation of the geometric features present in the 3D environment.

The system is based on a VRML browser modified to allow the user to draw on a virtual (and invisible) canvas placed close to the near clipping plane. For the drawing to be interpreted the user has to sketch over an existing environment that can be either a VRML model or a picture. This has to be done for two reasons:

1. To give the user some clues about the vanishing point used and therefore to make the recognition process much easier.
2. To *ground* the interpreted sketch within the model space. The interpreted model is placed over the surface on which the user has drawn the sketch.

The drawback of this system is however that the interpretation is not done in real time, but after the sketch has been completed and the user has asked for *interpretation*. According to the authors in fact, the routine would take approximately 10 seconds on an Sgi O2 R10000-based workstation.

After the interpretation has been done the system creates the relevant line of VRML code and the scene is upgraded.

3.4.2.4 Sketchpad

Ellen Do (2000 and 2001) describes the *Sketchpad* system as a tool to create three-dimensional models from 2D sketches.

The metaphor chosen is a sheet of paper where the user draws plans of the design proposal. Exploiting the author's previous experience in drawing

interpretation (Gross et al., 1996a; Gross et al., 1996b) the work proposed "an interface that allows direct interpretation of the drawing marks" (Do, 2000, p. 265).

The system was in fact based on the concept of the *Napkin* (See Section 3.4.2.1) where the user draws on a sketchpad area and the result is visualised into a VRML browser. Therefore the interface is very natural, since the only thing the user has to do is to sketch a floor plan with walls, columns and furniture to be interpreted by the system and eventually it is shown in the 3D world (See Figure 3.3). The user can also draw arrows to indicate the points of view, which are converted into corresponding *cameras* in the VRML files once they are interpreted.

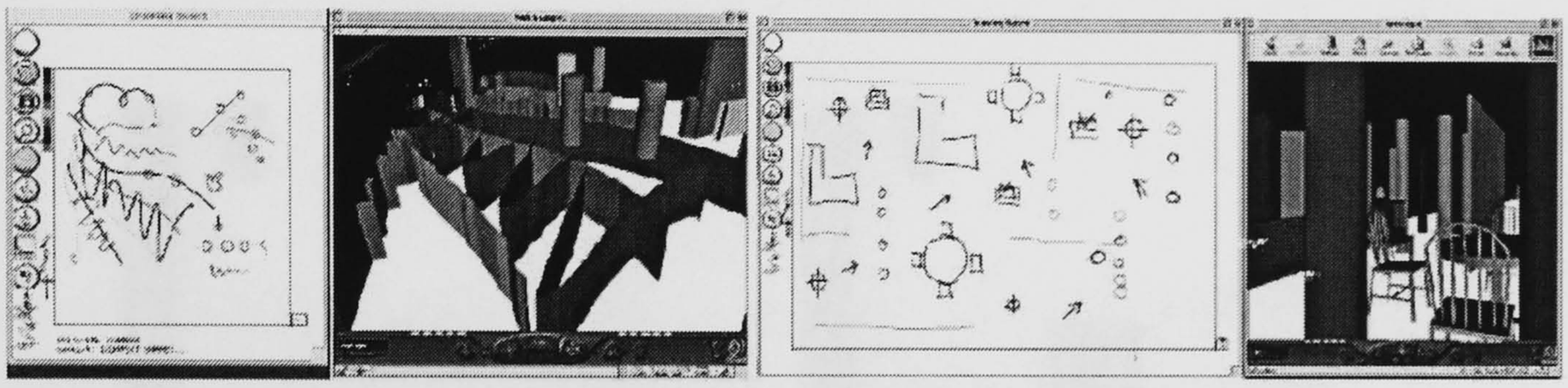


Figure 3.3: A screenshot of Sketchpad (Do, 2001)

The implementation is based on a Java™ applet that ensures the communication between *Napkin* and the browser. On one side, it monitors the browser's point of view to report to *Napkin* the position of the user. On the other, it listens to the messages sent to *Napkin* by touch sensors in the virtual world containing the information that the user has touched an object. A Perl script then changes the VRML file and therefore provides the necessary consistency across the system.

3.4.2.5 Redliner and Space Pen

The *Space Pen* system, developed by Jung et al. (2002), proposed a method to draw shapes onto existing 3D environments. The system, coded in Java 3D™, is an evolution of Redliner (Jung et al., 2002). This is a system used to annotate text in a VRML environment which benefits from the experience the research team had in developing *Napkin* (Gross et al., 1996a, 1996b) (See Section 3.4.2.1) and Sketchpad (See Section 3.4.2.4).

There was difficulty in navigation in Redliner, due to the Cosmo Player plug-in and the fact that "Text annotation was considered limited" (Jung et al., 2002, p. 100) and this encouraged the research team to develop another system called Space Pen.

The graphical user interface is reduced to minimum, there are colour selection buttons, a "save" icon and a text area that echoes the shape that has been recognised. In Space Pen the shape is actually drawn directly into the 3D environment and if it is recognised it is converted into the equivalent topological element according to the context (See Figure 3.4).

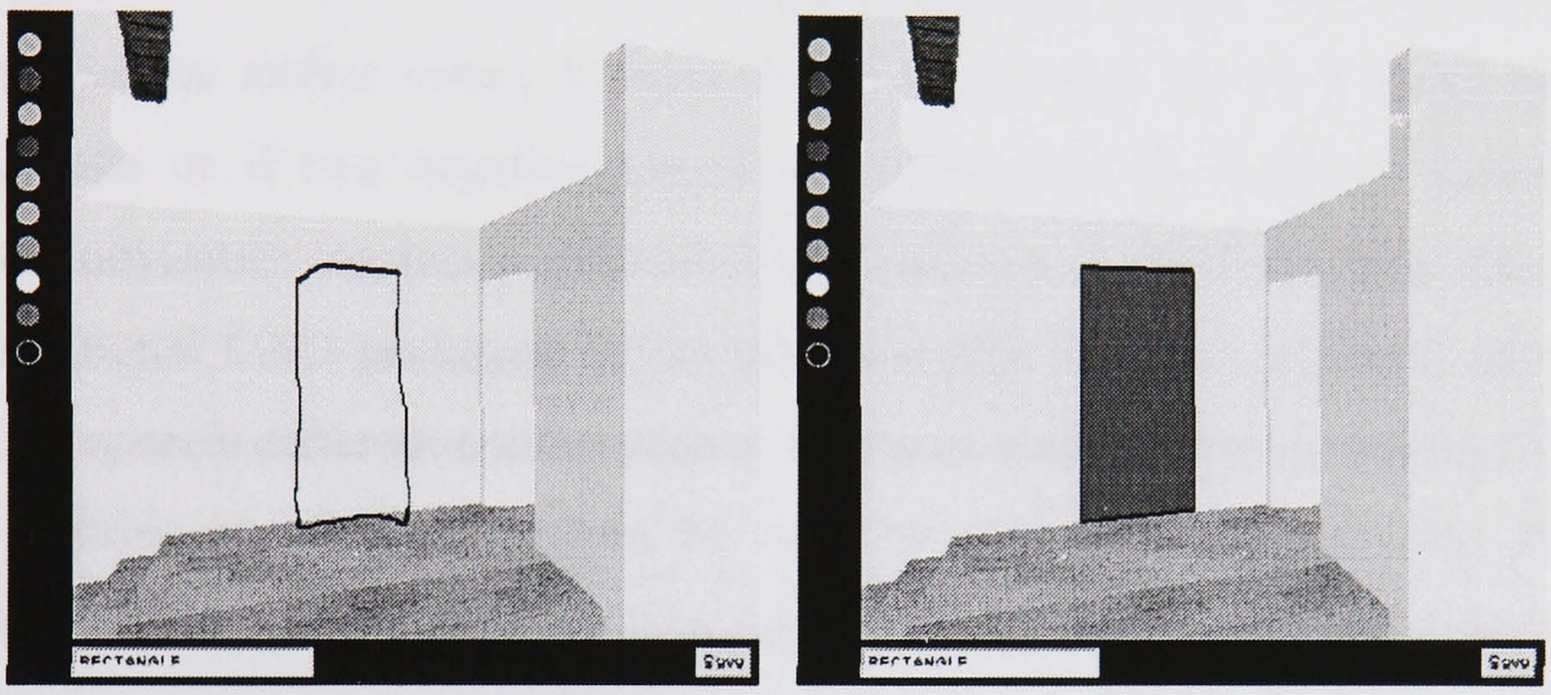


Figure 3.4: The Space Pen interface and the shape annotated after being recognised (Jung et al., 2002, p. 100)

Although the system does not support modelling, it provides a very powerful interface and it could be used to support the manipulation of existing geometries. Current research is trying to link the objects created to a database that would allow further data processing.

3.4.2.6 Sculptor

Sculptor, the software developed by Kurmann (1995 and 1998) at UTH in Zurich, proposed a different approach to the traditional sketchpad so far described. Different traditional media like pencil drawings and cardboard models share what Kurmann (1998) calls the "transformation of ideas". They provide the means to shape the idea which is then *transformed* every time it is *re-presented*. Therefore

"the more hurdles there are between the model and the designer the less intuitive the interaction process is" (Kurmann, 1998, p. 319) and since architects think graphically (Kurmann, 1998) a design tool has to support this attitude by promoting immediate graphical representation.

From the interface point of view *Sculptor* is much closer to a *modeller* in traditional terminology. But rather than developing a solid modelling tool the system is based on what the author calls *void modelling.* In fact Kurmann has taken inspiration from the concept of positive and negative objects to model space which was proposed by Yessios (1987).

Standard Boolean operations can be done between positive and negative objects but the result is different if compared with traditional 3D modelling. For instance, as the author noted, a negative volume always carves a space out of a positive one or if two negative spaces are intersected no result is shown. This approach obviously requires a different data structure and a different implementation than traditional CAD packages. It has the advantage however in that it allows the system to specify different entities such as rooms as spaces to be carved out of solids. These entities are then recognised by *Sculptor* and stored in the data structure embedded in the system. *Sculptor* then can use the dataset to provide support to the designer through a number of *agents*.

The *Navigator Agent* for instance, helps the user to go to a specific room or place. It can understand simple commands like "go", "jump to" etc. and it is aware of the configuration of the space from the data structure of the model. For instance it can be used to drive the user to a certain destination without colliding with objects and via the shortest route.

The *Sound Agent* adds an auditory dimension to the visual experience. This is implemented outside Sculptor and communicates with it through NCSA Data Transfer Mechanism (DTM) (Terstriep et al., 1993). The *Sound Agent* can play what Kurmann et al. (1997) called *sound labels*, which are pre-recorded sounds associated with actions such as the sound of footsteps while walking. By being aware of the configuration of the space, the *Sound Agent* can also play a sound compatible with the purpose of different rooms.

The last agent, the *Cost Agent*, estimates the cost of the project. The visualisation is provided through bars which turn from green to red if the cost is too high. Like the previous agent the *Cost Agent* is made by software connected to Sculptor via DTM.

To help the user experience the design, the system uses a collision detection functionality which gives them "a very intuitive way to have the experience of the scene" (Kurmann, 1999). If an action upon an object will result in a collision it will not be allowed. According to Kurmann another constraint function, which helps the user feel a sense of presence, is gravity. This is implemented as a force pulling the objects to the ground.

By turning these two constrains on and off the design state changes and the scene might evolve automatically towards a new configuration. For instance some objects might move downwards once gravity is activated.

The prototype has integrated an architectural floor planning software. This allows constraints such as minimum and maximum size of spaces to be set. The results are visualised into sculptor and the user can proceed to further refinements.

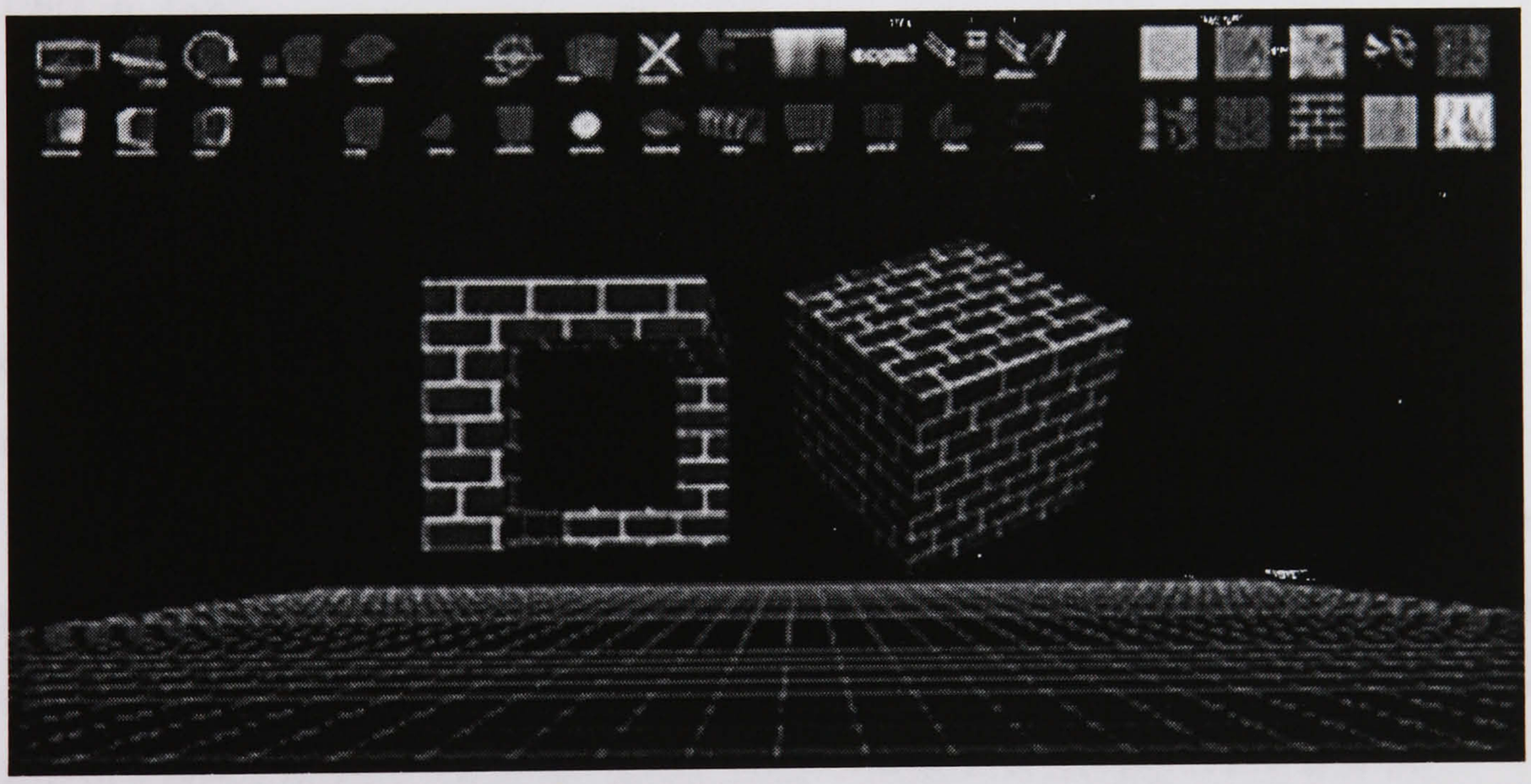


Figure 3.5: A view of Sculptor's interface (From Kurmann, 1999)

3.4.3 Primitive-Based Systems

These systems are characterised by providing the user with some basic shapes that can be edited. The main difference to the free-form approach (illustrated later in Section 3.4.6) is that the surfaces are either already defined or are created through extrusion, without either involving complex algorithms or handling of profiles or control points.

3.4.3.1 3DM

One of the first primitive-based systems was developed by Butterworth et al. (1992) at the University of North Carolina. Called 3DM it used an Eyephone HMD manufactured by VPL and a 6 DOF mouse.

The user interface involved a floating toolbox that contained 3D-icons to create geometry or to command other functions. Navigation is ensured through the use of a "magic carpet", an area that marks the boundaries of the tracking system. As the authors noted "remaining within tracker range is important because the virtual world will begin to tilt as the user moves farther out of range" (Butterworth et al., 1992, p. 136).

The toolbox remains attached to the user when they move and has icons to represent different actions.

Through the toolbox, the user can create simple shapes like cylinders, cubes, triangles or triangle strips. More complex geometrical shapes could be obtained by extrusion, twisting or translating the surface.

The user can scale, move and rotate the object or they can move, fly or *grab* the whole world. To analyse the world at different scales the user can "shrink down to bird size in order to add eyelashes to a model of an elephant and then grow to the size of a house to alter the same model's leg" (Butterworth et al., 1992, p. 136).

3.4.3.2 HoloSketch

According to the author (Deering, 1996, p. 59) HoloSketch is a "virtual reality-based 3D geometry creation and manipulation tool" to be used by non-computer scientists.

It supports several visualisation devices but in particular Deering focused his work on the FishTank-VR interface. A 6 DOF wand device made from a 3D mouse and a tracker couples the stereoscopic visualisation system.

The choice of FishTank-VR and the consequent use of a high-resolution CRT screen has brought several optimisations, great precision to the system. This, according to the author, can reach 0.5 mm. This is a result of several parameters from the size and the curvature of the screen to the *intraocular* distance of the user (Deering, 1992). As the author stated "HoloSketch is so accurate that one can hold up a physical ruler to a virtual object and make accurate measurements" (Deering, 1996, p. 56).

To create the 3D-interface Deering referred to traditional 2D CAD packages where a set of icons and menus gives access to commands. But as the author noted (Deering, 1996, p. 56) "there are several problems with duplicating this set-up in 3D": first of all the issue of "screen real estate" occupied by the interface, secondly the rendering resources to use it and last but not least the ergonomics of these interfaces. According to the author it actually takes more time to hit a widget located in a 3D-world then a traditional menu on a 2D-window.

In answer to these problems a new approach was followed. Here a single complex menu pops-up when a certain button on the wand is pressed. The menu, which fades from the background, is a "3D pie-menu" centred on the wand. When the user presses one of the 3D-buttons the menu flashes before it fades away. Sub menus appear on top of the main one.

Although this lets the user access a large number of tools in a small space it unfortunately requires "users to practice in order to accustom themselves to the process" (Deering, 1996, p. 57).

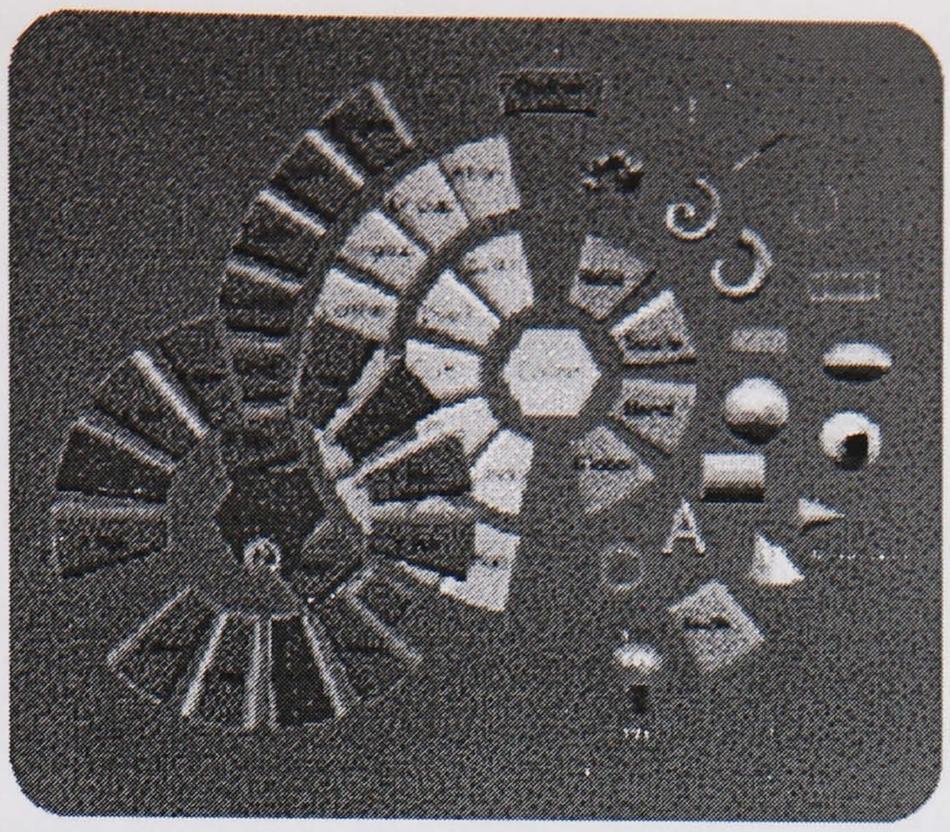


Figure 3.6: The 3D menu and a sub-menu developed in HoloSketch (Deering, 1996, p. 56)

HoloSketch was designed to create a number of geometric primitives including cubes, cylinders, spheres, 3D text and a free-form "toothpaste" shape. These shapes can be edited using the wand. For an object to be selected, the side buttons of the wand must be pressed while the wand is placed inside the object. As the authors reported "this action is very similar to physically grasping an object" (Deering, 1996, p. 57).

Further finer editing is possible by accessing the parametric data of the shapes. This can be done by invoking a 3D-property sheet where the user after selecting the desired field in the space with the wand, types the new value using 3D-text. In this way, for the first time the user can access the code from the virtual world.

3.4.3.3 VADeT

The VADeT system was developed as the answer to the question of whether it was possible to design at full scale. Chan et al. (1999a) noted that the design process usually starts with a 2D-based thinking operation and after that the designer moves on to 3D sketching. The designer would usually sketch in miniature, very rarely at actual size. The authors wrote (Chan et al., 1999a, p. 43) the consequence of this would be that "designers cannot project themselves into the space in the same

way as they are physically existing, the experience of seeing-as and reflection in-action is limited, and the scope of design is narrowed".

Nevertheless, the authors noted that the main problem of working at actual size is the limit to perceive "the entire site and its adjacent context [...] for immediately comprehending relationship among objects macro-wide" (Chan et al., 1999a, p. 50). This limitation is similar to the problem experienced by people who have difficulties perceiving the city pattern from a street level point of view.

This limitation however, is resolved by controlling the use of the scale. The user can work at a small scale to create large objects and later at full scale after the overall form is decided.

This system, like its precursor (Deering, 1995) uses a toolbox where a number of commands are placed including creation, editing, texture and save\load. The submenus are arranged in a tree structure and they are shown only when the main icon is triggered (See Figure 3.7).

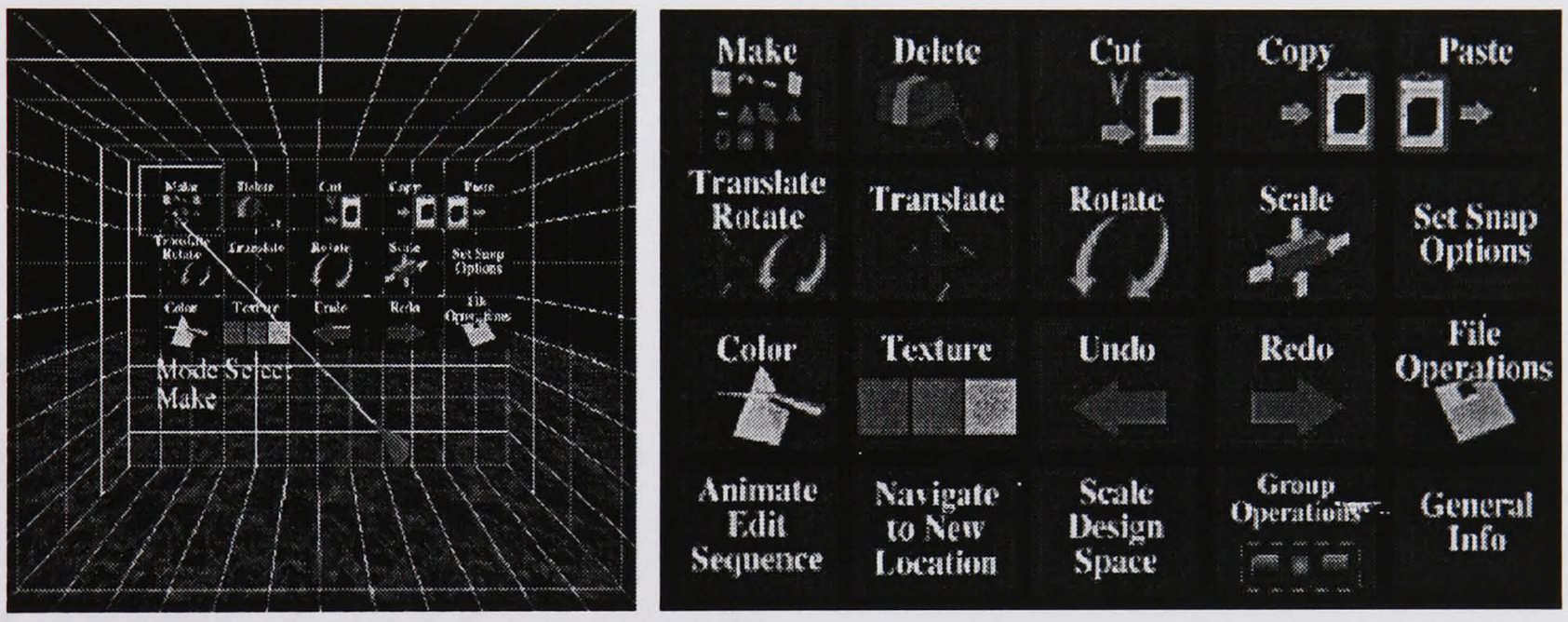


Figure 3.7: VEDeT initial view and the tool palette (From Chan et al., 1999a, p. 46)

The system provides 11 predefined geometric primitives. After the objects are created, the designer can modify their status by scaling, moving or rotating them. In addition colour and texture palettes allow the alteration of the visual surface appearance of objects (See Figure 3.8).

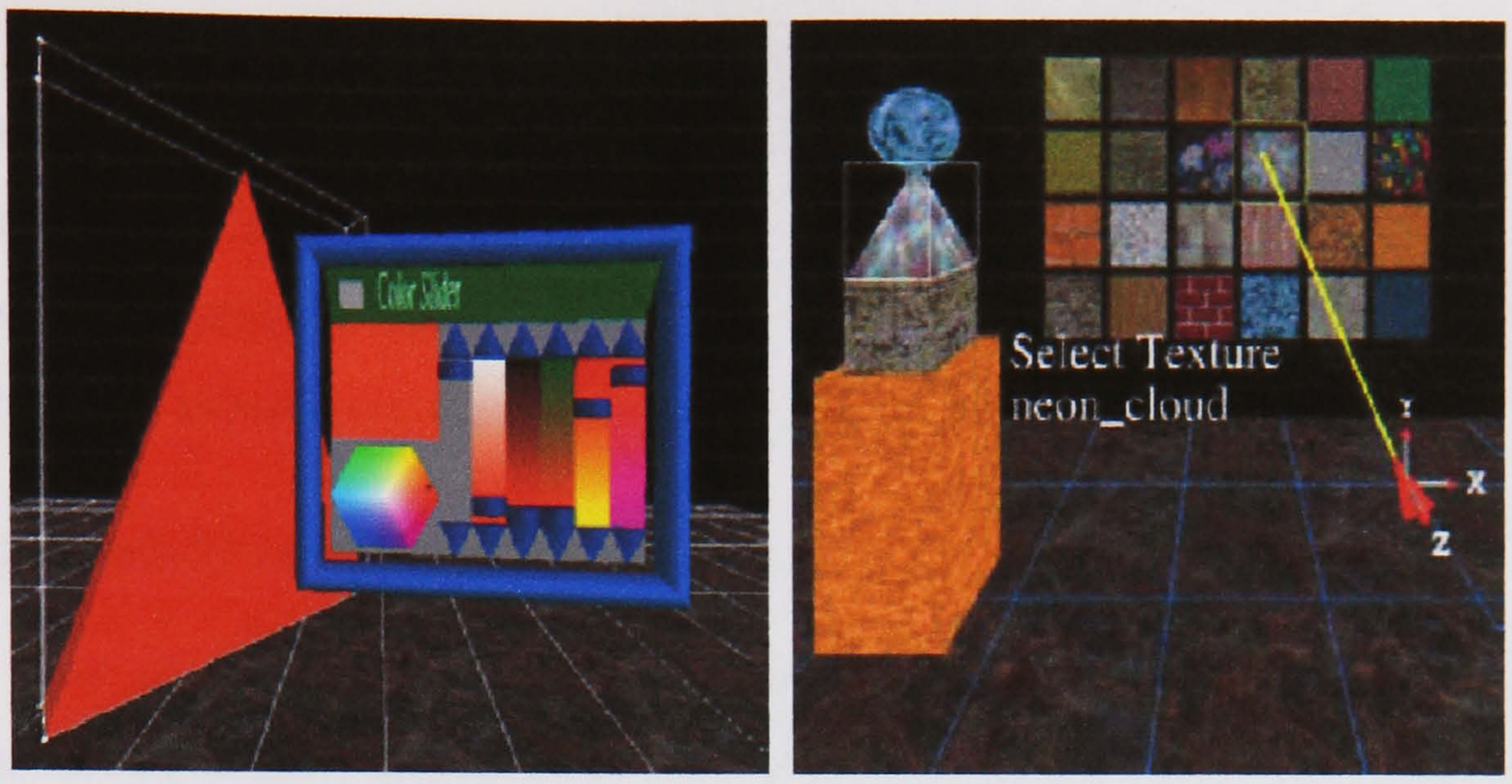


Figure 3.8: VADeT colour and texture palette (From Chan et al., 1999a, p. 48)

Due to the problems of the accurate placing of objects in 3D space, three types of translation, rotation and scale mode have been provided. The free mode, as the authors (Chan et al., 1999a, p. 47) reported "is useful for translating in a large distance, and designers can quickly place their entity at an approximate location". In the second mode a set of constraints can be applied to the action so that for instance a translation can be constrained to an axis. The third and last mode uses a snapping process to define the granularity of the space.

The system is coded in C++ using the C2 software library. It runs on a fully immersive 4-wall CAVE powered by two multiprocessor Sgi Onyx mainframes. A 3D sound device also provides also localised sound.

3.4.4 Particle-based design tools

Unlike the systems presented so far, *particle-based* systems do not create definite shapes. Instead, these applications *spray* polygons, or *particles*, into the space which are dynamically adjusted by the system. Since the system "welds" the particles into shapes in real-time, the user can progressively build new surfaces by spraying layers of particles over existing shapes.

Due to this simple approach this type of application usually does not require special hardware to be used. The computer is only required to render a number of polygons.

This approach is effective for approximate or organic shapes while it is not suitable for 3D-shapes that require a high level of precision. This technique also does not give the user any control over the mathematical representation of the shape. To overcome this problem some authors (Szeliski et al., 1992) have proposed hybrid spline/particle based approaches, as shown in the following examples.

3.4.4.1 Szeliski and Tonnensen's System

One of the first *particle-based* systems was developed by Szeliski et al. (1992). As the authors pointed out, until that moment particles were only used to model natural phenomena as a representation of finite elements, but they proposed the use of particle systems to create surfaces. The idea behind the system was to find a way to generate surfaces "by creating a number of particles in a plane and allowing the system dynamic to adjust the particles into a smooth surface" (Szeliski et al., 1992, p. 189).

The system was designed to be a surface modelling tool where the user could *spray* particles in space to create objects. The system also had a dynamic/automatic feature to create new surfaces as a result of a manipulation. In the example shown, starting from a sphere, the user can push into it at two points to create a torus and the system will automatically create the new particles required (See Figure 3.9).

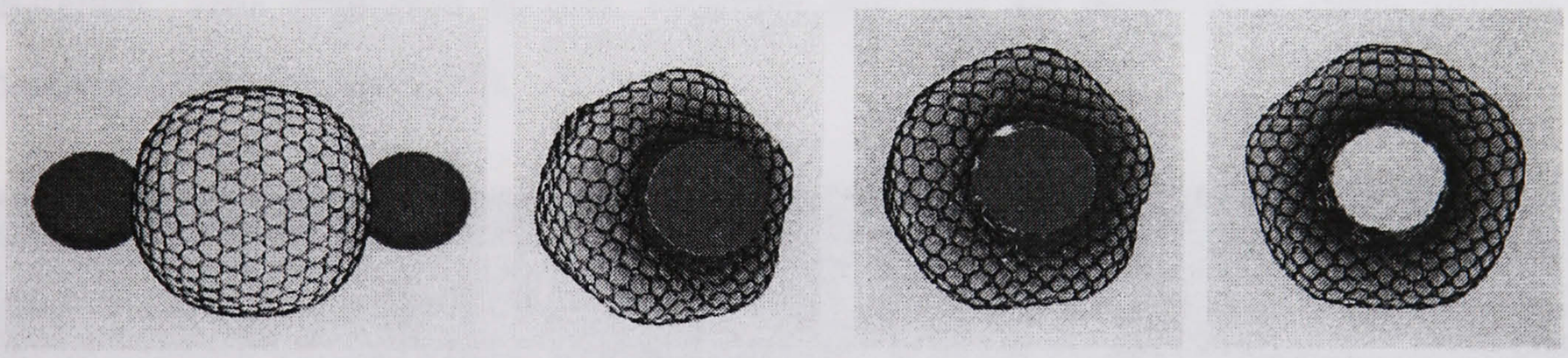


Figure 3.9: The example of a torus created from a sphere being pushed by two spheres (Szeliski et al., 1992, p. 191)

The user used a mouse to create and manipulate these particles and their depth also had to be interpreted. The research team tackled the problem by calculating the z-coordinate according to the nearby particles.

As the authors point out, a drawback is that with this system "it is hard to achieve exact analytic (mathematical) control over the shape of the surface" (Szeliski

et al., 1992, p. 191). For instance, in the case of the torus just mentioned, the torus would not be perfectly symmetrical. To solve this problem the authors suggest a hybrid spline/particle based system.

Although this did not take place in a VR environment it represents the first example of its kind and it inspired some of the applications discussed in the following chapter. The authors declared that "adding 3-D input devices for direct 3-D manipulation would be of obvious benefit" (Szeliski et al., 1992, p. 189).

3.4.4.2 Skin

As Markosian et al. (1999, p. 393) described it, *Skin* is a "particle based surface representation with which a user can interactively sculpt free-form surfaces. [...] A user interactively guides the particles, which we call *skin*, to grow over a certain collection of polyhedral elements (or *skeletons*), yielding a smooth surface (through subdivision) that approximates the underlying skeletal shapes".

The system allows the creation of complex shapes by initially constructing a representation of the underlying structure. Then the user can *oversketch* the pre-existing geometry with a surface, in a process that resembles placing the skin over a skeleton. The over-imposed structure is created according to an algorithm which produces a shape that roughly fits by offsetting the original skeleton by a certain distance. This approach is fast, efficient and can be done with basic graphics hardware (See Figure 3.10).

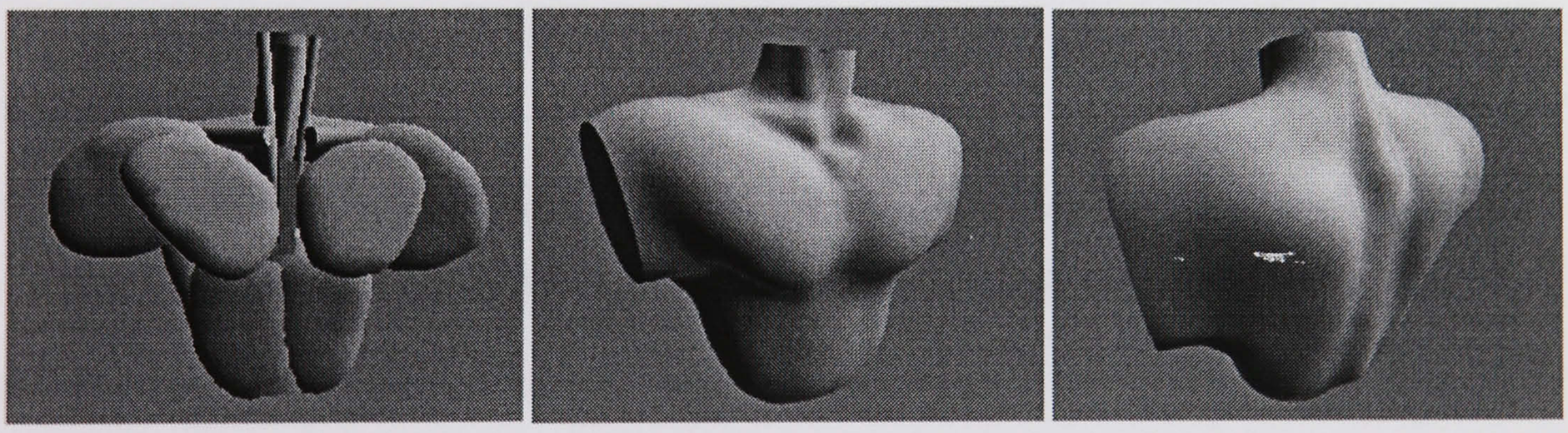


Figure 3.10: The model of a torso made with Skin starting from the skeleton at the left (Markosian et al., 1999, p. 400)

The authors have said that this method is good for the quick creation of *approximate shapes* but it is not suitable for objects that are required to have a high

level of precision. The user can also include constraints to the surface they create. For instance to simulate the space between toes in a foot, a curve drawn onto a surface can be used to create first indentation. Triangles intersected by the curve are split and the new structure re-triangulated. The new edge can then be moved to simulate the space between toes.

3.4.5 Voxel-based design tools

Similar to particle-based systems, voxel-based applications allow users to create shapes through the clustering of more elementary shapes. In this case the polygon, or particle, is replaced by a three dimensional entity called a *voxel*.

As shown in the application developed by Achten et al. (2000) voxel-based systems can be used to create precise geometries through the adoption of an algorithm to rigorously generate the voxels. As illustrated by Donath et al., (1998) this technique can however also be used to achieve an effect similar to particle-based systems, by spraying voxels in the space.

3.4.5.1 DDDoolz by Design System Group, Eindhoven

DDDoolz is a PC-based Desktop-VR "three dimensional voxel sketchtool" (Achten et al., 2000, p. 460) developed by the Design System Group at the Technical University of Eindhoven in The Netherlands. Voxels are 100-cm rib cubes that are used as building blocks.

As Achten et al. (2000) report DDDoolz was designed to fulfil a number of requirements:

- The achievement of a natural interface: the approach adopted followed the point and gesture paradigm without the use of 3D-buttons in space.
- Easy creation of objects in the virtual world.
- Easy manipulation of the object present in the world.
- Easy navigation that should add to the "spatial understanding of the design" (Achten et al., 2000, p. 460).

In addition the architecture of the system is such that no special hardware is required and the system can be used by normal PC clones. The system is based on a GUI where the main window shows a view of the virtual world using menus and icons (See Figure 3.11).

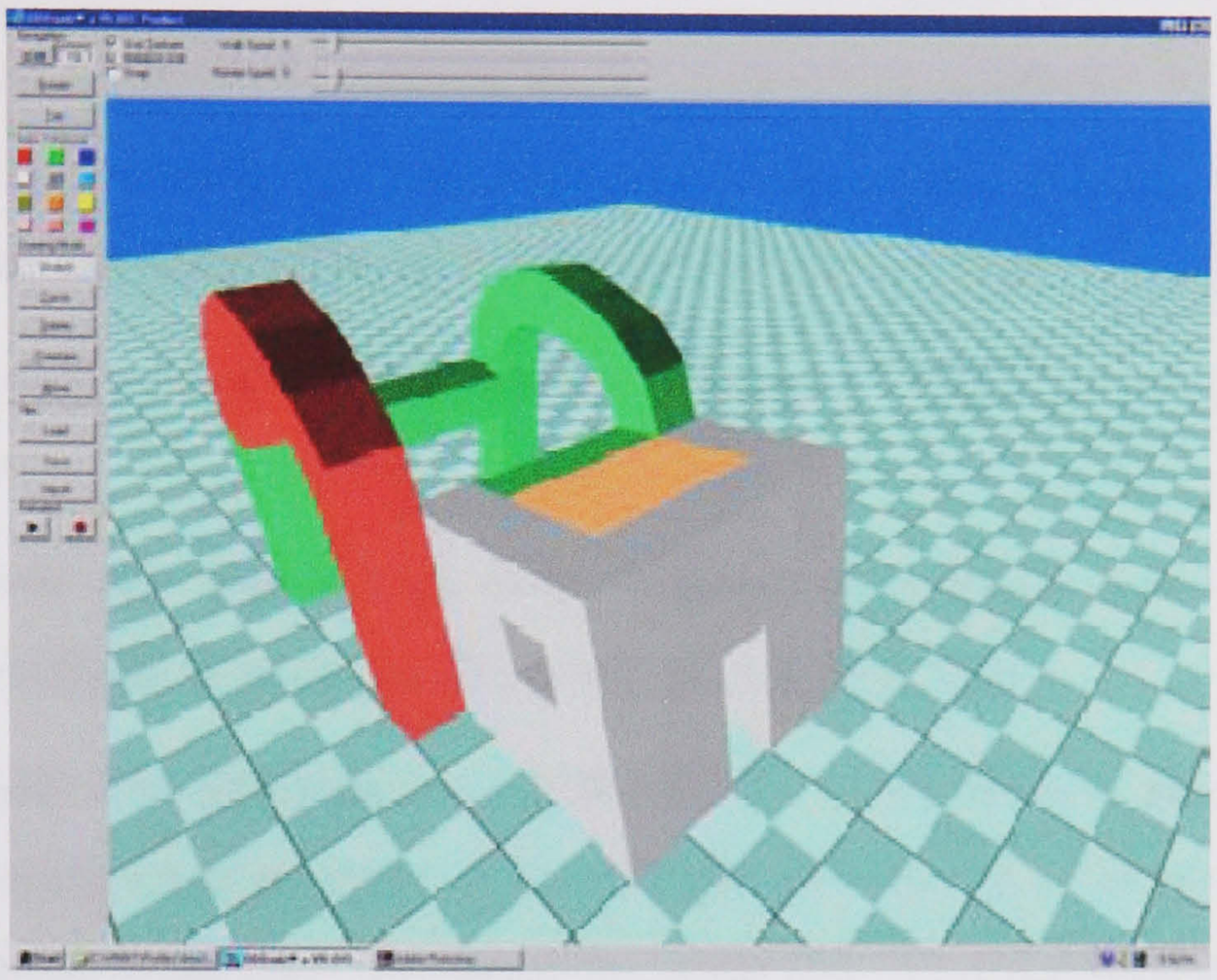


Figure 3.11: A screenshot of DDDoolz (Achten et al., 2000, p. 461)

DDDoolz uses building blocks to create shapes. When the software is in drawing mode the user can select a face of a cube and from it create a new adjacent cube.

As Achten et al. (2000, p. 461) state, "drawing in this way seems very natural as an interface: the side of the voxel you start with implicitly defines the drawing plane in which you are working". This approach was called the Face Orientation Method (FOM) by the authors.

The advantage of the FOM is that "sketching direction is inferred in a very intuitive way from the actions of the user and the starting point of the pointing device" (Achten et al., 2000, p. 461). Due to this simple approach the learning curve it was very fast. The system has been used by professional architects on actual projects and the feedback showed that this system could have a "strong benefit over traditional 3D CAAD representations" (Achten et al., 2000, p. 468).

The major disadvantage is that only certain shapes can be created unless the shape is approximated to a sum of cubes. In answer to this problem, to avoid the

limitation of what Achten et al. (2000, p. 262) called "orthogonal architecture" another drawing mode was introduced to allow drawing along curves. In this case the *drawing cubes* are rotated to follow the user's drawing direction. A set of editing tools such as *move*, *delete* etc. is provided. In addition holes can be created by deleting one or more cubes.

The use of 100-cm rib cubes helps the user to have a sense of scale. The size of the voxel was chosen to "force the user to keep any model limited to a rather sober outline, thus enforcing the character of the application" (Achten et al., 2000, p. 461).

Navigation in *walk* and *fly* mode is controlled through a standard click and drag sequence.

DDDoolz can import files from commercial packages and an AutoCAD macro was developed to allow the importing of DDDoolz files.

DDDoolz was developed using WorldUP by Sense8 and geometries were coded instantiating a box primitive in BasicScript, WorldUP's scripting language. As soon as a voxel is created the scripting properties of the previous object are transferred to it so that the object remains consistent (e.g. has the same colour). The authors have also tested the system with other devices such as 6D mouse, *flock of bird* and voice control. The group also plan to introduce a shape recognition algorithm in order to pass from voxel-based shapes to parametric ones.

3.4.5.2 VoxDesign

According to the authors DDDoolz was inspired by another tool called VoxDesign. This was developed in 1995 at the University at Bauhaus of Weimar (Donath et al., 1998). In this system 2.5x2.5x2.5 cm voxels could be "sprayed" inside the 3D space.

"A crucial difference of DDDoolz lies in the 1 m^3 voxel size which truly restrict the user to a sketch-like approach. The FOM of DDDoolz also provides additional implicit structuring so that actions yield structures and spaces that can have a ready architectural interpretation" (Achten et al., 2000, p. 463) while in VoxDesign the shapes created retain an artistic feeling rather than a geometrical feel.

3.4.6 Free-form applications

The idea of sketching free-form shapes directly into a 3D space rather than on a 2D surface has been presented in the previous paragraph on voxel-based systems. A different approach is taken by those systems which the literature refers to as *free-form* (Wesche et al., 2000; Markosian et al., 1999) or *spline-based* (Wesche et al., 2001) applications. These systems often use 2D curves to create and control 3D-surfaces.

As Wesche et al. (2001, p. 168) noted the spline-based approach has five important advantages over the other systems such as voxel-based ones:

1. "closed-form parametric representation"
2. "easy transfer into standard CAD packages"
3. "fast triangulation and evaluation algorithms"
4. "infinitesimal smoothness of curves and surfaces"
5. "efficient deformation algorithms based on variational modeling"

The following sections will introduce a number of free-form systems, starting from the groundbreaking application which was developed at the University of Utah by James H. Clark (1976) on the very same hardware engineered by Sutherland (See Section 2.3.4.1) up to today's modern systems based on CAVEs or Virtual Tables.

3.4.6.1 Clark's Experimental CAD in 3D

Clark's system, originating in 1976, was the first example of free-form VRAD (Virtual Reality Aided Design) system. The author described it as "an experimental system for computer-aided design of free-form surfaces in three dimensions" (Clark, 1976, p. 454).

Using this visionary system, the user could create surfaces using an HMD and a mechanical wand. The system used Sutherland's *Sword of Damocles*, the first HMD using a mechanical wand, which had been designed a few years before at the University of Utah (See Section 2.3.4.1).

As the author explained, the system was created with 3 goals in mind (Clark, 1976):

- Creation of objects directly in a 3D-environment. The user should be able to see the environment though a 3D-visualisation device and manipulate it with a 3D-input device.
- Real-time response, "as the designer moves his head about in the environment and makes changes to objects' shapes, all changes should instantaneously appear on the display" (Clark, 1976, p. 454).
- The user friendliness of the system. As the author stated (Clark, 1976, p. 455) "the third goal of the system was to provide an effective mathematical formulation for the designer that required him to have little or no mathematical knowledge of the formulation".

This system already included most of the important features of modern free-form VRAD systems although it was constrained by the limits of the technology of that time. The system could achieve real-time behaviour by using a special processor which refreshed the display and calculated the matrixes necessary for the display of the correct point of view. The main processor was used to access and manipulate the data structure.

The wand used a mechanical system to track the position of the pointer. As the author wrote (Clark, 1976, p. 457) "the wand's position is sensed mechanically by recording the lengths of three monofilament lines that extend from the wand handle to three housing mounted on the ceiling" (See Figure 3.12).

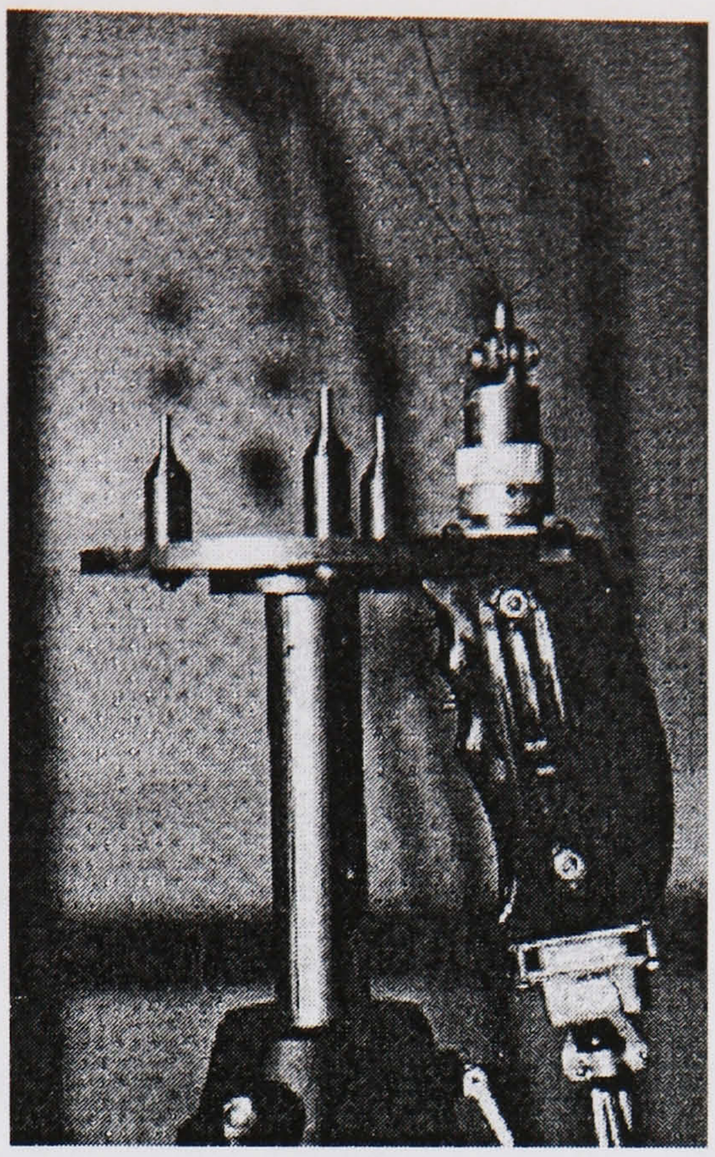


Figure 3.12: Clark's mechanical wand (Clark, 1976, p. 458). Note the three filament lines

At that time the two mechanical devices, the HMD and the wand, severely limited the usability of the system. As Clark (1976, p. 459) wrote "the most annoying problem is caused by the position sensing mechanism. Because of their physical proximity, the mechanical sensing mechanism of the HMD and wand interfere with each other, making it difficult to maneuver about in the room".

3.4.6.2 Teddy

In 1999 at SIGGRAPH Takeo Igarashi et al. (1999, p. 409) presented Teddy, a "sketching interface for quickly and easily designing freeform models".

The new idea behind Teddy was to automatically create *plausible* 3D surfaces from 2D freeform strokes sketched interactively on the screen. This placed *Teddy* half way between non-photorealistic, free-form and semantic-based systems.

The resulting models have the *handcrafted* appearance usually achieved with non-photorealistic systems. In addition it created 3D surfaces using 2D curves as in free-form applications. The interface is limited since commands are handled through different strokes rather than by using menus and windows. Commands are inferred

through the interpretation of strokes as gestures, as in semantic-based systems (See Figure 3.13).

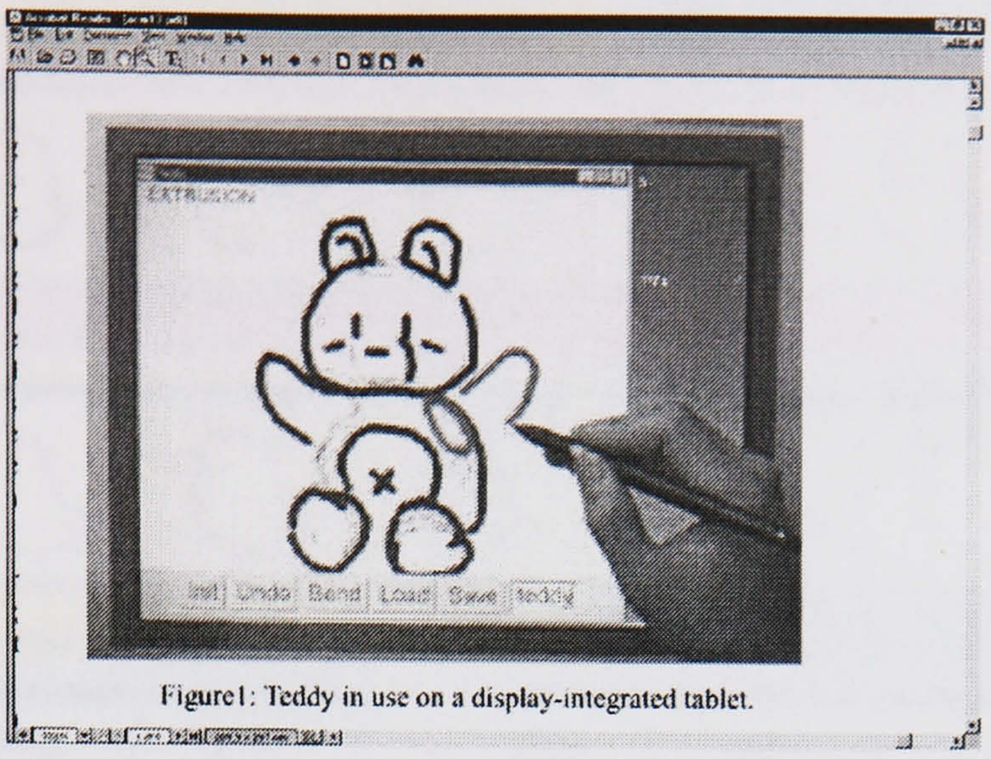


Figure 3.13: An image of Teddy's interface (Igarashi et al., 1999, p. 409)

The authors emphasised that this approach had the advantage of being very intuitive and "even first-time users can create simple, yet expressive 3D models within minutes" (Igarashi et al., 1999, p. 409). In addition the system is designed to allow *incremental learning*: the user can start by using simple tools and then move on to using more complex techniques.

Like other VRAD systems, this technique "is designed for the rapid construction of approximate models, not for the careful editing of precise models" (Igarashi et al., 1999, p. 409).

Rather than starting with a primitive shape and then applying several transformations to it as in the case of traditional CAD programs, in Teddy the user just starts by sketching.

The user takes a pen to draw 2D shapes on the screen or on a digital table. The use of widgets is limited to operations like save and load and Teddy's modelling functions are activated through interactive strokes on the screen (See Figure 3.14).

Figure 3.14: A sequence of screenshots showing Teddy's interactive modelling technique (Igarashi et al., 1999, p. 410)

When the user draws a stroke on the canvas, the system automatically connects its starting and ending points. It then constructs a *plausible* 3D model based on what the authors call the "*inflation*" process: "the system inflates the closed region in both directions with the amount depending on the width of the region: that is, wide areas become fat, and narrow areas become thin" (Igarashi et al., 1999, p. 412). Due to the nature of the method, the object must have a "spherical topology; e.g., the user cannot create a torus" (Igarashi et al, 1999, p. 411).

Once the object is created the user can the edit it in various ways. They can draw on the surface, extrude profiles from the model, cut the model and smooth the surface. These operations are done in real time and interactively according to the nature of the strokes which are interpreted and then the corresponding action is performed on the model. The result is a very simple and effective interface than can be easily used by non-computer scientists.

The prototype is written in Java™ and can run on a normal PC without any dedicated hardware.

3.4.6.3 Digital Tape drawing

Another relevant example of a free-form application is *digital tape drawing* where the metaphor of using tape was borrowed from the automotive industry. Hand made real size drawings of vehicles are usually not done by using pencils but by using black photographic tape laid onto paper. The designer would unroll the tape with one hand sliding the other hand to fasten it to the work surface. This method has a fundamental advantage in allowing the creation of smooth curves without the need for French curves.

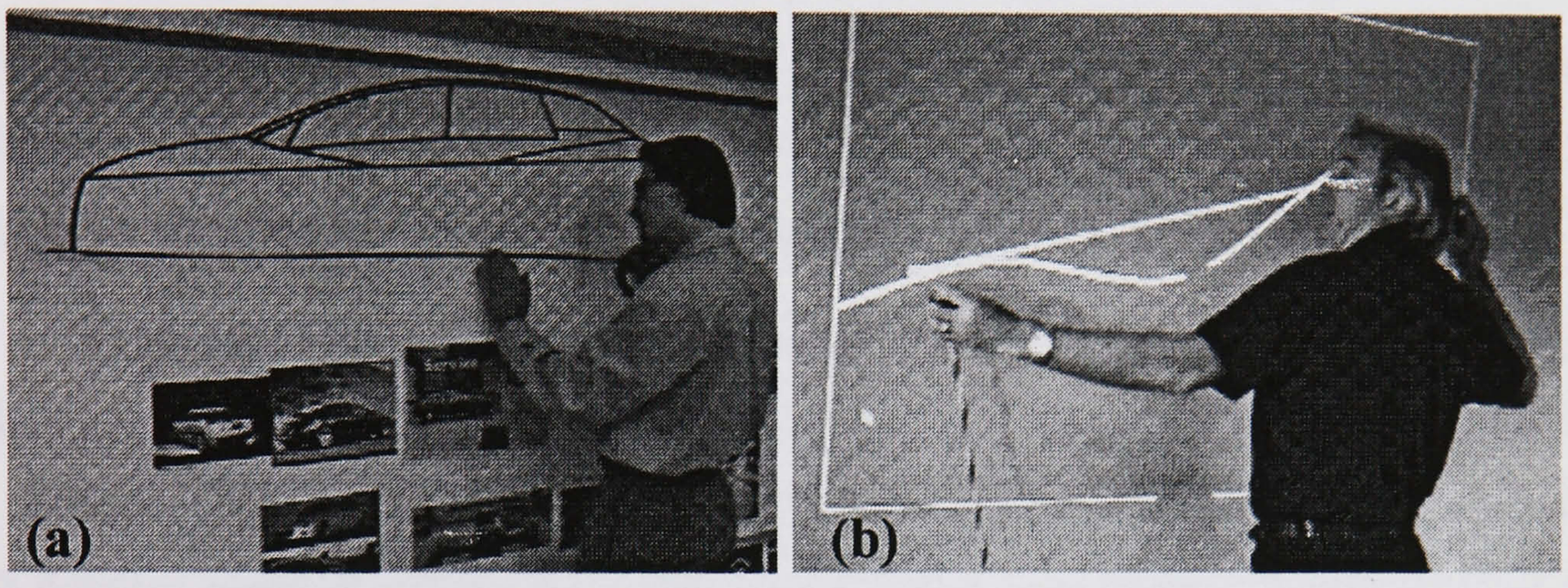


Figure 3.15: Traditional (a) and digital (b) tape drawing (Balakrishnan, R. et al., 1999a, p. 161)

The idea was borrowed by VR technologists and the result is now commonly called the Digital Tape Drawing approach. Balakrishnan et al. (1999a) have developed a system that would retain "much of the fluidity and affordances of the physical technique, while providing the advantages inherent in using electronic media such as storage, retrieval, lossless transfer" (Grossman et al., 2002, p. 121).

After the development of a first prototype only capable of drawing 2D curves the research group investigated the possibility of a more complex system that would support 3D curves (Balakrishnan et al. 1999a). The new system follows the method of creating 3D surfaces which was proposed by Cohen et al. (1999). This method is based on a two-step approach: first the user draws a "shadow" curve that "defines the shape of the curve in the third dimension" (Grossman et al., 2002, p. 122) and then they draw the final curve on top.

The *shadow* curve, as Cohen et al. (1999, p. 18) explained, is "a 3D curve obtained by projecting another 3D curve along a fixed vector, which we call the *projection* vector, onto some surface". The projection vector used by Cohen was always the Y-axis, orthogonal to the world plane, but the same principle could be applied in any other direction. The advantage of this approach is the ability to specify "3D curves with 2D input from a single point of view" (Cohen et al, 1999, p. 17). Following this approach and with the use of the Digital Tape Drawing paradigm the user can create 3D curves.

The user, who can use both hands to interact with the systems, can use the two trackers in the same way that they would lay the tape onto the paper and afterwards the computer creates the relevant curve. The operation is repeated twice, first to create the *shadow* and then to generate the proper surface.

The main problem of using this approach is the fact that "tape drawing is inherently a 2D technique, and all curves are initially drawn on 2D drawing planes. [...] Hence there is the need to switch back and forth between orthographic view and 2D drawing plane(s) and the perspective view that is necessary for inspecting the 3D model" (Grossman et al., 2002, p. 123). The authors solve this problem in two ways.

First they interpolated the viewing matrix over 50 frames from the orthographic view to the perspective one. In this way the transition is progressive from orthographic to perspective view and vice versa.

Secondly, to control the transition phase, they propose a method called *OrthoTumble*. As the authors explained most CAD systems have a *Tumble* camera that "gives the user the sense of controlling a two degree-of-freedom turntable on which the model sits" (Grossman et al., 2002, p. 123). With the *OrthoTumble* mechanism, the user follows the same paradigm: the *non-dominant* hand is employed to control the point of view whilst the dominant one handles the creation. Previous researchers (Balakrishnan et al. 1999b) have proved that this was successful and it gives a 20% increase in speed according to the author.

Whenever the user wants to change the point of view therefore they use their non-dominant hand to click and drag to control the *Tumble* camera. The system immediately fades into perspective mode and lets the user choose the point of view.

Once the drag is completed the system fades again to the closest orthographic view. This system "allows for the user to quickly inspect the model in a 3D perspective view and then automatically return to appropriate orthographic view to continue drawing curves" (Grossman et al., 2002, p. 124).

The two handed interaction is also used for pan-zoom operations. In this case the authors implemented a method initially developed by Kurtenbach et al. (1997) where the user could pan and zoom following the metaphor of a picture printed on a rubber sheet.

According to this principle when the users moved their hands together or apart the system would zoom out or in. If the user moved their hands apart at a constant distance the system would pan the image. To allow independent panning and zooming operations, which were difficult to obtain following Kurtenbach et al.'s (1997), approach Cohen's research team introduced a threshold that filters the distance after which the command is triggered (Grossman et al., 2002).

Although this system proved to be very powerful it had a major drawback in its complexity of control. As the authors admitted, "our system is by no means *walk and use* and therefore the researcher who constructed the system spent time training the user on the basic capabilities of the system" (Grossman et al., 2002, p. 127).

3.4.6.4 The Responsive Workbench (RWB™).

Another system based on two handed interaction is the Responsive Workbench (RWB™). As the authors (Wesche et al., 2000; Wesche et al., 2001) wrote, this is a "two-handed 3D system for free-form surfaces in a table-like Virtual Environment" (Wesche et al., 2000, p. 83).

The developers' goal was to allow the users to draw shapes using their two hands to control a *Virtual Workbench* environment (See Section 2.4.1.3). As the authors noted in addition they wanted to create a system that could easily create closed curves in space (Wesche et al., 2000).

As in the case of the Digital Tape Drawing (See Section 3.4.6.3) the creation process is divided into two steps. In RWB™ surfaces are created "indirectly by drawing the primal curves of the objects" (Wesche et al., 2000, p. 84). As the authors

(Wesche et al., 2001, p. 168) stated "we do not support *direct* surface drawing. We believe that a surface has too many degrees of freedom for defining all of them just with a single drawing sweep".

Therefore the design is made using traditional drawing planes with the difference that in RWB™ those planes are in space and can be interactively moved. The user first draws the skeleton of the curve and then it creates the surface. Once defined the surface can be interactively deformed. The system can draw 3D curves as well as 2D ones since "2D drawing is a very fundamental technique" (Wesche et al., 2001, p. 169).

The user draws the surfaces using a tracked stylus as the input device while the "non-dominant" hand is used to control the environment. The designer can also, by using the non-dominant hand, control the position of the object, define symmetry planes or choose the drawing plane onto which curves are projected.

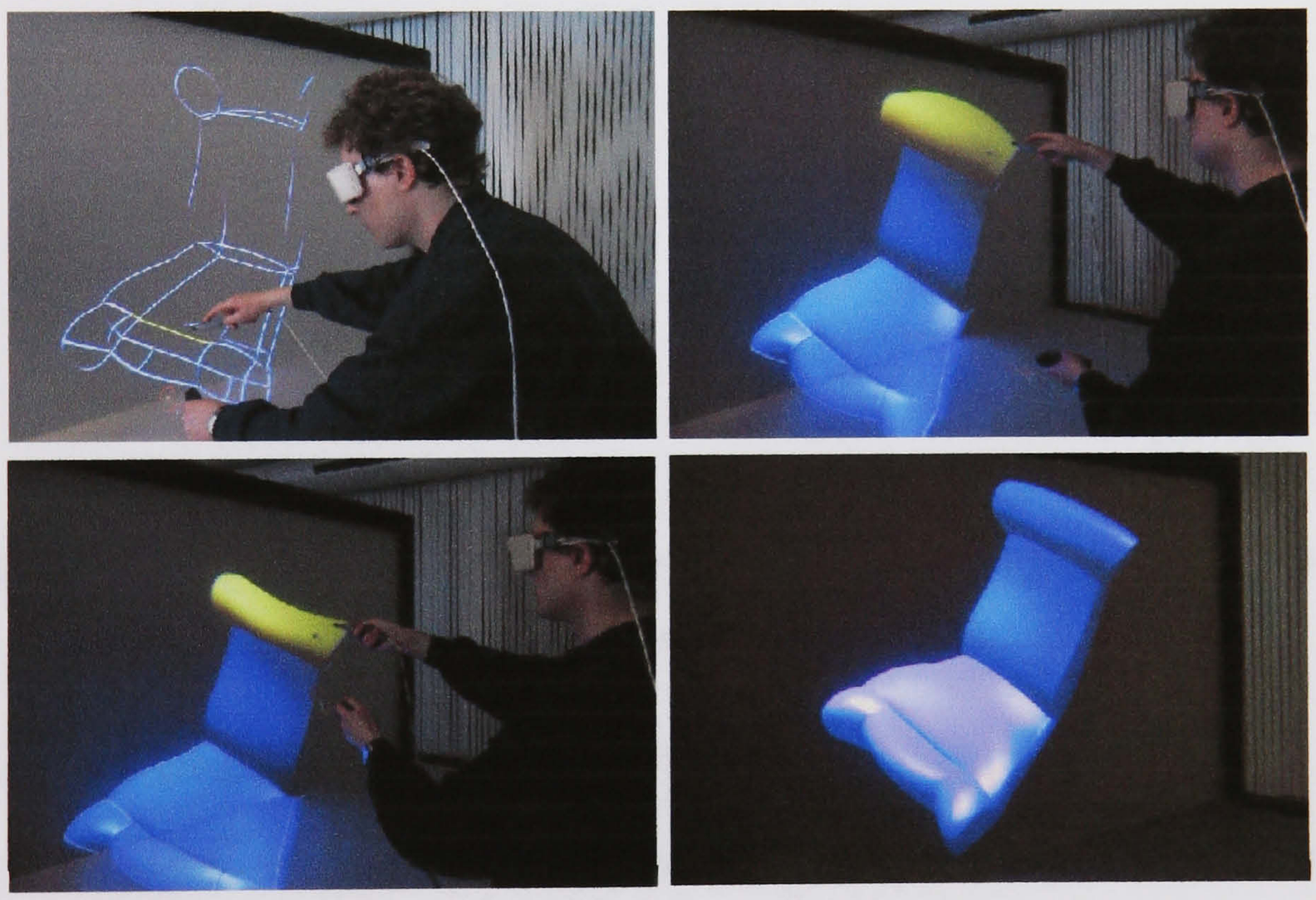


Figure 3.16: Some views of a designer using the RWB™ (Wesche et al., 2001, p. 172)

The menus are integrated into the modelling spaces to allow support in a "non disturbing way" (Wesche et al., 2000, p. 88). Pressing a button on the pointer activates the set of tools relative to the chosen object. Since the position of each hand is tracked the toolbar follows the movements of the user's wrist. In this way, the user

can simply select a tool by rotating their hand and pointing to the desired command icon (See Figure 3.17).

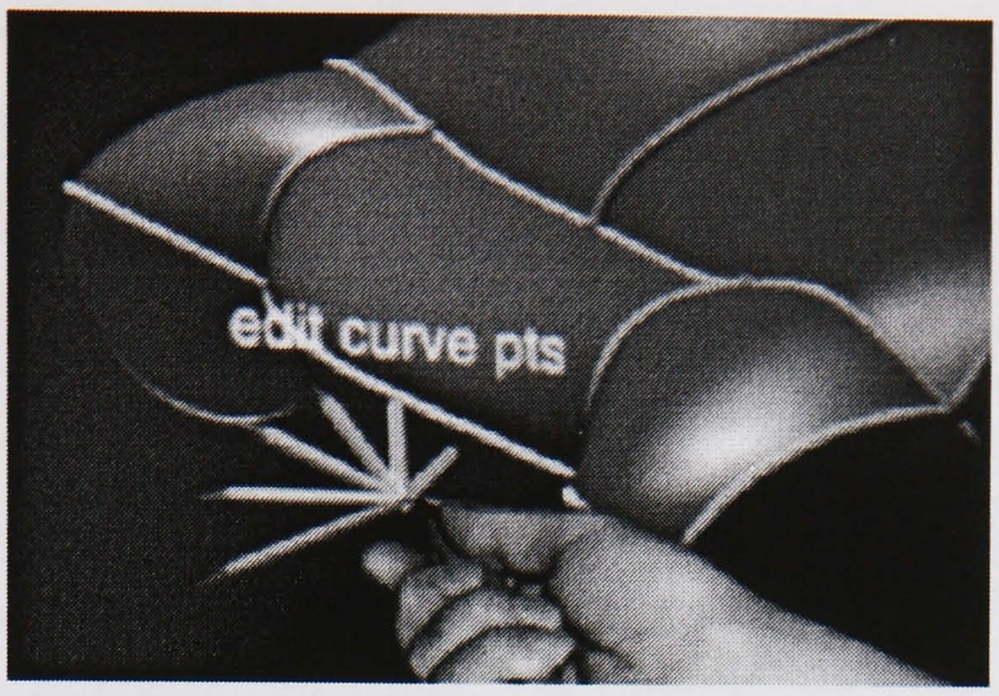


Figure 3.17: A view of the toolbar used in RWB™ (Wesche et al., 2001, p. 171)

To minimise the size of the toolbar the authors have exploited "the three-dimensionality" of the system. Pointing to objects with different inclinations activates different types of selection (See Figure 3.18).

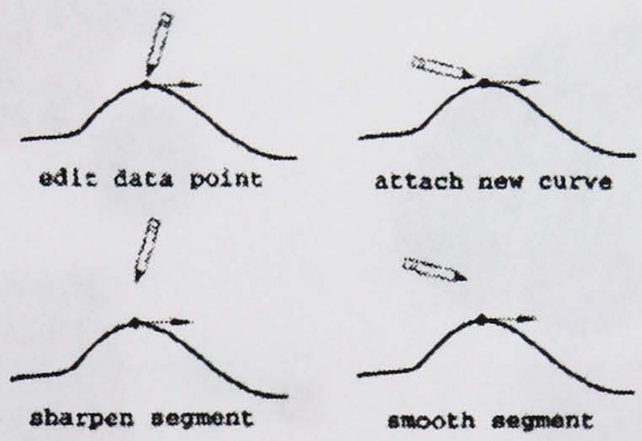


Figure 3.18: Different operations according to different pointer positions in RWB™ (Wesche et al., 2000, p. 89)

3.4.6.5 Spacedesign and the Eraser Pen

Two other Virtual Workbench-based systems developed at the IGD in Darmstadt, Germany, are Spacedesign developed by Fiorentino et al. (2002) and the Eraser Pen described by De Amicis et al. (2002). The two systems use the same hardware: a Barco Baron (Barco, 2002) and three Polhemus Fastrak magnetic trackers (Polhemus, 2002) connected to a pair of stereoscopic shutter glasses and two pointers.

Although the *Spacedesign* and *Eraser Pen* share the same hardware they focus on different issues related to the interface.

The first, as Fiorentino et al. (2002) wrote, aims to address the difficult exchange of information between the user and the system. "In virtual reality it is difficult to communicate information to the user: floating messages or external objects can be unfriendly and confusing" (Fiorentino, 2002, p. 481). As an answer, they proposed a "Windows-like" interface integrated within the 3D world (See Figure 3.19). In this way accessing information about the object becomes "simple, and not intrusive in the virtual workspace" (Fiorentino et al, 2002, p. 481). As the authors report this interface has proved to be very useful. It is an excellent feedback tool for novice users because it provides them with all the status of the system and all the information required.

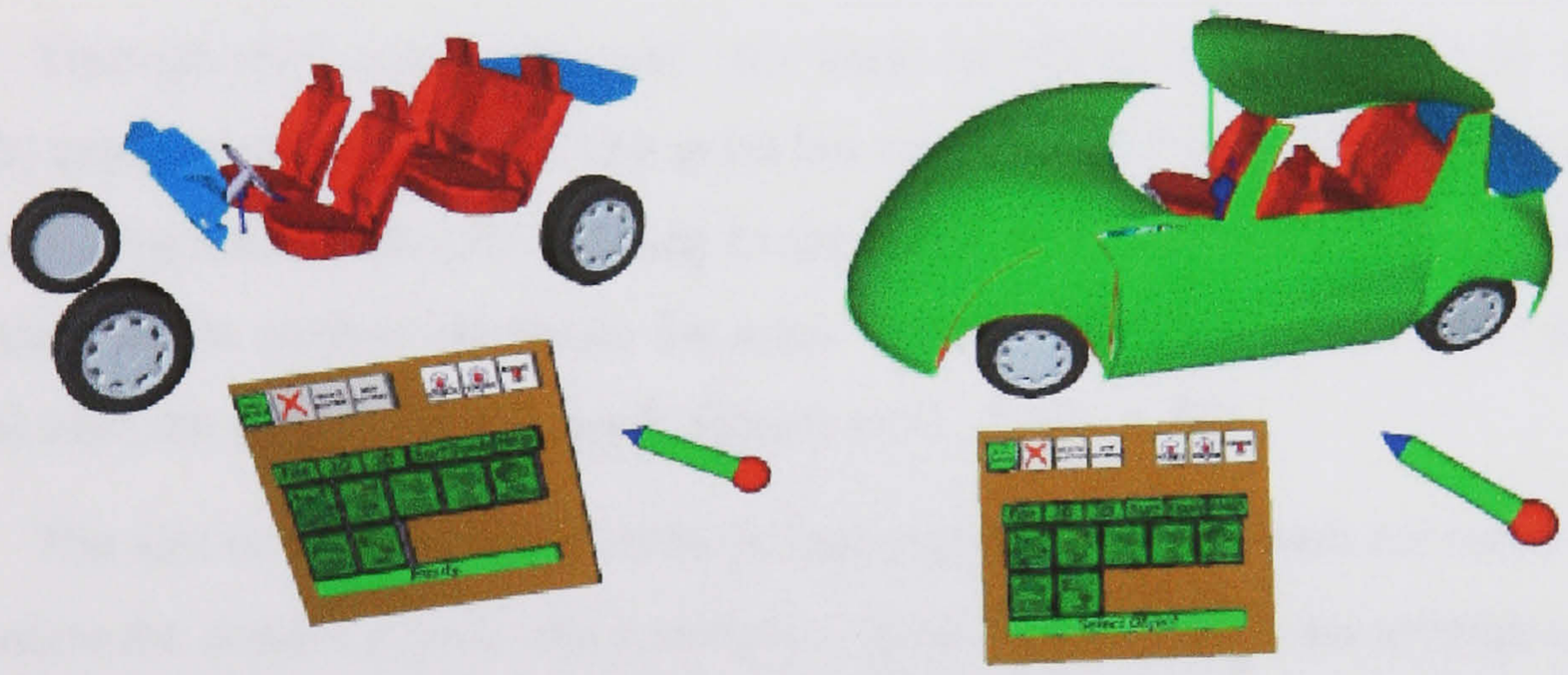


Figure 3.19: Spacedesign's interface (Fiorentino et al., 2002, p. 481)

The second system, the Eraser Pen, tries to address the difficult task of creating and then editing 3D-shapes in the space. As the authors (De Amicis et al., 2002, p. 465) noted, most free-form applications "use the control points for creating and modifying the curves, thus requiring a certain mathematical knowledge and experience to know how to modify these points in order to obtain the desired surfaces".

Alternatively this system, suggested a new approach where the metaphor of a pencil and an eraser is used to create 3D-surfaces. Deleting and editing a shape is a fundamental feature of a modelling tool according to De Amicis et al. (2002). In

practice the system constantly checks the position of the pen against the previous points and when the angle between the vector drawn and the sum of a number of previous ones is below a certain value the system interprets it as an inversion of direction and therefore it starts a deleting action.

Following the metaphor of the pencil and the eraser the user can create and delete shapes, entirely or partially, just by retracing back along the path of the curve. "By means of this technique if the resulting polyline, curve or surface is not exactly what the designer wants, he/she can delete part of the sketch just going back and do it again until he/she reaches the final shape" (De Amicis et al., 2002, p. 465).

3.4.6.6 CavePainting

CavePainting was conceived the answer to a different issue: the creation of 3D artistic works. The system as presented by Keefe et al. (2001) is a four wall CAVE-based environment especially designed for artists, and the metaphor adopted is the act of painting.

Through this device the user can draw in 3D as a painter would with a canvas, using strokes of colour. The artist has control over the space and with preset tools can create an artistic 3D painting in the space. "The artwork is created entirely from these basic strokes elements, the artist must be provided with a great deal of control over the type of strokes used" (Keefe et al., 2001, p. 85).

The aim of the system is neither to interpret the strokes which are made nor to interpolate the resulting path into a surface. "CavePainting does not attempt to be a modelling system [...] CavePainting aspires to be an extension of painting to three dimensions" (Keefe et al., 2001, p. 90). The strokes are layered to create a scene. The artist can decide from a choice of different kinds of strokes resembling different brushes, sponges or even a virtual bucket (See Figure 3.20 and Figure 3.21).

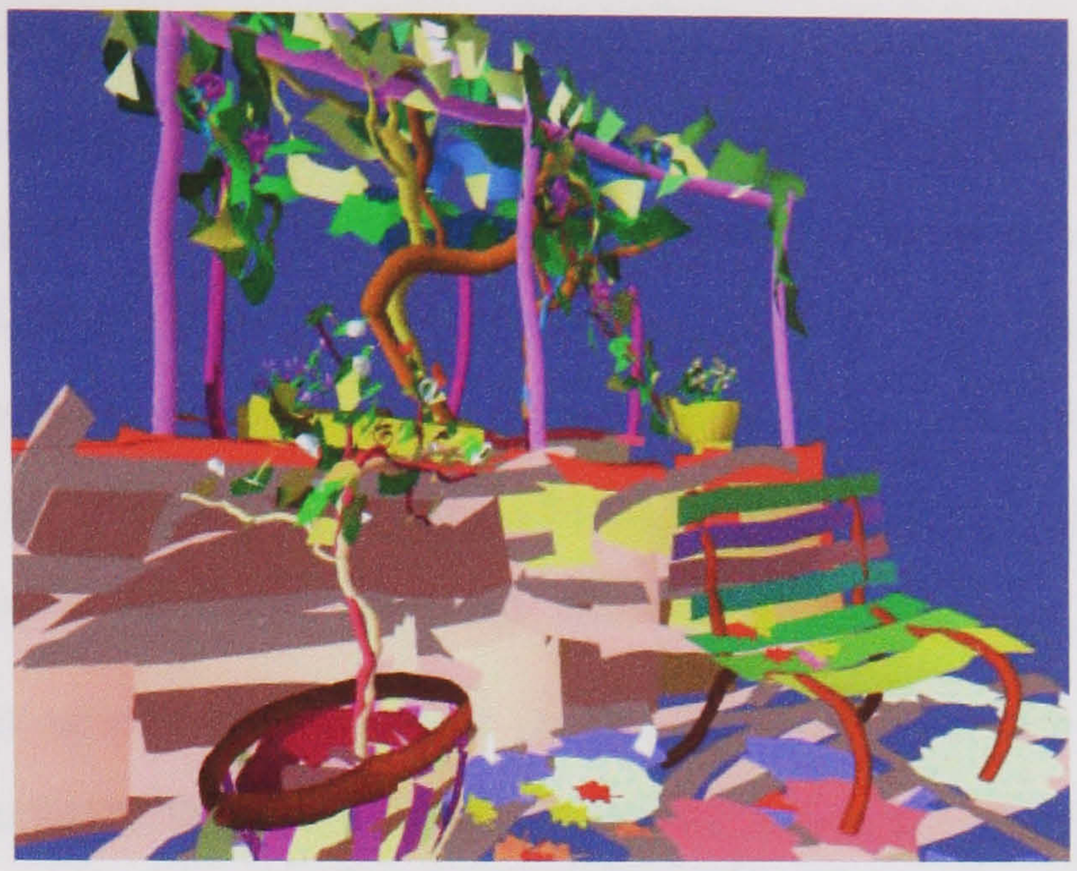


Figure 3.20: A *3D painting* made with CavePainting (Keefe et al., 2001, p. 90)

Consequently, the system interface is intended to create the feeling of using traditional painting tools. The artist can select the virtual painting tool by dipping a real brush into a real cup placed on a table close to the edge of the real sidewall. When the user dips the brush into the cup the conductive cloth at the end of the brush closes a circuit with the cup, and the action is consequently interpreted as a change of brush. The same augmented approach can be followed with a tracked bucket that can be used to splash virtual paint rendered according to the speed and position of the bucket (See Figure 3.21).

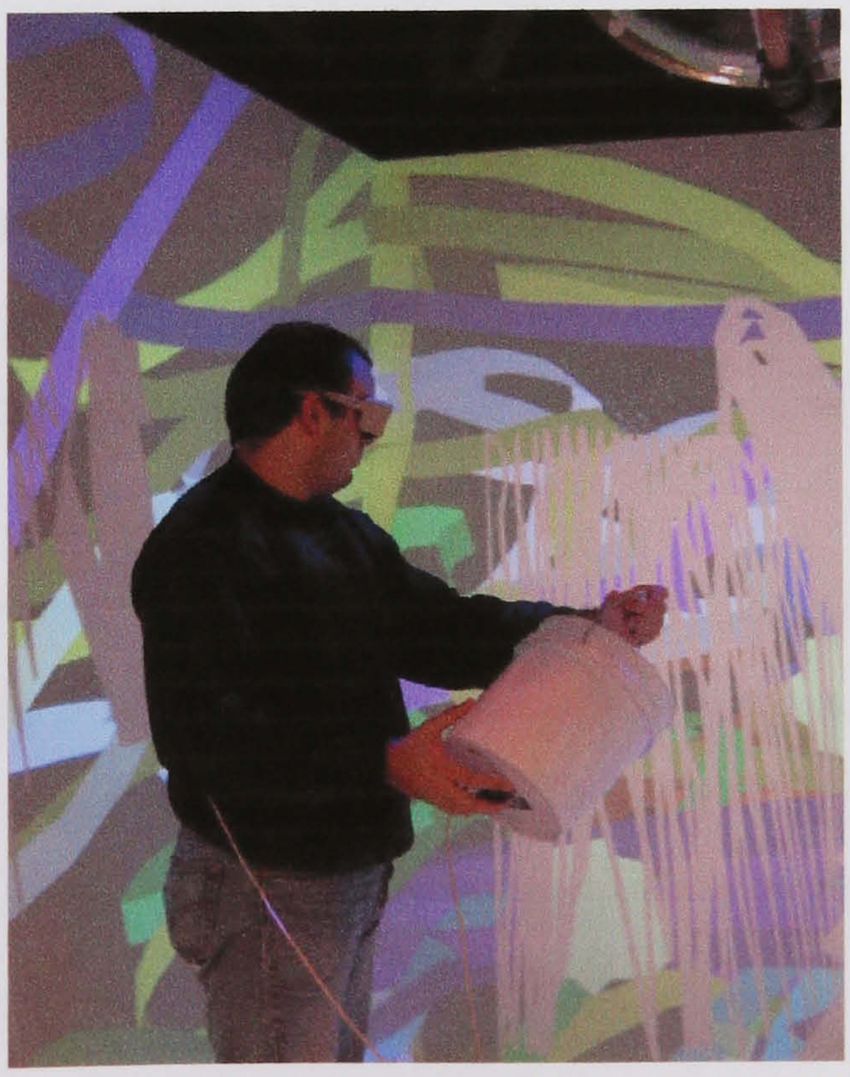


Figure 3.21: An artist using the bucket in the CavePainting environment (Keefe et al., 2001, p. 89)

The main difference between the real world and CavePainting is that there the artist is not constrained to the rules of physics. As the authors wrote "The Jackson Pollok stroke drips a line of virtual paint, reminiscent of the drip paintings done by the great expressionist artist. In virtual reality, we are free from some of the limitations imposed on Jackson Pollock. For example, our paint does not need to drip according to gravity. We take advantage of this, and are able to drip on all six sides of the cube defined by the Cave" (Keefe et al., 2001, p. 86).

Due to the artistic nature of the system it must have very effective colour selection. The interface follows an approach close to Deering's (1995 and 1996) (See Section 3.4.3.2) colour selection tool: a 3D colour picker where the RGB values, lightness and hue can be selected by dipping the brush into it. A sphere attached to the end of the brush provides the necessary feedback showing the colour selected. The colour picker is called up through gesture recognition when the user draws a circle upwards with the brush.

For advanced users another a two-handed option was provided. Using a glove in the non-dominant hand the artist can have control over position. Working artists tend to move a lot, they paint, step back, look and then paint again. The user can change the work of art at any time during the creation process and edit it through a timeline.

3.4.7 Virtual Clay Modellers

Virtual clay modellers are a relatively recent evolution in free-form systems. From these they have inherited an approach towards the creation of the geometry, but the interface and the consequent creation processes are based on the metaphor of working with clay.

As Kameyama (1997, p. 197) wrote, "the key component of its hardware is a special input device with a 3-D position tracker and a tactile sensor". Another author however reported the development of similar systems using a simpler 3D-mouse (Pratini, 2001). The advantage of virtual clay modellers is the possibility to create relatively complex shapes in a very intuitive fashion.

On this regard Matsumiya et al. (2000, p. 67) noted: "to apply geometrical and topological operations to free-form objects, users must have enough mathematical knowledge and flexibility of special recognition. So, the user's attention then focuses on the model decomposition into patches and the continuity relationship between them, rather then on the shape itself".

Systems developed so far usually fall into two categories: simple shape modellers and more complex *physically based* modellers. In the former "shape deformation is expressed by simple formulas without complex calculation" (Matsumiya et al., 2000, p. 67). The latter augment "geometric objects with physical attributes such as mass, damping and stiffness distribution. Geometric parameters can be hidden from the user by providing natural, force-based interfaces that facilitate *direct* manipulation of solid objects through virtual sculpting tools" (McDonnell et al., 2001).

Several authors have stressed the tremendous acceleration of benefits being brought to the design process with the use of Virtual Clay Modellers (Matsumiya et al., 2000; Kameyama, 1997). Kameyama (1997) has proven in practice the efficacy of a *virtual clay* modelling system connected to a rapid manufacturing machine. The author reports on the outcome of an experiment where a number of users were asked to design a set of plastic lenses. According to the author, the entire process of designing and manufacturing of the plastic model took on average less than one hour.

3.5 Conclusions

This chapter illustrated several systems that make use of Virtual Reality as a design tool. Specifically this chapter gave a taxonomy of VRAD systems used in the early stages of the design process. A number of applications using different techniques were presented.

The following chapter will outline the role of Human-Computer Interfaces (HCI) in Virtual Reality systems. Further, it will introduce a number of theoretical concepts and give a rigorous classification of several geometrical aspects related to the development of HCIs. This will include the classification of a number of reference systems used to describe mathematically aspects of the interaction using

computers. This chapter will also illustrate the parameters used to develop a computer representation of the user's point of view inside the Virtual Environment.

The next chapter will finally analyse a number of techniques commonly adopted to interact with VR systems. The last sections of the chapter will specifically focus on issues related to navigation, manipulation and system control.

4 Interface as a Metaphor to Interact with Virtual Environments

4.1 Introduction

The previous chapter has shown the growing interest within the design community in highly interactive tools that do not force the user to think of the underlying system architecture. This chapter will focus on the interface that is the foundation of these systems and it will provide a theoretical background to the discussion of the following chapters.

The last decade has been characterised by a constantly increasing interest in the emerging field of Human-Computer Interfaces (HCI). As Mountford (1990, p. 439) noticed, user-computer interfaces "are becoming the most distinguishing feature of computer products".

Bowman et al. (2001, p. 98) note that, although the development of haptic, tactile, or auditory devices is broadening the concept of a human-computer interface, "a distinction must be made between input devices and interaction techniques [...] In general, many different interaction techniques can be mapped onto a given input device". Following from this, the focus of this chapter will be on interaction techniques rather than on technical devices and therefore it will not consider either the myriad of devices available on the market or their direct applications in Virtual Environments.

This chapter, instead, will present an overview of the theories behind the study of user interfaces for Virtual Reality systems and provide examples of past and current developments. Specifically, the focus will be on the interface as metaphor in interaction with the environment. As a consequence, fields like *tangible interactions* (Anderson et al., 2000) or *multimodal interfaces* (Oviatt et al., 2000; Wu et al., 1999; Johnston et al., 1997; Cohen et al., 1998) will not be discussed.

The first paragraphs will introduce the general issues related to human-computer interaction, the following ones will give an overview of the spatial

implications that influence the development of 3D-interfaces, and finally a classification of the interaction techniques will be provided.

4.2 Human-Computer Interfaces

As Pimentel et al. (1995) report, in 1968, almost concurrently with the development of Ivan Sutherland's Sketchpad system (Sutherland, 1963), Douglas C. Engelbart (Bootstrap Alliance, 2002), at the Fall Joint Computer Conference in San Francisco showed how to interact with a computer using a pointing device. This breakthrough achievement was to radically change the way users interface with computers and it paved the way for the field of Human-Computer Interaction.

In the scientific literature there are a number of definitions for the idiom *user interface*. One of the most comprehensive is proposed by Barrilleaux (2001, p. 6) who writes that "the *user interface* is everything having to do with the computer that the user can touch, see, and hear (and someday taste and smell)". From the programmer's point of view, Barrilleaux (2001, p. 8) considers instead the user interface as the "spirit" of the developer: "the designer is in the system, not in body but in spirit, to give the user guidance and feedback about how to use the application and its data".

Interestingly Barrilleaux chooses the idiom *user interface* although, as Grudin (1990a) notes, the term originates from the more comprehensive *user-computer interface*. According to Grudin (1990a), the misuse of the term has generated a great level of confusion and it is often the source of misinterpretation. The author (Grudin, 1990a), sarcastically emphasising the consequence of such a shift, wonders whether we should refer to *the user interface* of a computer or *the computer interface* of a user. In fact, as the author notes, the idiom *user interface* has ironically become a technology-centred term "where the computer is assumed, the user must be specified" (Grudin, 1990b, p. 261).

Consequently, as the author suggests, the more comprehensive idiom *Human-Computer Interface* (HCI) should be adopted while the general term *interface* should only be used to refer to the technical implementation of a human-computer interface.

According to Grudin (1990a, p. 271) the term *interface* is in fact only "the segment of the software program that handled dialogue with users".

This thesis will adopt Grudin's rigorous classification and in addition, due to the focus on three-dimensional systems, it will adopt the term *3D Human-Computer Interface* (3D-HCI) and its implementation known in computer literature as *3D-interface*.

4.3 3D-Interfaces

In the case of 3D-HCIs the dialogue between user and machine, according to Barrilleaux (2001, p.6), happens over three layers: the *primal*, the *virtual* and the *analytical*.

The *primal level* resides at a "subconscious level" where stimuli are interpreted and commands are sent by the human brain to the computer. As the author points out (Barrilleaux, 2001, p. 6) this level "deals with matters such as hand-eye coordination, stimulus-response, and reflexes". The user exploits his/her senses to perceive what the computer is communicating through its devices and he reacts accordingly without the need to consciously think about the action. It is this primal level, for instance, that is responsible, for the user's reaction to steer right on auto racing game when a right hand curve is approached.

The *virtual level* instead deals with the communication between machine and user about information regarding the virtual world. This communication is based on some form of abstraction where the machine shows the state of the system through some parameters. The user, in turn, commands the computer over which parameter needs to be changed and how. This level is of crucial importance for the success of the application since the computer system is "working hard to maintain the illusion that the data is real and the user can really change it" (Barrilleaux, 2001, p. 7). In the previous example of the auto racing game, the user is able to see the state of the system through a dashboard and he/she can modify its state by changing gear, slowing down, etc.

The *analytical level* is logically placed between the first two. It deals with the specific details of the communication between the user and the machine. The programmer is entirely responsible for this level since he/she has to decide how those modifications ordered by the user have to be interpreted by the computer and consequently how the system has to respond.

The development of effective 3D-interfaces for Virtual Reality systems is a crucial factor for the success of a VR application and it represents a challenging and controversial issue widely debated in research literature. In fact, although its development should positively benefit from the three-dimensional nature of VR systems being able to reflect real life experiences at the same time it has to consider the fundamental differences between the virtual and the real world.

As a consequence, the designer of the system has to choose whether to leave the user complete freedom in their interactions with the virtual world or to constrain the user's power according to certain rules. The adoption of constraints can deeply influence the behaviour of the system. For instance, on the one hand it would not be necessarily positive to leave the user completely free to create a chair that is a kilometre tall, but on the other hand, forcing the user to respect too many physical laws could minimise the advantages of using VR.

Consequently Barrilleaux (2001) proposes a classification of VR systems by the number of constraints they implement: the *simulation*, the *arbitrary* and the *mixed* approach. In the *simulation* metaphor the virtual world tries to be a close replica of the real world with all its rules. Contrary to this, the *arbitrary* approach leaves the user free to interact with the system. Practically, excluding military and flight simulators, most VR applications follow a *mixed* approach where the boundaries and constraints are placed according to the type of user and the task to be accomplished.

4.4 The Geometrical Nature of 3D-Interfaces

From the previous paragraph it is evident that the complexity of the issues involved in the development of 3D-interfaces requires a comprehensive outline of the specific theories and techniques developed until now. The following paragraphs will

first introduce the geometrical implications behind 3D-interfaces and then provide an overview of present techniques.

4.4.1 The "Spaces" of 3D-Interfaces

Although a VR simulation can be n-dimensional, for its visualisation it eventually requires a cast to a three-dimensional reference. Furthermore, since any rendered data must be shown on a device that is two dimensional, most likely a screen or a set of screens, a further cast is required.

These casts introduce several layers of abstraction and therefore a number of definitions are necessary to unequivocally interpret the issues related to virtual spaces. More precisely, the complexity behind the visualisation of spatial information requires a number of definitions that univocally describe the different coordinate systems that are used to represent different aspects of three-dimensional computer representations. Regarding these Barrilleaux (2001) provided a comprehensive analysis of the geometrical transformations that are necessary for this shift from the *real* to the *virtual world*.

According to the author the *real world space* is where the user lives and it is the three-dimensional container of a two-dimensional sub-space: the *2D screen space*.

The *2D screen space*, also called by Sowizral et al. (2000) *Image Plate Coordinate System,* is a portal to the virtual world that presents the view of that environment to the user. It can be considered the local coordinate system of the screen, a sub-space of the real world space. The *2D screen space* can be fixed in position in relation to the real world as in the case of a computer monitor, CAVE or Reality Center™ or it can be physically moved within it, as in the case of the HMD.

The *2D Screen Space* is not a separate space from the *real world space*: it is only a different reference system where every rendered pixel is conveniently localised through a couple of x-y coordinates. More precisely the relationship between the *real world space* and *2D screen space* is always described through a matrix containing the information necessary to re-map one system onto the other. The fields of this matrix are normally constant with the exception of HMD systems

where they are provided by the tracking mechanism. Obviously in the case of multiple screen systems there will be as many *2D screen spaces* as the number of physical screens installed.

As well as 2D *screen space* it is possible to define *2D display space* as the place in the virtual world where the virtual world is rendered. In fact if the screen is thought of as the plane where the virtual world is projected "the display can be though of as living in the [virtual] world at the position of the view and facing in the direction of the view" (Barrilleaux, 2001. p. 26). In other words the 2D *screen space* and the *2D display space* are placed in the same position but are relative to two different references: the *real world* and the *virtual world* coordinate systems. Any object rendered in the *2D display space* will be anchored to the point of view of the user and therefore its position will continuously change in relation to the *3D virtual world space*, as the user moves.

Ultimately the *3D virtual world space* represents the space where the computer representation exists and where the objects visualised through the screen are located.

4.4.2 The User's Point of View

The classification of the different spaces provides a general framework to help understand the different techniques that are used to interact with VR systems. However, to fully analyse the geometrical features involved in the development of 3D-interfaces, an overview of the user's point of view must be implemented. In fact the analytical representation of the user's view plays a major role within the overall 3D human-computer interface since, as Barrilleaux (2001, p.24) notes, it ultimately "defines how the user sees the world".

Barrilleaux (2001) proposes an analysis of the viewing architecture based on two different aspects: the internal and the external features.

The external geometry of the view, according to Barrilleaux (2001, p.31), "involves geometrical relationships outside the view space itself, specifically the position and orientation of the space as it exists in the world space". A number of parameters influence its state (See Figure 4.1) such as:

- The LAP (look-at point): the point observed by the user.
- The LFP (look-from point): the user's position in the virtual world.
- The LFO (look-from offset): "offset of the LFP from the LAP relative to the view space" (Barrilleaux, 2001, p. 32).
- The LAD (look-at direction): the direction the user is looking at, which consists of two components, the LAD-DV (LAD direction vector) and LAD-UV (LAD up vector). The first component represents the direction the user is looking at while the second defines the direction in the virtual world representing the "up" for the user.

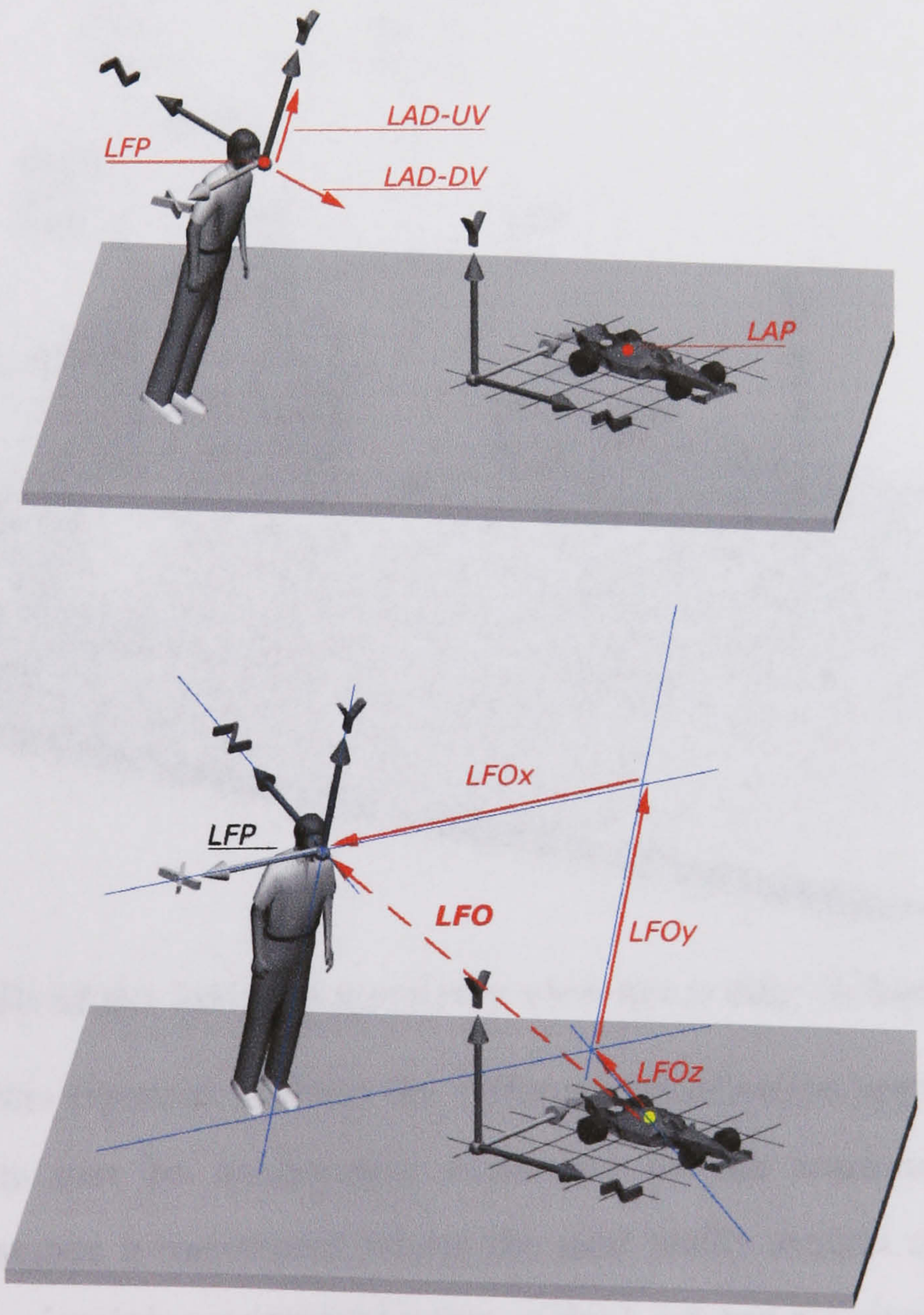


Figure 4.1: Details of the external geometry view according to Barrilleaux (2001)

The internal geometry of the view instead, deals with the projection of the space. The parameters describing its state are (See Figure 4.2):

- The DVO (display-view offset): the translation necessary to move the centre of a display area from the centre of the screen. This value becomes zero if the application is running at full screen.
- DS (display size)
- The FOV (field of view)
- The VSF (view scale factor) and DSF (display scale factor) are the two scale factors used to project the content of the virtual world respectively onto the *2D display* space and *2D screen space*.

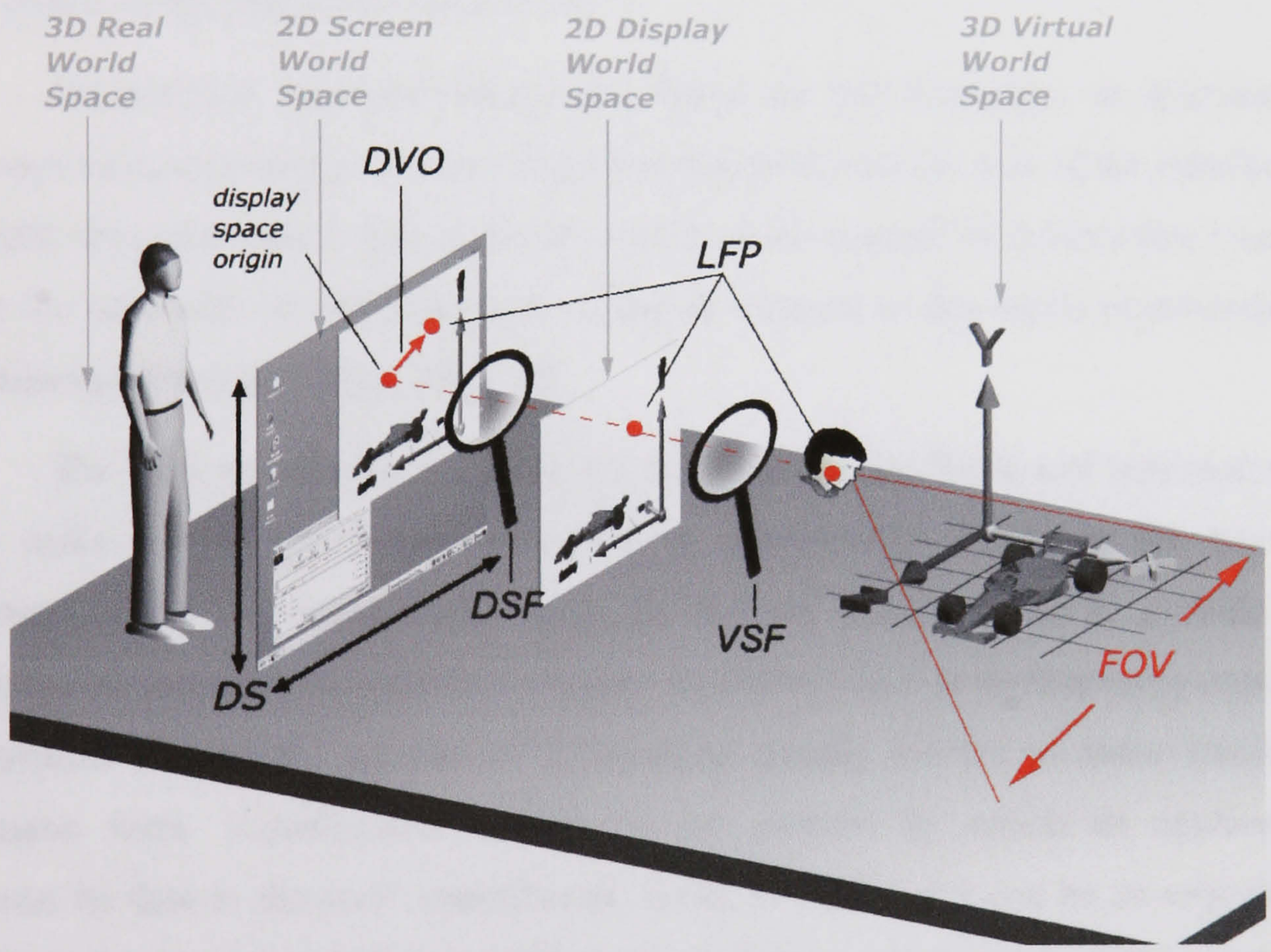


Figure 4.2: Details of the internal geometry view according to Barrilleaux (2001)

Through this rigorous analysis the different visualisation approaches plus the user's movements can be analytically expressed by the manipulation of these variables. For instance a movement where the user walks around an object can be implemented by maintaining a constant value of the LAP and moving the LFP.

4.5 Three Dimensional Interactions and Techniques

On the basis of the theoretical background set out in the previous paragraphs it is possible to understand how different interaction metaphors can be implemented. The implementation of a 3D-interface is of crucial importance to the entire VR system. As Bowman et al. (2001, p. 96) note, "great care must go into the design of user interfaces and interaction techniques for 3-D applications" since they determine its overall behaviour and ultimately contribute to its success.

The implementation must be targeted to the specific application since, as Barrilleaux (2001, p. 39) states, "there are no right or wrong answers in choosing one technique or variation over another. [...] Application design must be based on an understanding of what the user needs to accomplish, both within a given task and in the context of the application as a whole".

3D-interface implementations are based on the exchange of information between machine and user. From a high level point of view the role of the interface is twofold: from one side it has to decode one or more streams of information coming from the user and, on the other side, it has to respond to the user's commands by visualising data or providing feedback.

The flow of information from the machine via feedback and rendered data may make use of the same media, but it substantially differs in the type of information the two components carry. In fact the feedback has to provide the information necessary to confirm or support an action and it is consequently required to transmit a restricted amount of information usually limited to some iconic or idiomatic form. Visualisation instead "is the process by which an application presents its data to the user" (Barrilleaux, 2000, p. 100) and it can be an articulated combination of several data with different meanings and it can potentially provide a much broader range of information.

The flow of information from the user to the machine is more articulated since it requires the machine to interpret the wish of the user according to the state of the system. There is often insufficient rigour shown in trying to describe this

(Hinckley et al., 1994) and in the definition of the various actions happening and the consequent computer-user interfaces techniques adopted.

Barrilleaux (2001, p. 47) proposes an approach that classifies the interaction in terms of the function of the "control *personae*, which mirrors those used for participants in everyday speech: first person, second person, and third person". In a *first person* system the user directly controls the content of the virtual world. In the *second person* approach the user acts on the system through the control and manipulation of objects in the environment. Finally in the *third person* example the user interacts with the environment through the use of buttons or interfaces external to the environment.

Furthermore Barrilleaux (2001) and Bowman et al. (2001) present two similar classifications of the function of the actions made by the user. According to both authors interaction happens through *navigation*, *manipulation* and *access* or, as Bowman et al. (2001) refers to it, *system control.* The first "moves the view", the second "moves the data" and the third "gets the data into and out of the virtual world" (Barrilleaux, 2001, p. 8).

4.5.1 Navigation

In most VR applications navigation is the most prevalent interaction with the system. It "presents challenges such as supporting spatial awareness, providing efficient and comfortable movement between distant locations, and making navigation lightweight so that users can focus on more important tasks" (Bowman et al., 2001, p. 98).

As Bowman et al. (2001, p. 98) note, navigation in the real world is often an intuitive action, "conceptually a simple task" where the user does not think of how to go in one direction but rather focuses on where to go.

Navigation takes place, according to the authors, on two different levels: *travel* and *wayfinding*. The first is the motor act while the second is its cognitive complementary "process of defining a path through an environment, thereby using and acquiring spatial knowledge to build up a cognitive map of an environment" (Bowman et al., 2001, p. 100).

The study of the cognitive and perceptive issues of real or virtual spaces has already been studied by a number of authors (Reeves et al., 2000; Weber, 1995; Hubona et al., 1999; Durand, 2002) and as has already been mentioned, it is outside the scope of this thesis. Nevertheless it is acknowledged that although we live in a 3D-world, the human experience is fundamentally two-dimensional since most of the movements take place on a plane. In fact quite often the real-world navigation experience mainly relies on forward movement and rotation about the vertical axis.

This leaves the designer of the interface, as Bolas (1994, p. 51) notices, with the serious question of "how to allow the user to control such motion effectively". This issue becomes crucial when using two degree-of-freedom devices. In these circumstances, as stressed by Hanson et al. (1997, p. 175), it is important to restrain the users to "a constrained subspace" rather than allowing them full control over the movements.

But the issue of constraining motion has important consequences in any VR system. Subsequently several general techniques, which have been developed to tackle this issue in different contexts will be stressed in the classification of the navigation techniques provided in the following pages.

4.5.1.1 Classification by Personae

Barrilleaux (2001) proposes a categorisation of navigation according to the *personae* or what Bowman et al. (2001) calls the *frame of reference*.

In *first person* (Barrilleaux, 2001) or *egocentric* (Bowman et al., 2001) navigation the user controls the view directly by pointing to a certain direction and moving towards it. As Barrilleaux (2001) stresses, a special case of first person navigation takes place when the user is in *first person* navigation mode using *third person* controls "such as a virtual steering wheel and throttle control in a display dashboard. The difference [...] is that the controls are part of your vehicle's metaphor. They are an integral part of your first-person experience, the same as would be a real steering wheel in a real car" (Barrilleaux, 2001, p. 128).

Despite the apparently close relationship with the natural experience, researches have shown how the *first person* or "point to fly" metaphor (Bolas, 1994)

presents severe pitfalls. Krueger also agrees that this approach is counter-intuitive because "you point and fly, instead of walking around" as in real life (Krueger in Morgan, 1994, p. 173).

In experiments reported on by Bolas (1994) a number of users, wearing an HMD and using a Virtual Glove to point towards a direction, were asked to fly around a track as fast as possible. The experience highlighted two critical issues, which arose when approaching corners on the track.

First of all the daily experience of driving a car suggested to the users that as the turn was made the entire reference system, i.e. the car, would rotate. Unfortunately in the point-to-fly metaphor the user himself is the reference system and therefore he/she must actually turn his/her body according to the trajectory he/she wants to follow. Secondly, some users were physically leaning into turns as if riding a motorbike, but keeping on pointing the system onwards and "the farther off course they became, the farther they would lean while continuing to point straight ahead" (Bolas, 1994, p. 52).

The author eventually proved that placing an object, i.e. a small airplane, in front of the users would suggest them how far to rotate his body when going around a corner. Moreover this approach proved successful when it was decided to support the person leaning. In fact the airplane helped the user understand that the more he/she would lean the more the airplane would bank and therefore the faster the rotation would be.

In the *second-person* (Barrilleaux, 2001) navigation mode the user is not directly present in the world but they interact with it through controls present within the scene. This approach might at first seem less natural since our experience in the real world is mainly first-person navigation, but it can be useful in certain circumstances, for instance in the case of a camera used to navigate around an object or systems that take one towards an object (Igarashi et al., 1998). Another approach is to follow a representation of the user in the world and use that to navigate. For instance in the case of a simulated car it might be possible to drive through the world by acting on the model of the car. The difference with the example of *first-person navigation* is that in this case the user would not have the feeling of being inside the

space manoeuvring certain mechanisms to drive (steering wheel, gearbox etc.) but they will be outside the space manipulating a model of a car to drive around.

In the *third person* (Barrilleaux, 2001) or *exocentric* (Bowman et al., 2001) case, navigation is achieved through a control that is completely separated from the virtual space, outside the scene. "A good example of third-person navigation is moving a cursor on a map, which correspondingly moves the user's view through the virtual world" (Barrilleaux, 2001, p. 129).

This approach, which could seem to be the least close to the real experience, has actually proved to be particularly powerful under certain circumstances. Haik et al. (2002) have carried out a comparative test to evaluate the effectiveness of the three different navigation modes, first, second and third person modes respectively, when using two degree-of-freedom devices with desktop-VR systems. This test, which replicates the common conditions of a browser being used to navigate 3D-spaces, represents a challenging example where "the user's concentration could be distracted by problems experienced with the mouse" (Haik et al., 2002, p. 59).

The outcome of the experiment surprisingly showed that the use of a navigation map to browse the space was the easiest tool since it "enabled navigation by single mouse clicks it made the mouse-usage very simple and prevented the users from experiencing difficulties" (Haik et al., 2002, p. 63).

4.5.1.2 Classification by Action and Metaphor

Bowman et al. (2001) also propose making a shift in the focus of the analysis of the navigation techniques from the user to the action or to the metaphor adopted. According to the authors navigation techniques can be divided into three action categories: *exploration*, *search* and *manoeuvring*.

Exploration is for the authors (Bowman et al., 2001, p. 98) "navigation with no explicit target" where the user is free to study the space in which he/she is immersed. *Search* is where navigation aimed to approach one specific place in the environment. Finally *manoeuvring* is the set of actions "characterized by short-range, high precision movements that are used to place the viewpoint at a more advantageous location for performing a particular task" (Bowman et al., 2001, p. 98).

In terms of metaphor, the authors propose categorisation into five different groups.

1. *Physical movement*: where the user directly controls the motion of his/her body, which may correspond to Barrilleaux's (2001) *first person* navigation.

2. *Manual viewpoint manipulation*: where the user can manipulate their point of view through controls, which corresponds to Barrilleaux's (2001) *second* and *third person* mode.

3. *Steering*: where the user is continuously specifying the direction of movement not necessarily through the use of a steering wheel, but perhaps through an HMD gazing at a point.

4. *Target-based travel*: where the user specifies the final location and the system moves the user through incremental or instantaneous movement. For instance this can be achieved through selection from several viewpoints in a case where the user wishes to move to a specific location of the world. Other implementations include for example in Mackinlay et al.'s (1990, p. 171) *Point Of Interest* (POI) technique, where the user "indicates a point of interest (target) on a 3D-object and uses the distance to this target to move the viewpoint logarithmically, by moving the same relative percentage of distance to the target on every animation cycle. The result is rapid motion over distance that slows as the viewpoint approaches the target object".

5. *Route planning*: where the user decides the path to be taken throughout the environment and the system in turn handles the movement. This can be achieved through a traditional 2D-map or through more sophisticated techniques like the "Path Drawing" approach (Igarashi et al., 1998) illustrated in Figure 4.3, where the user can draw a walkthrough path in a virtual space and the point of view follows the path automatically. As the authors note "using this technique, the user can specify not only the goal position, but also the route to take and the camera direction" (Igarashi et al., 1998, p. 173).

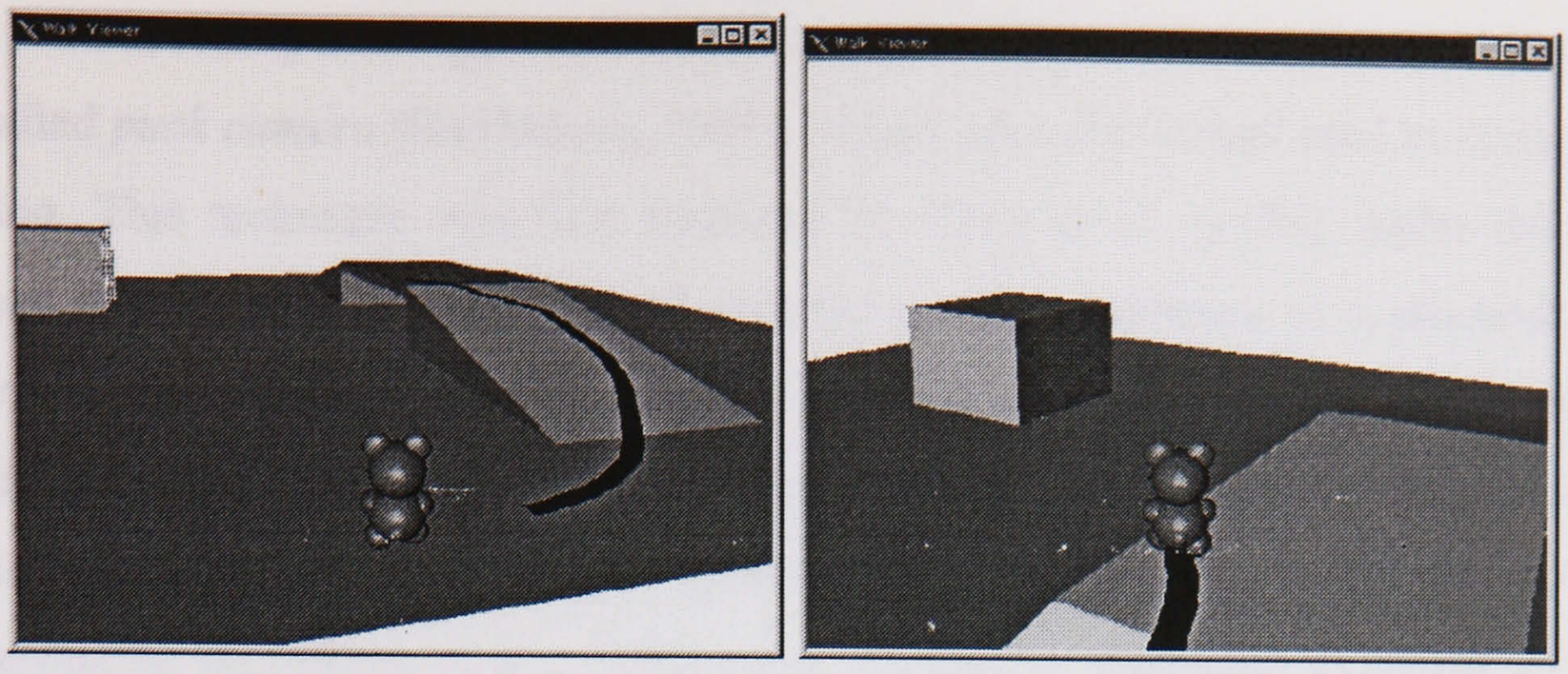


Figure 4.3: The "Path Drawing" technique (Igarashi et al., 1998, p. 173)

4.5.1.3 Classification by Camera Technique

When the user's actions require an object to be studied, moving them around the space can be quite ineffective. In this circumstance better control can be achieved through the manipulation of a camera, the latter being the metaphor for the user's point of view.

In the *orbit camera* technique the LAP (look-at point) is fixed but the user can change the LFP (look-from point) and the LAD (look-at direction) (See Section 4.4.2) and as a result the user is free to *orbit* the object at a fixed distance. It is possible to introduce a further constraint by restricting the user's inclination to a fixed value. This way the user remains free to rotate and tilt the camera making their interaction even simpler. If a greater degree of freedom is required the user could also be allowed to zoom.

The orbit camera technique is now implemented by most desktop-VR systems but experience has proven that other hardware configurations can benefit from this approach, as in the example shown by Koller et al. (1996) where HMD was used.

The main drawbacks of this approach are firstly that "in a densely populated virtual world, orbital viewing is vulnerable to occlusion of the point of interest" (Koller et al., 1996, p. 82) and secondly, experience shows that it is cause of more frequent symptoms of simulator sickness occurring (Koller et al., 1996).

A second-person approach using the orbit camera is improved by the to the so-called *puck camera* (Barrilleaux, 2001), named after the widget used to control its motion. This technique was first proposed by Chen et al. (1988), under the title *virtual sphere*, with the intention of simulating "the mechanics of a physical 3-D trackball that can freely rotate about any arbitrary axis in 3D-space" (Chen et al., 1998, p. 123). The same interface, but with a more elegant mathematical implementation, was also proposed by Shoemake's (Shoemake, 1992) ARCBALL. Experiments (Hinckley et al., 1997) have shown no statistical evidence of any improvement over the original technique introduced by Chen et al. (1998).

Zeleznik et al. (1999) proposed an innovative implementation, called UniCam, where the user can accomplish different camera tasks through single-button actions, including translation, orbiting, animated navigation, zooming, saving and restoring. This is achieved through a complex procedure that interprets the position of the pointer and the direction of the gesture into effective combined actions.

As Barrilleaux (2001) notes the presence of the control in the space makes it easy to switch between manipulation and navigation. At the same time however, the constant presence of the control puck in the scene may represent a major drawback.

Finally Barrilleaux (2001, p. 137) calls *Pinocchio Camera* a "third-person camera that offers puck-like navigation control" where the user can move the camera through a controller placed outside the virtual world.

4.5.2 Manipulation

In the real world the difference between navigation and manipulation can sometimes be very subtle whilst with computers it can become extremely clear: "in navigation, the user or system moves a view object, making a new portion of the virtual world visible. In manipulation, the user or system moves a data object, modifying the contents of the world that the user sees" (Barrilleaux, 2000, p. 125).

Direct manipulation is an integral part of our common physical experience where all our senses are involved: "people rely on sight and touch to help them manipulate objects" (Barrilleaux, 2001, p. 148). Therefore the role of the

manipulation interface becomes crucial for the overall quality of the entire VR system.

It requires a successful integration between the control system, feedback mechanism and visualisation flow and the development of smart strategies to supply the necessary clues to our brain that are required to recreate the experience of having something real to manipulate.

4.5.2.1 Classification by Personae

As in the case of navigation Barrilleaux proposes a division of manipulation techniques into *personae*.

In a *first-person* case "the user is essentially controlling himself/herself, the first person" (Barrilleaux, 2001, p. 150) and he/she can notice the effects of their manipulation on his/her representation. This approach is suitable if the object is a vehicle and the user has the feeling of being inside it. In some circumstances though, as Barrilleaux (2001) noted, this approach may lead to the awkward feeling of being an object, which is being dragged throughout the environment.

In the *second-person* mode, which is also called by Hinckley et al. (1994) the *scene-in-hand metaphor*, the user manipulates an object in the world, rather than manipulating his representation. Therefore this approach is perfectly suited for intuitive manipulation of objects in the scene. As Barrilleaux (2001, p. 152) observes "the advantages of second-person control are that the interaction is quite familiar to users – reach out and move objects – and that the target can be seen in the context of its surroundings, something that is missing in the first-person control".

In the case of the *third-person* technique the user manipulates a control device rather than the object itself. This device can be a knob or a slider that is placed inside or outside the virtual world. Although at first this approach might not seem intuitive it has the advantage that the action is made explicit and consequently it can be unequivocally performed, for example using a slider to lift objects. At the same time though, it must be made clear which action each controller performs and this can contribute to the entire application feeling less intuitive.

4.5.2.2 Selection

Every form of manipulation starts with the act of selecting. In VR the simplest form of selection is obtained through the *ray-casting* (Bowman et al., 2001). The object selected is the closest one encountered by a ray cast into the scene passing from the point of view of the user and from the pointer on the screen. The technique, also called the *laser pointer* metaphor (Forsberg et al., 1996), has a number of variations including the *spotlight* or *cone-based* technique, the *cylinder-based* selection or the more complex *aperture-based-selection* technique proposed by Forsberg et al. (1996). All differ in the selecting volumes used to pick the object.

Where a 3D-tracked device is used the physical set-up is often represented through a virtual hand or a virtual pen that can be used to select objects. The main limit of this approach is that the user cannot select objects outside his/her arm's length. To overcome this issue and to provide the user with a more powerful selection capability than they have in real life several authors (Poupyrev et al., 1996; Song et al., 2000) have proposed also applying a non-linear mapping technique of the user's hand movements to enable them to grasp objects far from their reach.

4.5.2.3 Mapping

The problem of mapping the user's movements into 3D computer actions is not only relative to the act of selection and this issue becomes crucial when the user needs to interact with objects. More specifically the transformation of spatial commands into a coordinate system, called by Barrilleaux (2001) *mapping*, is fundamental to link the *source space*, that is the reference used by the control input device, to the *target space*, that is the reference relative to which the movements of the object to be manipulated has to be interpreted (Barrilleaux, 2001). For instance when the user wants to move an object over a certain distance it is essential to know whether the object will move according to an absolute or to a local coordinate system.

As Barrilleaux (2001, p. 59) notes *direct mapping*, the direct wiring between the user input device and the object, is "the simplest form of coordinate mapping". Unfortunately this technique is of limited use in most Virtual Reality applications.

Through direct mapping in fact, a user's movement along the x-axis for instance, would cause a movement of the object according to the wiring mechanism, causing the object to move along the x-axis too, independently from the point of view of the user.

But in VR, in most cases users want to move an object relative to their position or to the local reference of the object itself. For instance a user may want to move a chair towards his/her left or right or slide it along one of its sides. To do so other forms of mapping must be taken into account according to the specific needs of the user and the task to be accomplished.

4.5.3 System Control

According to Bowman et al. (2001, p. 102) *system control* "refers to a task in which a command is applied to change either the state of the system or the mode of interaction". This action, which is also called *access* by Barrilleaux (2001), takes place when the user needs to deal with the data in the scene rather than with only geometries.

The information to be accessed can be of two kinds: data regarding the configuration of objects in the scene and data external to the virtual world. The first, for instance could include an object hierarchy structure or some other form of configuration of the environment whist the second is usually more abstract and does not take advantage of the visualisation capability of the 3D-environment. Usually the user can access this data through palettes, graphs or tables.

Access can take place through traditional WIMP-based (Window, Icon, Menu, Pointer) interfaces or through 3D-Human Computer Interfaces. However, there has been an animated discussion within the research community on whether the use of 3D-HCIs would bring any benefit to VR at all.

Some authors stress the "innately intuitive" (Barrilleaux, 2001, p. 5) nature of 3D-HCIs, others argue that its usability remains uncertain. According to the latter group the use of 3D-*widgets,* "an encapsulation of geometry and behaviour used to control or display information about application objects" (Conner et al., 1992, p. 183), presents a number of serious issues in terms of usability. As Tromp et al.

(1997, p. 41) note "an immersed user with a tracked hand may find it relatively easy to manipulate 3D-widgets, whereas a desktop user might find it difficult since a lot of navigational effort would have to be expended in order to reach the correct position and orientation to manipulate the menus".

As several authors note (Bowman et al., 2001) in some circumstances 2D Human-Computer Interfaces can be more effective than 3D-HCIs. The studies carried out by Cockburn et al. (2002) have demonstrated that access time might be significantly slower using 3D-HCIs compared to more conventional 2D-HCIs as the portal to the virtual world. In fact 3D-HCIs can create difficulties since an object can be more difficult to reach in a three-dimensional space.

Therefore as Bowman et al. (2001) report, the majority of the authors have developed hybrid approaches where the interface is placed on a 2D-surface in the space. In fact the metaphor followed by most interfaces for modern 3D-applications extends the approach followed by traditional GUIs, where menus, windows and buttons have been turned into their three-dimensional counterparts. As Gentner et al. (1990, p. 281) noted "within a few years after the beginning of a new technology, we often see attempts to broaden the market by building interfaces that simulate the previous technology. The goal is to reduce or eliminate the required learning, making the technology widely available to people who are unwilling to adapt to a new system".

This is what has happened in recent years to 3D-HCIs where, as Conner et al. (1992, p. 183) note "the significant difficulties of 3D input and display have led research in virtual worlds to concentrate far more on the development of new devices and device-handling techniques than on higher-level techniques for 3D-interaction".

4.6 Conclusions

This chapter concluded the overview of current research in the field of Human Computer Interfaces and Virtual Reality. It presented the theory behind HCIs, illustrating the most important geometrical issues and providing a number of definitions.

More precisely the chapter presented a description of the coordinate systems used to represent the different aspects of three-dimensional computer generated worlds.

The chapter also presented a taxonomy of the interaction techniques used in *navigation*, *manipulation* and *access*. The interaction techniques were presented from different points of view. The classification considered the type of *control personae*, as suggested by Barrilleaux (2001), as well as the type of metaphor being implemented.

The next chapter will introduce the research framework, showing the general structure of the research through an overview of the features of each module. The chapter will also justify technical solutions and will give the basis for a detailed description of the prototype called JCAD-VR (Java Collaborative Architectural Design tool in Virtual Reality), which will be fully reported in the following chapters.

5 Development of a Framework: JCAD-VR (Java™ Collaborative Architectural Design tool in Virtual Reality)

5.1 Introduction

The previous chapters have provided an overview of current research in the field and have supplied the theoretical background to the following chapters. This chapter will introduce the proposed research framework and show the development of the working prototype called JCAD-VR: Java Collaborative Architectural Design tool in Virtual Reality.

The aim of this chapter is to show the overall structure of the research framework and to provide a broad picture of the entire system. Technical details along with theoretical and practical choices will then be discussed in depth in the following two chapters.

The first part of this chapter provides reasoned justification for the way the framework was developed, it introduces the system and it outlines the technical choices. The second part provides a general introduction to the framework, to its features and to its architecture as well as a description of the number of modules that comprise JCAD-VR. The end of the chapter summarises the scope of the research and gives consideration to the technical choices arising from the project.

5.2 VR as a Collaborative Design Tool for Conceptual Design

Conceptual design at the initial creative phase of the design process is a complex activity characterised by the use of intuition rather than mathematical formulae. During this stage designers try to give concrete form to their abstracted models through mental simulation. Designers traditionally make sketches as a way of doing this, exploiting their abstract and ambiguous nature to explore new design

solutions. As Shukur (2000) notes, sketches on paper are a natural and intimate medium that allows the down/uploading of complex information to and from the designer's brain, they are also easy to correct or change, portable, extremely flexible and they can store information conveniently and keep track of ideas and solutions.

During this complex undefined phase computer-based systems are not usually employed due to the discrepancy between the designers' conceptual models and the limits of present-day Computer Aided Design (CAD) interfaces. These bind the designers' freedom through their imposition of a formal and constrained graphic language rather than them supporting uncertainty, flexibility and dynamic manipulation. Consequently, as Turner et al. (1999) note, this means designers have a large conceptual gap to bridge between their abstract and possibly vague mental representations in sketches and the formally defined shapes created by current CAD systems.

In addition, the CAD industry often trades the qualities of having advanced geometrical control for the qualities of interactivity and usability, therefore delivering complex interfaces that often lack in flexibility and user-friendliness. The effect of this is that although CAD systems can feature a great number of functions allowing the modelling of complicate shapes, they cannot be used at the very early stage of product design where pencil and paper are still the most effective. Commercial CAD systems are designed to fulfil the needs of a wide range of engineering fields and to support the construction process, they have not been designed to promote the creative acts typical of the initial stages of design.

As a consequence, the early design phase is disassociated from the product definition stage that uses CAD/CAAD because the designers are forced to adopt traditional tools like paper and pencils. This creates an amount of inefficiency within the design cycle and consequently higher production costs. There is an evident need for innovative computer-based tools specifically tailored to support creativity within the design community.

In the last few years, the growing awareness of more natural, user centred form of interaction, has promoted interest in expanding the field of Human-Computer Interface (HCI). As a result, a new generation of computer-based

applications has embraced innovative technologies whose adoption in turn, has fostered original interfaces aimed at promoting user-friendliness and more interactive behaviour.

The speed at which technology is evolving is bringing the application of VR technologies within the sphere of the design professions. The increasing growth of computational resources and hardware power is facilitating a transition to the use of desktop VR applications as truly feasible tools for everyday use.

As a consequence, some design and manufacturing companies have already started to investigate how VR could be used effectively within the design process. Although VR is now a mature technology, it is seldom used in architecture during the design process it is more often merely used as a powerful presentation technique. Virtual worlds are often created using CAAD/CAD packages and deriving world representations by conversion, in this way, becomes a time consuming and therefore expensive task.

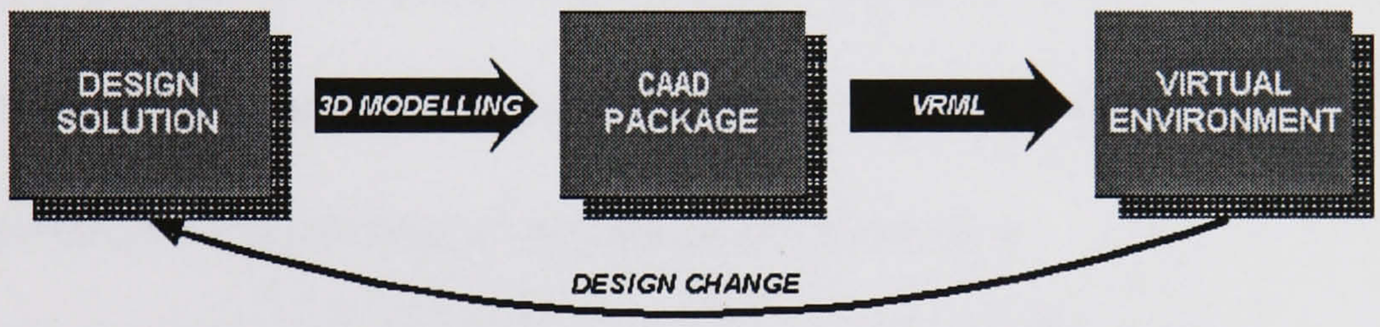


Figure 5.1: The traditional scenario showing the relationship between the 3D-modelling phase and VR

A virtual model, if it is employed, is usually only created at the end of the design stage when most key decisions have been evaluated, and mainly to impress contractors and clients (See Figure 5.1). It is therefore evident that under these circumstances the use of VR is inefficient and consequently leads to growing costs rather than fostering creativity and ultimately improving design quality.

With the benefits brought by its highly dynamic nature, the early use of VR could allow the designer to study design solutions directly through the manipulation of simple shapes in a virtual space, and hence successfully combine the creative and modelling stages (See Figure 5.2).

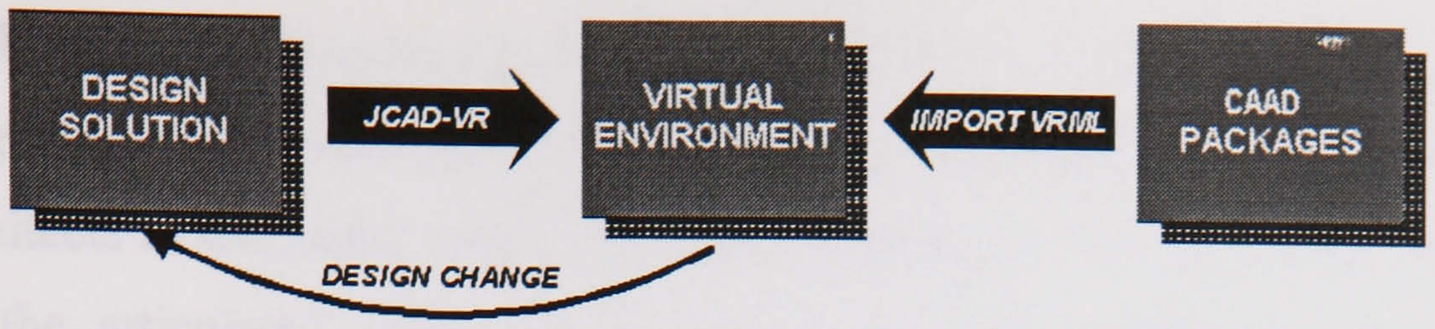


Figure 5.2: The proposed scenario

It is also highly predictable that in the near future VR will become the interface for the next generation of Computer Aided Design (CAD) applications described in research literature as Virtual Reality Aided Design (VRAD) systems. VR-based systems have the potential to become highly effective tools letting the user explore design solutions in a more intuitive and natural way. The use of VRAD tools has a number of potential advantages over traditional CAAD systems:

- It provides real-time interactivity
- It involves the designer's imagination to a greater degree
- It provides quick visualisation of mathematical entities
- It enhances creativity in the conceptualisation phase (Sener et al., 2002)
- It promotes a more natural approach to computers
- It helps designers make the transition to digital format earlier
- It has the potential to shorten the time taken to manufacture physical models through rapid prototyping processes (Ucelli et al., 2000)
- Ultimately it improves design quality by increasing the level of control on the design of the product.

Another crucial problem of current CAAD systems is their complete absence of any form of synchronous collaboration between architects within design teams. With the present CAAD technology there is little room for multi-user interaction during the creation of a 3D-model since most systems allow only asynchronous collaboration. The recent growth of network-based virtual communities however has brought a new level of complexity to the notion of virtual spaces by providing the technology for remote presence and collaborative experiences.

The research described in the ensuing chapters takes advantage of these new emerging technologies through the development of a system specifically designed to help architects in the initial stages of the design process. The JCAD-VR framework pursues the articulated vision of VR as an instrument developed to assist the participants in the design process – the professional or client body - during these early stages. It is a collaborative design tool developed to ease the initial stage of the design process providing the capability to create 3D-shapes and to share them among the users in the virtual world.

5.3 Justification of the Project

As noted by Lawson (1990), design methodologists in the past agreed on the need for iterative cycles between several phases of the design process. From studies of designers' behaviour many authors observed that there were an indefinite number of return loops from the moments when the gathering of information and structuring of the design problem took place, known as *analysis*, to the time when design solutions were generated, known as *synthesis* (See Figure 5.3).

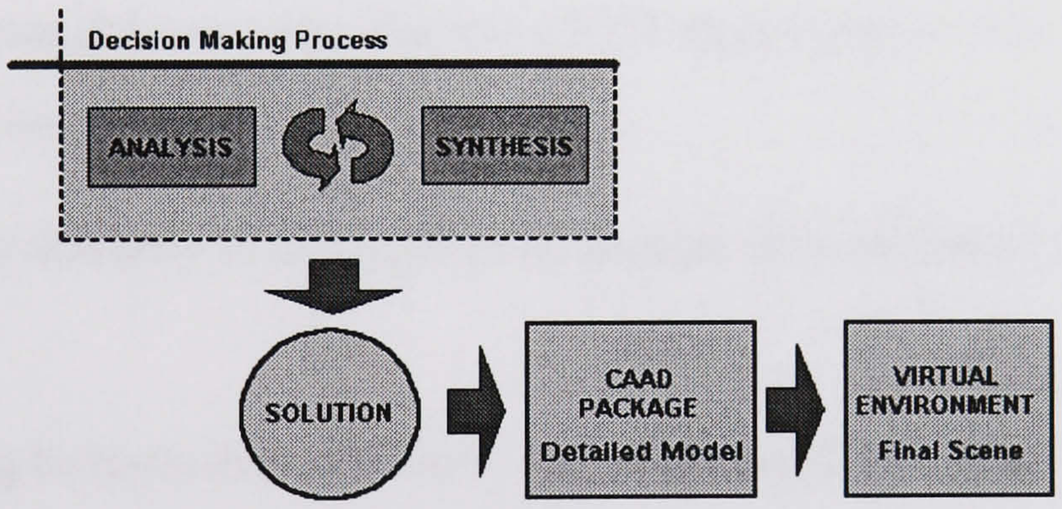


Figure 5.3: The use of CAAD and VR in the traditional decision making process

The development of JCAD-VR provides the designer with an appropriate, quick and practical response to their need for iteration in their search for design solutions. Therefore JCAD-VR provides the means for a more effective use of VR at the very beginning of the decision making process thus helping architects bridge the gap between the *analysis* and *synthesis* phases (See Figure 5.4).

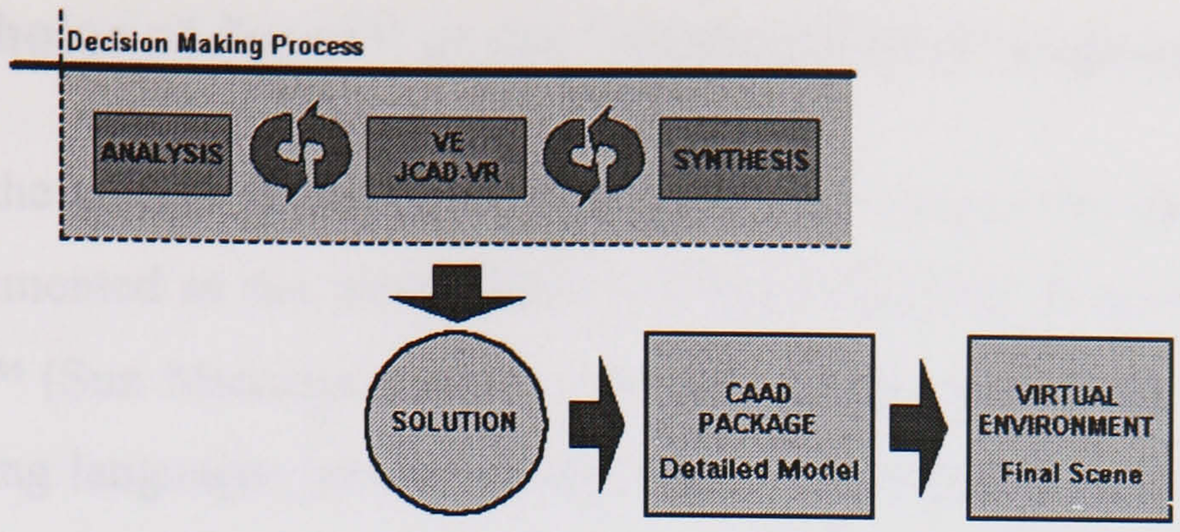


Figure 5.4: The role of JCAD-VR in the decision making process

The development of the JCAD-VR framework proposes an innovative approach to deal with some important issues:

- The use of VR in the early stages of the design process
- The traditional complexity of creating a VR environment
- The lack of a VRAD tool specifically tailored for architects' needs
- Lack of true, synchronous collaboration in commercial CAAD packages.

JCAD-VR was designed to tackle these issues through a number of characteristics and features:

- It is a system that proposes the use of VR technology at the early stages of the design process
- It is a user-friendly interactive environment that promotes a visual approach to design
- It is an application that combines the advantages of both VR and Computer Supported Cooperative Work (CSCW) systems.
- It supports communication through multimedia features
- It has an overall architecture that encourage synchronous collaboration
- It is intrinsically cross platform and hardware independent promoting platform portability.

5.3.1 The Choice of Java™ as the Programming language

Due to the amount of flexibility required by the project the entire framework has been implemented in the multi-platform Object-Oriented Programming (OOP) language Java™ (Sun Microsystems Inc., 2002a). As stated by Eckel (2000, p. 30) "all programming languages provide abstraction [...] Assembly language is a small abstraction of the underlying machine. Many so-called 'imperative' languages that followed (such as Fortran, BASIC, and C) were abstractions of assembly language".

Object-Oriented Programming (OOP) languages, such as Java™ or C++, have introduced a new level of generalization. In contrast with traditional procedural languages where the level of abstraction provided still forces the programmer "to think in terms of the structure of the computer rather than the structure of the problem" (Eckel, 2000, p. 30) OOP languages help the user develop code in a manner that is closer to the mental process that the programmer would normally follow.

As Eckel (2000, p. 31) notes, "the idea is that the program is allowed to adapt itself to the lingo of the problem by adding new types of objects, so when you read the code describing the solution, you're reading words that also express the problem. This is a more flexible and powerful language abstraction than what we've had before. Thus, OOP allows you to describe the problem in terms of the problem, rather than in terms of the computer where the solution will run". These *objects* are self-enclosing elements of the software that can be used to accomplish a certain task and whose mutually logical relationships with other objects significantly facilitates code development and efficiency through code reuse.

More specifically, Java™ is a last generation mature and complete Object-Oriented programming language that is extremely flexible and relatively easy to implement, deploy and maintain. It is intrinsically multi-platform and it supports multithreading and multiprocessing making it possible to develop programs that run on simple PCs as well as supercomputers (Sun Microsystems Inc., 2002a).

A number of features made Java™ the perfect choice for the development of a VR-based multimedia collaborative environment such as JCAD-VR, these include its:

a) Multi-platform nature

b) Network-oriented architecture

c) Easy database management

d) Availability of existing APIs dealing with:

- 3D Graphics through Java 3D™ (Sun Microsystems, 2002b)
- Audio and video support and real-time network streaming through Java Media Framework (JMF) (Sun Microsystems, 2002d)
- Networking within standard Java™ 2 Software Development Kit (Sun Microsystems, 2002c).

Although it is sometimes less efficient in terms of performance if compared with other languages, the adoption of Java™ as the programming language offered great flexibility, true scalability and last but not least, complete multi-platform support. The system demonstrated Java™ platform's independence successfully when tested in different configurations where PCs and Sgi workstations were concurrently used. The network-centric nature of Java™, and its multimedia integration and multi-processor support make it the obvious choice for the development of a real-time multimedia collaborative system.

5.4 JCAD-VR: the Framework

As described the overall JCAD-VR framework has been developed to allow dynamic interaction with the virtual environment during the early stages of the design process. The following paragraphs will provide an overview of the functions and of the architecture of both the overall framework and the working prototype. The description in this chapter provides a general picture of the system while specific details of the technical implementation will be provided in the following two chapters.

5.4.1 The Idea Behind it

JCAD-VR has been developed based on two main ideas:

1. All the users present in the virtual world have to be able to share the very same virtual environment in a "transparent fashion"
2. Instead of being based on traditional WIMP (Window, Icon, Menu, Pointer), the Human-Computer Interface is part of the virtual world itself where each element of the HCI becomes an object belonging to the 3D world and so can be manipulated like any other entity within the environment: the HCI becomes a *3D-HCI* perceived as part of the virtual world itself.

The first point results in the development of a collaborative architecture for the system. Every user that logs in can seamlessly interact with the environment, communicate with other users and manipulate objects while the system ensures that the environment in the background is fully consistent across all the users.

The second point has totally influenced the way the Human-Computer Interface has been developed. The implications of an interface that is a part of the virtual world are twofold:

1. From the technical point of view, once the interface has been designed, it becomes independent from the visualisation device used. The system can therefore be easily adapted for different devices by just rewriting the code that is handling the device. No matter whether the application is running on a simple screen, on a Reality Center™ or linked to an HMD, the interface will always be consistent and placed in the virtual world where specified by the programmer.
2. From a more theoretical point of view, the interface also becomes one of the elements of the virtual world and can therefore be treated like any other object in the virtual scene. Elements of the HCI such as panels, icons and rulers, are treated just like any other 3D entities within the VE. Each element can be moved or scaled according to the user's needs. The user interacts with the objects through elements of this interface such as arrows which are placed

to help them edit the object, and feedback is provided through the visual modification of the same object in the scene.

The result of this approach is that any object in the virtual environment can be replaced, dragged or re-scaled at the convenience of the user, regardless of whether it is something they created or whether it was created by someone else present in the virtual world. In addition the same direct approach is followed for the interface which becomes just another part of the environment contributing towards an increase in the feeling of presence.

The 3D engine renders all the changes possible in the VE: movement of avatars, video conferencing streams rendered on 3D panels, textual communication through 3D chat, changing of the interface and most importantly, the creation and modification of objects created within JCAD-VR.

5.4.2 System Features

When JCAD-VR is initiated, the user is asked for a login name to be used to communicate within the virtual world, and through an options panel they decide which server to connect to and choose the server port (See Figure 5.5). In case no value is given the application can run in stand-alone mode.

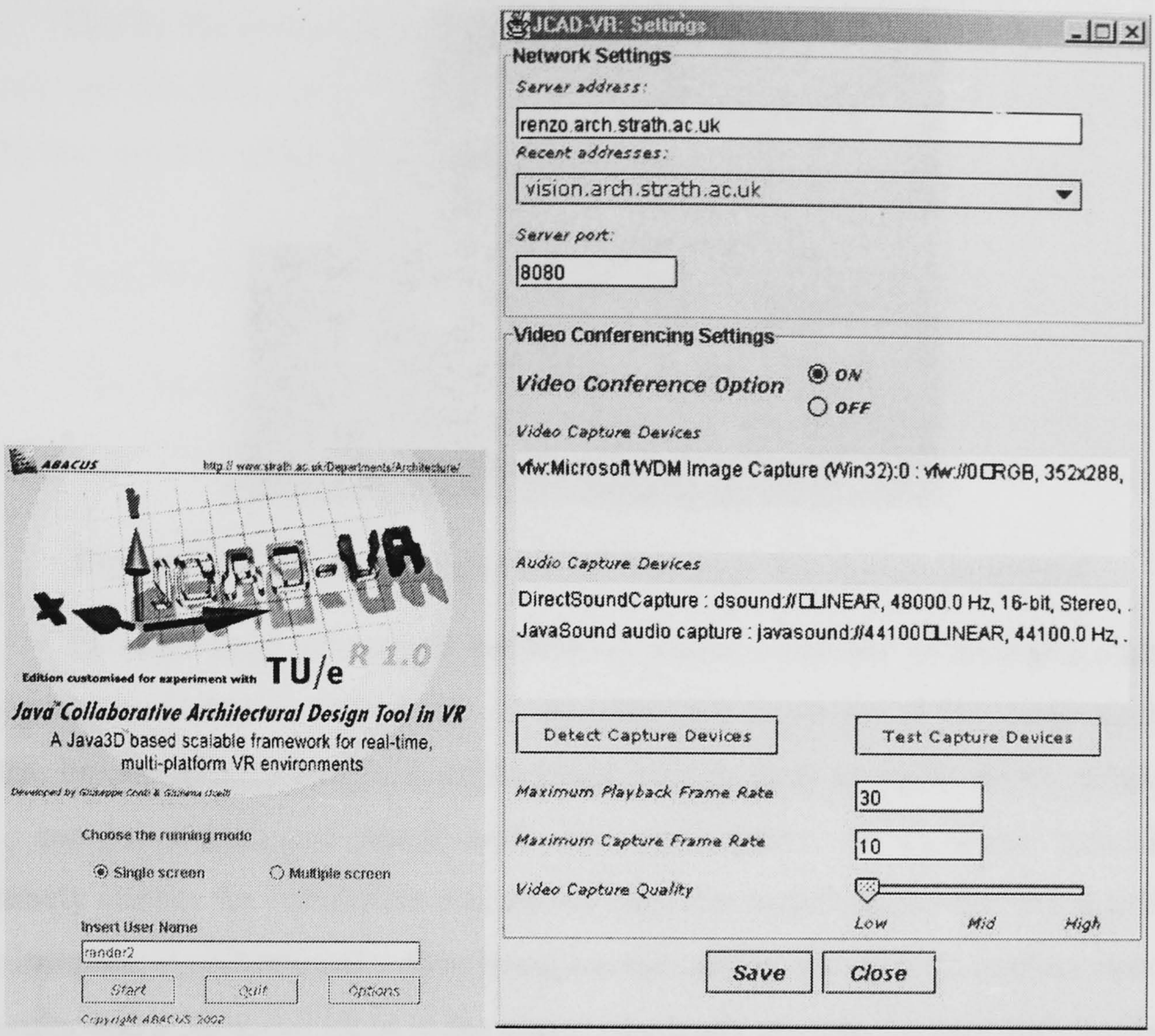


Figure 5.5: The initial panel and the option panel of JCAD-VR

The user can also decide whether to initiate the system in single or multiple screen modes. The former is provided for standard computer screens whilst the latter has been customised for the multi-projector Reality Center™ used throughout the research. Finally from the initial panel it is also possible to activate or de-activate the video conferencing facilities and if video conferencing is activated then support for video capturing device recognition and checking is provided.

Once the system is initialised every window disappears freeing the space for the 3D graphic user interface of the system and to the environment loaded during the start-up. A set of 3D menus and icons appears on the screen and through them each user can interact with the system and with the other participants (See Figure 5.6).

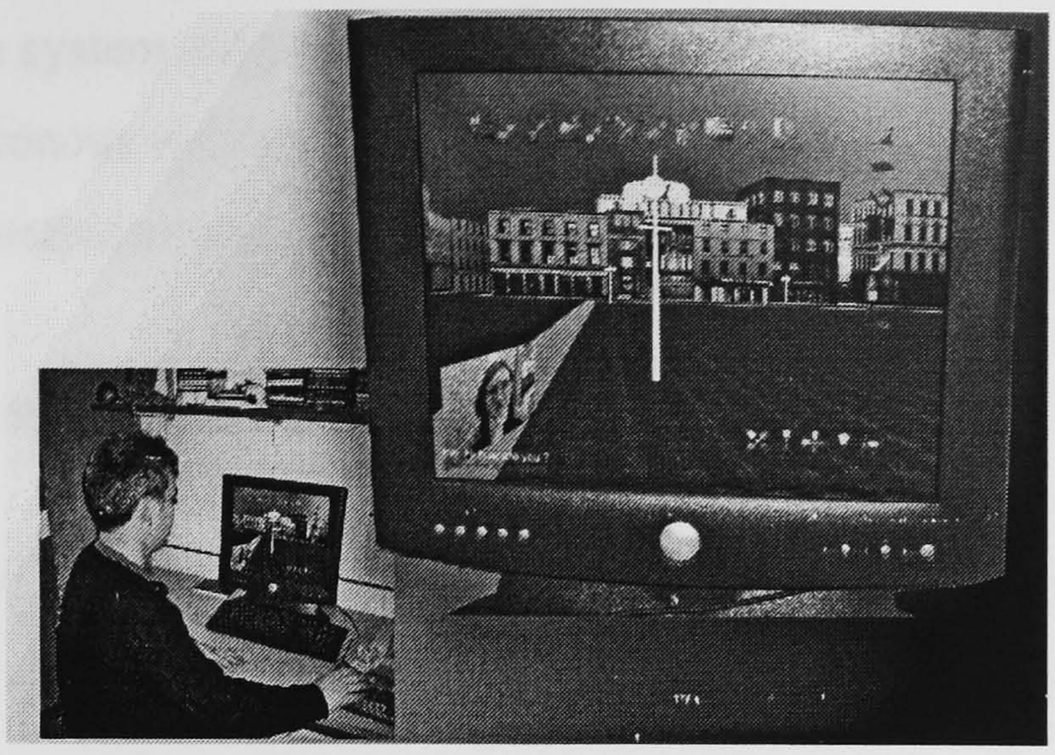


Figure 5.6: The interface of JCAD-VR after the system is started

Through these menus it is possible to access a number of functions such as navigation and creation of objects. In the latter case a number of 3D shapes such as cones, boxes, spheres etc., and architectural objects such as walls, doors, windows etc., can be created and shared with other participants. An automatic procedure routinely checks for constraints and allows only the modifications that are possible. For instance, it will prevent a door being moved onto or too close to another door. A 3D-ruler and a 3D-panel close to the object constantly provide the user with feedback on the parameters that can be edited such as size, materials and cost.

The user can translate, rotate and scale the objects in the environment in every direction through the visually simple dragging of the arrow representing the x, y or z axis.

Users are represented in the environment through avatars labelled with their names and they can communicate through traditional chat as well as via voice and video conferencing or they can sketch freehand drawings on a shared electronic whiteboard. Moreover, due to its collaborative nature, any user can contribute to the creation and manipulation of the environment. Every time a user changes an object the system automatically upgrades this object's geometry and/or position throughout the network and the new configuration is made available to all the users in the world. This raises the issue of the possible concurrent action on the same object by more than one user. An object locking mechanism, implemented in JCAD-VR, tackles this problem by ensuring that every time an object has been selected it becomes inaccessible for any other user until it has been released.

Finally the system supports communication and interaction between users, it allows real synchronous collaborative design and it makes the design process a true multi-user collaborative experience.

5.4.3 The Modular Approach

The overall structure of the JCAD-VR system is shown in Figure 5.7. The system is divided into two different packages: a server and a client.

Following the Object-Oriented paradigm the architecture of JCAD-VR has been developed in a modular fashion. This has allowed scheduling of the implementation of independent self-functioning modules, concurrent software development and ultimately the delivery of a working prototype as a functioning core. Simultaneously this ensures the future expandability of the system through the implementation of new modules.

Each package is made of different units each containing one or more cores. Each core can be accessed through a number of modules that have a specific function within the main framework.

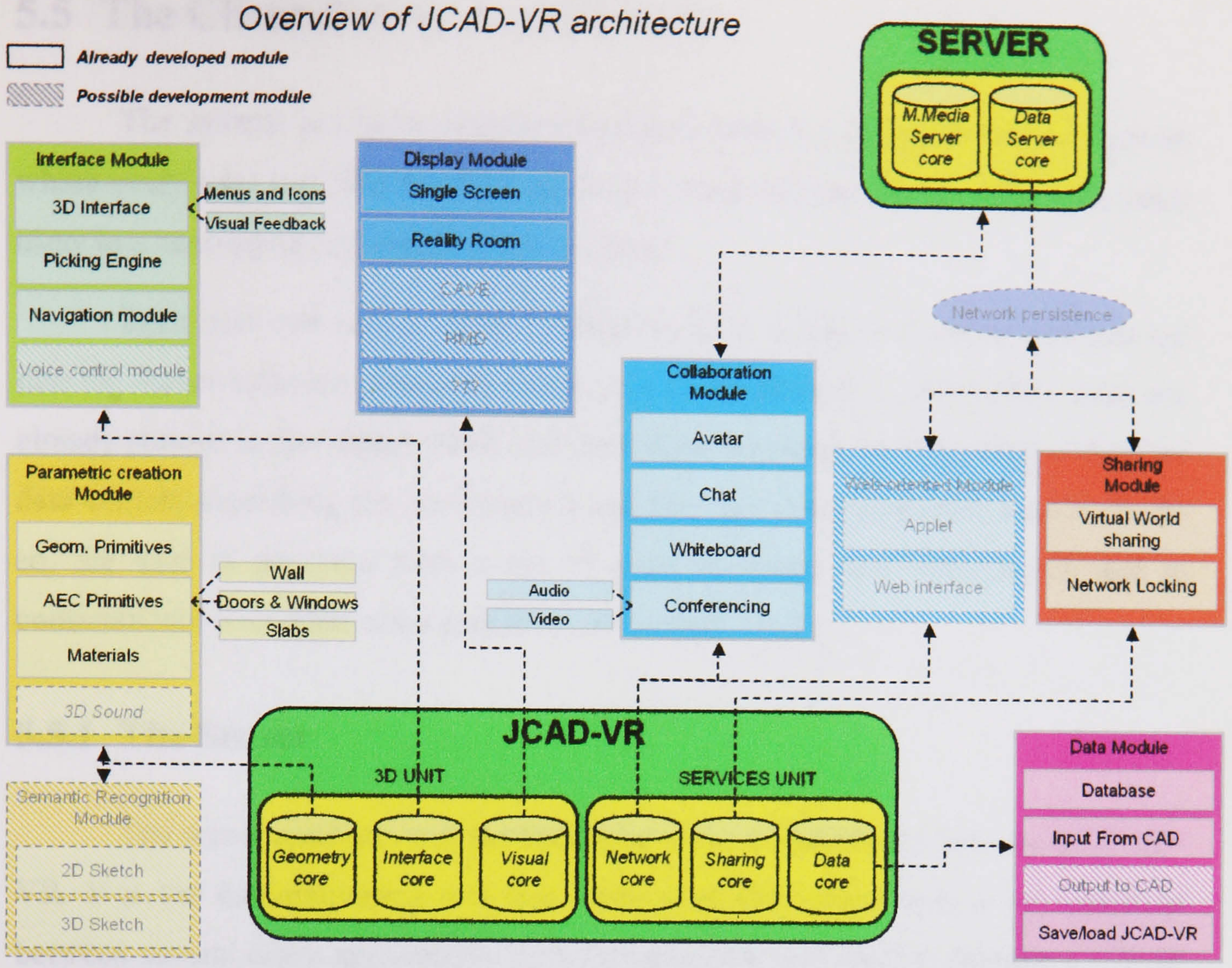


Figure 5.7: The JCAD-VR framework schema

The adoption of this approach gives a very high level of flexibility to the overall architecture since specific functions can be included at different times without interfering with the general structure of the framework. Rather than developing a monolithic closed design system, the modular approach provides ready-to-use sub-systems dealing with different functions. This way of working is therefore less error-prone, since it allows the control of independent sub-sections, it strengthens the general structure of the system and it brings extreme flexibility and expandability to the entire system.

In the following paragraphs an overview of Figure 5.7 will be provided and the meaning of the different elements will be given. The set of hatched modules in the diagram represent modules not developed and of lower priority. They will be subject of the last chapter which discusses the conclusions and possible further developments of the system.

5.5 The Client-Server Architecture

The system has been entirely developed around a client-server architecture where every user logs into a virtual world and starts sharing design tasks with other users in a concurrent and synchronous fashion.

Each user can start a client application from a remote location and can log into the server software. Once the connection is established, if some other users are already present in the virtual world and have already created objects, they receive the data content describing the environment and they can start interacting with it. To do so, the user is provided with a set of tools to create and edit objects and to communicate with other users present in the virtual world.

5.5.1 The Server

The server application is the backbone of the collaborative features of JCAD-VR. It is the data-delivering unit that looks after the information to be broadcast between several client applications performing actions and queries through a TCP/IP network and relying on the server for receiving data updates.

The server application is made of two cores (See Figure 5.7): the *JCAD-VR Data Server* which looks after the VE information to be broadcast, and the *JCAD-VR Multimedia Server* which streams audio and video for the video-conferencing tool. Both parts are closely linked to each other and they are seamlessly integrated to comprise the general server package.

As an independent part of the framework the server has an autonomous and simpler interface that provides primarily information about the network status.

The intrinsic multi-platform nature of JCAD-VR, inherited from the language used to code it, allows the server to transmit data to a broad range of platforms, from normal PCs to a supercomputer running a Reality Center™.

5.5.2 The Client

The client is the application used to run the virtual environment. To do so two closely connected units have been developed (See Figure 5.7): a *3D unit* and a *services unit,* each made of three different *modules*. Using the analogy of the human body the former could represent the heart while the latter could be considered the nervous system.

The *3D unit* is the broad part of the framework that handles all the information regarding the "visible" aspects of the virtual world. It is the logical container of three cores: *geometry*, *interface* and *visual core*. The first includes the code necessary to create and modify geometric entities, the second to interact with the human computer interface and the third to deal with several different display devices.

The *services unit* instead is the section of the framework that handles all the information regarding the management of the virtual world. It is the centre for the interconnection between users: it manages network connections and the exchange of data between users through a *network core*, it handles the sharing of the virtual environment through the *sharing core* and finally, through the *database core*, it keeps track of the state of the virtual world and it makes the retrieval of information possible about objects present in the scene.

5.5.2.1 The Geometry Core

The *geometry core* handles the creation of 2D and 3D objects. It permits the retrieval of 3D-objects present in the library and it allows the creation of three-dimensional geometrical primitives such as cones, boxes and spheres. Il also allows the creation of more complex and specific architectural entities, referred to in the following chapters as AEC (Architecture Engineering Construction) objects, for example walls, slabs, windows, and doors.

Even though further details will be provided in the following chapters it is worth mentioning the great difference between "simple" shapes and specific AEC objects. For instance, although a wall is essentially a box from the visualisation point of view the system treats it in a completely different way. In fact, while a box is just

regarded as a simple shape without any extra qualities apart from low-level attributes like material or cost, the wall is considered as an entity with "topological" properties. First of all it is made of two different surfaces - the internal and external face - and of an internal core. Furthermore it can be the parent of another object - such as a window or a door - and therefore it can hold information of a different type such as the number of windows or doors attached to it and their relative position.

The geometry module also provides the means for attaching materials to objects. These materials are stored, together with a number of 3D objects, in a library available to the user.

5.5.2.2 Interface Core

The *interface core*, as its name suggests, handles the Human-Computer Interface. As previously mentioned one of the main goals of the JCAD-VR framework was to achieve a *transparent* interface. The concept of *transparency* refers to the idea of an interface that is not detached from the 3D world but on the contrary is an integral part of it. As mentioned, instead of using traditional menus and toolbars, this 3D interface allows the user when immersed in an environment to find the means for the interaction within the VE itself. Therefore the user can interact through 3D widgets present in the virtual world which can be manipulated at their convenience just like any other object in the environment.

Likewise visual feedback is also provided within the environment, for instance in the form of 3D-rulers showing the size of objects or 3D-icons showing the operation being performed on an object.

Due to the limited need for advanced settings, such as front or clipping distance, field of view etc., a traditional window-based control panel was provided to complete the interface for advanced visualization options.

5.5.2.3 Visual Core

The *visual core* is the part of the framework that allows the interface with the visualization devices through *display modules*. The obvious computational constraints imposed by the use of different hardware is solved by creating a structure that is flexibly scalable which can also deliver images for a range of viewing devices,

from the simple desktop monitor to the more complex tessellated screen for immersive environments. When JCAD-VR is loaded the user is asked to choose whether to work on a single screen or in a multiple screen mode and they can switch between these two modes according to the machine the application is running on.

The adoption of a modular approach to the framework shows clearly its advantages in the development of these modules. In fact in response to the obvious hardware limits imposed by the use of different platforms, the system had to be written to be easily customised to run on PCs as well as on the Sgi supercomputer. In the former the system runs on PCs whose video-card displays the virtual world on a traditional window or at full screen while in the latter JCAD-VR can take advantage of the 12-processors Sgi Onyx2 system which power the Reality Center™, ABACUS, in the University of Strathclyde, Glasgow (University of Strathclyde, 2002).

The modular architecture of JCAD-VR allowed the development of independent modules that can be easily customised to respond to the needs of each specific hardware configuration. Therefore when JCAD-VR is launched on the Sgi the *display module* customised to handle the Reality Center™ can take advantage of the supercomputer's graphics power to run on a 5-metre wide 2-metre high, tessellated screen where 3 Barco projectors create a 160-degree panoramic image. In addition to this approach, other *display modules* might be easily adapted to allow use of different VR devices such as CAVEs or Head-Mounted Displays without interfering with the general structure of the system.

5.5.2.4 Network Core

The *network core* connects the independently coded client and server packages of the framework providing the means for the transmission of information through a communication channel based on a TCP/IP network.

To ensure communication between users, represented in the 3D world by avatars, different means are provided, from basic chat to voice and video conferencing. Freehand sketching in 2D is also possible through a shared electronic

whiteboard. Every user can take advantage of the synchronous multimode communication media while working collaboratively on a design task.

5.5.2.5 Sharing Core

Like the *network core* the *sharing core* bonds clients and server through the exchange of numerical information. This part of the system broadcasts information about the objects present in the virtual environment and transmits actions and modifications performed on them thus providing much of the foundation for the collaborative features of the system.

This core ensures network consistency through a distributed network-locking mechanism that attributes a unique number, consistent for all the users in the system, to any object created across the network and sets the user priority on selected objects. When a user selects an object, this becomes *locked* and an event is sent through the network to other users to prevent them accessing the object. Locked objects can no longer be chosen by other participants until they are unlocked. This mechanism does not allow more than one user to edit an object at the same time and it is designed in order to ensure consistency throughout the system.

5.5.2.6 Database Core

Finally the *database core* handles the internal database that keeps track of the creation or manipulation of objects in the virtual scene by all the users present in the environment. Through it the system can retrieve information on geometric primitives and materials, etc. The internal database is closely coupled with the network core. It does not only keep track of what is happening within the user's virtual world but most importantly, upgrades through the network, the information broadcast by other users' internal databases. If for instance a new object is created or its status is changed, the system will upgrade the internal database of each user in real-time no matter who is performing the action on the object.

Finally, for the convenience of the user, an I/O module allows storing and retrieval of the database content for the save/load operation.

5.6 Delimitation of Scope

The framework proposed in the previous pages tries to address the major issues of a collaborative VR-based design tool for architecture. Nevertheless a number of limitations narrow the scope of this framework, specifically:

- The implementation of the application is only at the prototype stage and it has to be considered as a proof of concept rather than a finished commercial package.
- In general the implementation of new tools was preferred and has been considered of greater scientific interest than the optimisation of already developed modules.
- The application has been specifically designed for architects and not for Virtual Reality experts therefore particular attention was paid to having a familiar graphical user interface and its ease of use.
- The research focuses on the early stages of the architectural design process thus the software application has been developed to be used during the conceptual modelling phase and not as an advanced modeller.
- JCAD-VR is a tool to promote flexibility in design not a CAD/CAAD package and therefore it lacks the routines and algorithms that can handle complex geometries and solid modelling tools usually provided in commercial applications. The abstraction required at the first stage of the design process justifies the support for elementary shapes.
- Although it has been shown to provide support for immersive and semi-immersive VR configurations, the implementation of the system as mainly a Desktop-VR application is justified by the collaborative nature of the system. The need to run several clients concurrently makes the Desktop-VR choice the only realistic approach for a feasible multi-user VR experience.

As already mentioned, due to the evident complexity of the framework proposed, the developed prototype represents a limited, yet fully functioning, part of

the main schema represented in Figure 5.7. Therefore the result should be considered as proof of the feasibility of such an application for architecture.

The previously mentioned shaded modules of Figure 5.7 represent modules with a lower priority which have not been developed and their further development will be subject of the last chapter of the thesis. An order of priority was imposed on the modules in order to deliver the most important or innovative features of the framework. In this way the overall system was provided with a substantial set of functions addressing all the main issues mentioned in the research and making the final prototype a fully functioning application.

5.6.1 General Technical Issues

From the technical point of view, the approach followed when programming JCAD-VR has mostly favoured functionality or user-friendliness over efficiency, both in the code and in the overall performance of the system. The choice could be summarised by stating that programming paradigms, as well as coding solutions, have been chosen to privilege the user's point of view rather than the absolutely perfect performance of the system. The choice has been justified by the ever-increasing power of the hardware that has brought the overall system to a satisfactory level of performance on most systems, well above the threshold of 30 frames per second.

Further, the multi-platform nature of the Java™ programming language has raised the issue of performance across different platforms and operative systems. The wide range of hardware configuration available across the number of platforms virtually supported by the system has lead to a choice in the optimisation of the code. Due to its wider availability, and its achievement of excellent rendering performance the Windows platform was considered of greater importance to the scope of the project. In contrast a lower priority was given to the optimisation of the code for the Sgi Irix platform, due to its range of hardware configurations, graphics subsystems and visualisation devices which would make it too limiting to use this particular configuration as the sole benchmark to evaluate the application.

The different ways the Java™ Virtual Machine (JVM™) handles threads in Unix-based systems, together with the thread-related issues typical of multi-processor systems and the inefficiency of the Irix port of the JVM™, specifically produced a lower, yet satisfactory, performance on the Sgi system used. In addition the system has not been fully tested using Linux and it has not been tested at all with the other supported operative systems such as Solaris, MacOs X, etc.

The fact of the always-increasing number of new releases of JVM™ and of the APIs used by JCAD-VR might contribute to incompatibility issues. Therefore Appendix D has been reserved to specify the hardware and software used to fully test the application.

5.7 Conclusions

The multidisciplinary nature of this research was the opportunity to investigate collaborative design issues, the role of interfaces inside CAAD packages, the design process in the first stage of its conception, the use of Virtual Reality in architecture as a design tool, the issue of collaborative systems and finally a number of technical issues.

As previously shown JCAD-VR proposes a user-centred prototype of a collaborative VR application where intuitiveness and control over the environment results in an innovative and more involving design experience.

The following chapters will detail the development of the system, and they will focus on the part of the framework depicted in Figure 5.8 and will provide an in-depth view of its features and the technical solutions.

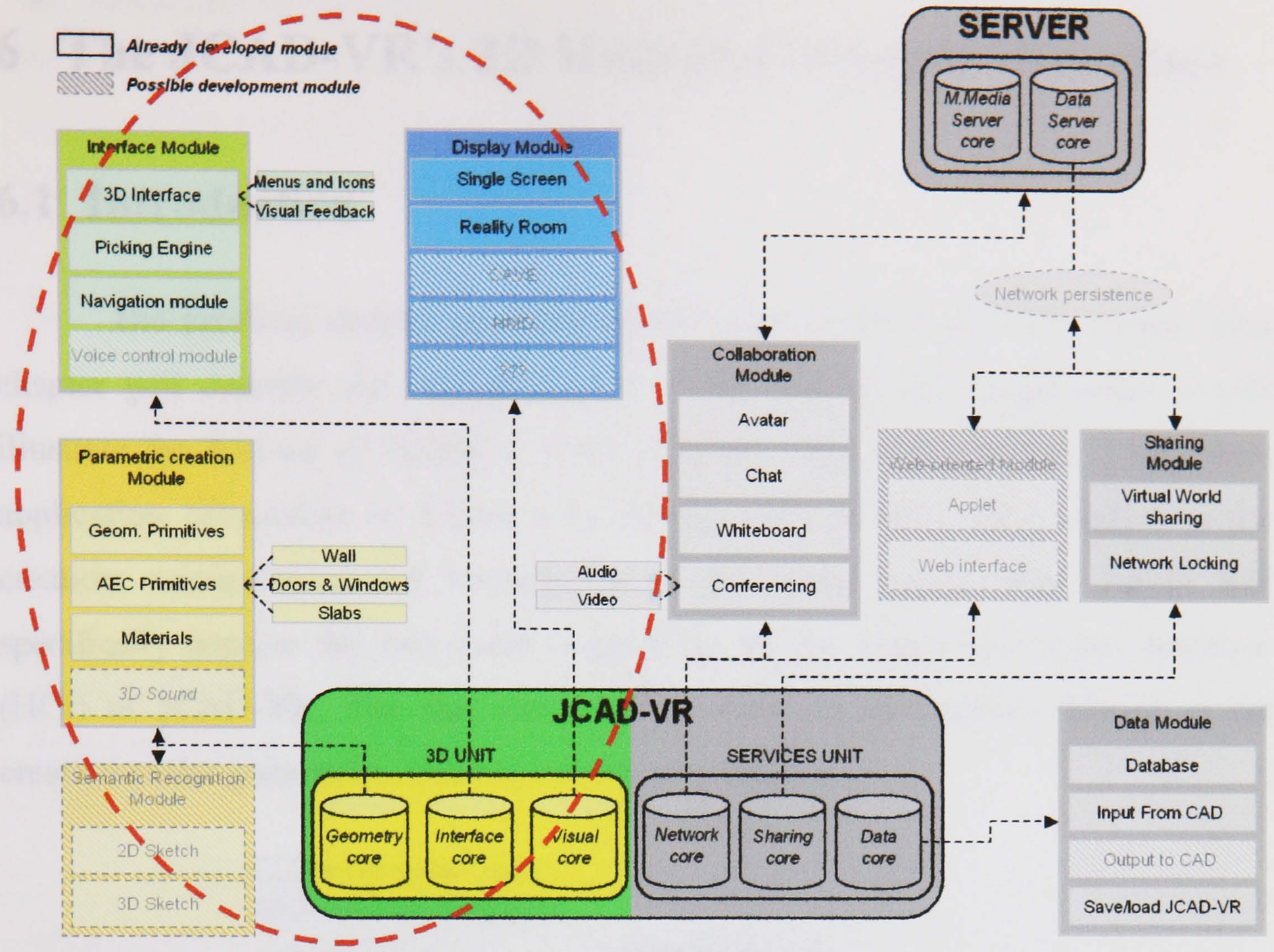


Figure 5.8: The portion of the framework studied in this thesis

6 The JCAD-VR's 3D Human-Computer Interface

6.1 Introduction

The previous chapter provided an overview of the JCAD-VR system. This chapter will describe the implementation of the system, and in particular, it will illustrate the first set of functions of the 3D-Unit. This is the section of the client application (illustrated in Figure 6.1) dealing with all the issues related to the creation, manipulation and visualisation of the virtual world. This chapter will specifically analyse the two cores responsible for the Human-Computer Interface (HCI) of JCAD-VR. The last core, dealing with the mechanisms related to the creation of the geometries, will be discussed in Chapter 7.

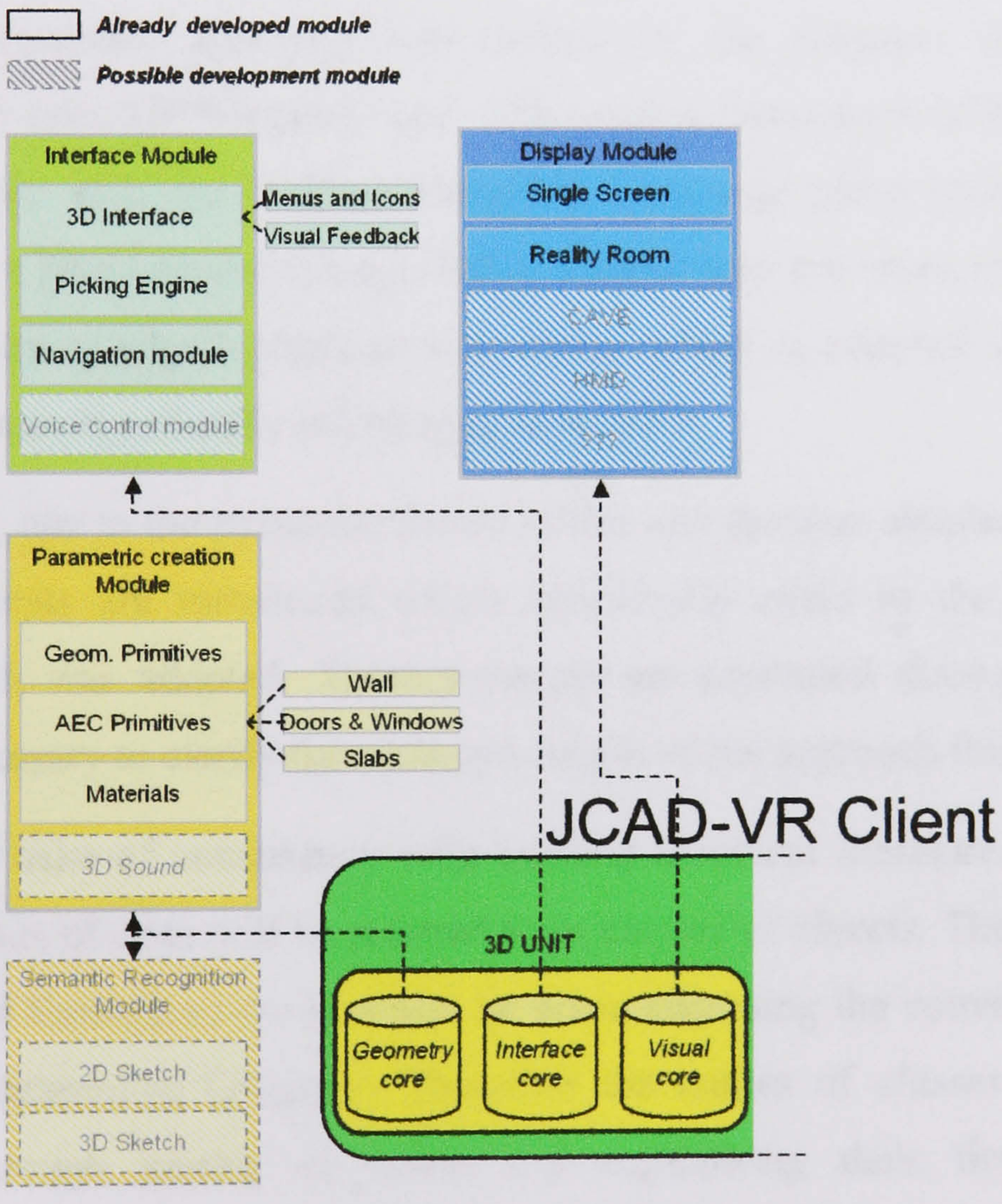


Figure 6.1: The architecture of the 3D Unit

The first part of this chapter provides a description of the methodology adopted to develop the HCI of JCAD-VR. The following sections describe in detail

the architecture of the Visualisation core which forms the part of the HCI dealing with the visualisation of the virtual world. Finally, the Interface core will be the subject of the last section which will provide details of the 3D-GUI, the interaction techniques adopted and their implementation.

6.2 Justification of the Methodology

This and the following chapter, as mentioned, will describe the implementation of the JCAD-VR system. Both chapters are characterised by a number of diagrams, due to their technical nature.

The diagrams are provided to illustrate the mechanism behind the system without the need to quote the code. In this way the different solutions adopted can be described visually, focusing on the general mechanism behind the technical solutions rather than on language specific implementations.

A tree-structure approach was chosen for the diagrams illustrating the structure of the Java 3D™ related code. The graphic convention of these diagrams does not comply with the Unified Modelling Language prescriptions for Object analysis (Object Management Group, 2002). Instead they are represented according to a widely acknowledged graphical convention which is adopted throughout the technical literature specifically referring to Java 3D™.

Further, due to the particular nature of this and the next chapter, a number of technical concepts are introduced which specifically relate to the programming language which was adopted. These concepts are presented throughout the text where it is necessary to clarify the technical details of the approach followed.

For the sake of consistency with existing technical literature, the logically independent units of code will be referred to as *classes* or *objects*. The names of the classes outlined throughout the text will be given according the convention adopted by Java™ programming language. Therefore the names of *classes* are logically constructed through joining of nouns and capitalising their first letter, e.g. SwivelWindow. The same composition rule is applied to the routines, or *methods*,

mentioned throughout the text; in this case the first name is left in lower case, e.g. openWindow.

Finally, to avoid misunderstandings between the class name and type, the name of a specific object will be written in italics whilst the type of a class will be left in plain text. Therefore, the *ViewGraph* BranchGroup is an object of the type BranchGroup named *ViewGraph*.

6.2.1 The Java 3D™ API

The previous chapter discussed the advantages introduced by the choice of Java™ as the programming language of JCAD-VR.

In particular, the use of Java™ for the development of the 3D-Unit was possible because of the availability of the Java 3D™ Application Programming Interface (API), a library specifically developed to handle 3D-graphics. Java 3D™ makes use of native hardware acceleration through the use of lower level APIs such as OpenGL® (OpenGL, 2002) or DirectX® (Microsoft Corp., 2002) and it can consequently support real-time 3D-graphics.

As Sowizral et al. (1999, p. 12) stated, "writing a VR program requires a substantial programming effort. A developer must either write code to handle the various inputs and display devices that the application might encounter or, alternatively, the developer will need to rely on a programming API designed to support VR applications. [...] The Java 3D API includes specific features for automatically incorporating head tracker inputs into the image generation process and for accessing other tracker information to control other features". Through Java 3D™ "developers can easily incorporate high-quality, scalable, platform-independent 3D graphics into Java™ technology-based applications. [...] This enables developers to build, render, and control the behavior of 3D objects and visual environments" (Sun Microsystems, 2000, p. 1).

Java 3D™ can be considered as belonging to the fourth generation of programming languages for 3D graphics (Sun Microsystems, 1998) (See Figure 6.2).

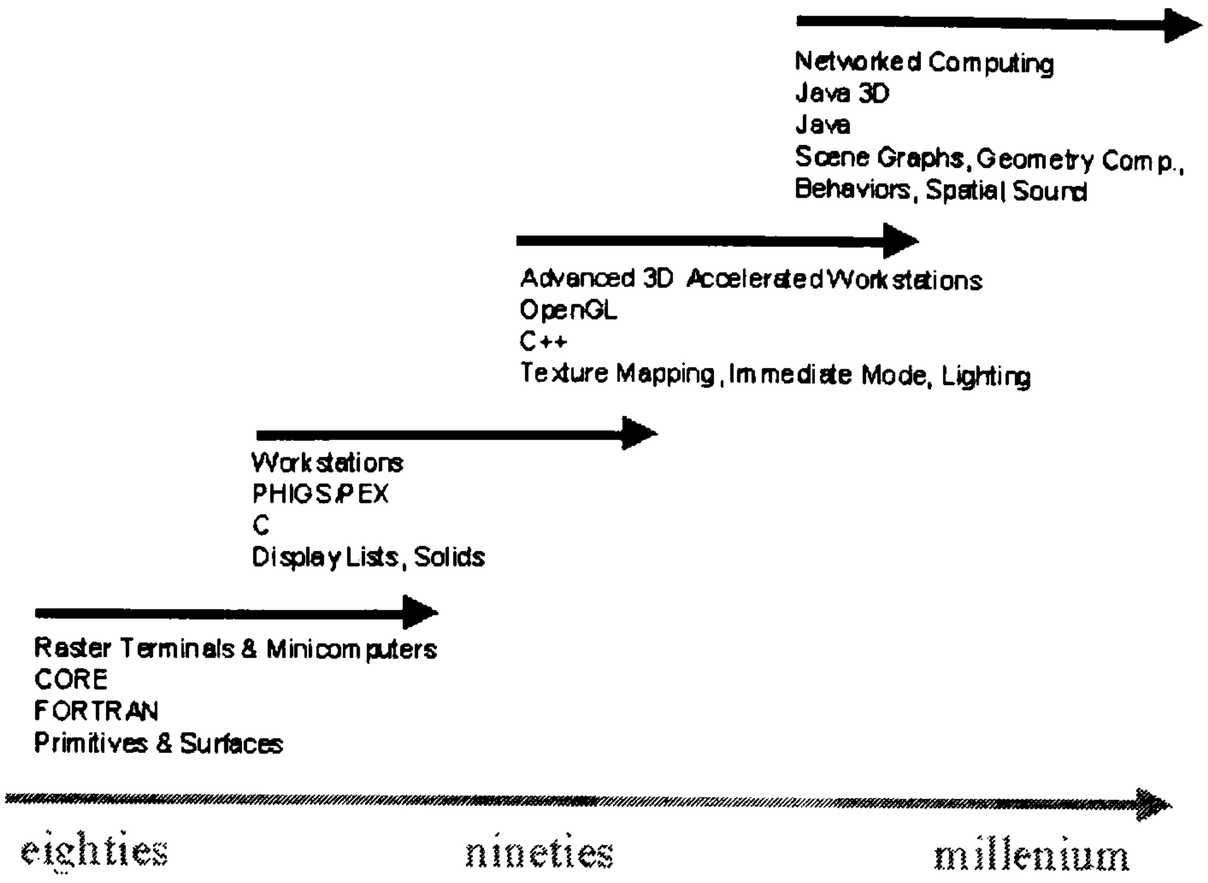


Figure 6.2: The evolution of 3D graphics (Sun Microsystems, 1998)

In CAD/graphics programs developed in the 1980's simple shapes were drawn on mainframes or minicomputers in Fortran using the CORE standard. Towards the end of the eighties, the arrival of workstations together with the adoption of C language and the development of libraries like PHIGS resulted in the first 3D graphics applications. In the nineties the fast development of specific hardware, fostered by the development of the C++ language and the OpenGL® API, led to the introduction of texture mapping.

Today, the need for more efficient software architecture has led to the development of fourth generation languages for 3D graphics like Java 3D™. As stated, Java 3D™ still relies on lower level APIs like DirectX® or OpenGL®, however a new higher programming layer has been introduced providing features such as multi-platform support, simplified handling of behaviours and geometries and better overall control of the system.

6.2.2 The Desktop-VR Choice

JCAD-VR is primarily a monoscopic Desktop-VR system whose flexible architecture has been expanded to support a semi-immersive visualisation configuration.

The Desktop-VR approach has been chosen to suit the scope of the application. JCAD-VR is a collaborative VR system and therefore the access to a

number of workstations was a fundamental requisite for the system. Desktop-VR systems can be easily deployed, although providing a lower level of immersion. The adoption of other solutions, such as those based on high-end immersive technologies would have heavily limited the scope of the research.

Nevertheless, as will be shown in the following sections, the modular nature of JCAD-VR encourages the potential expansion of the system into other visualisation set-ups.

In terms of the control devices, the choice of a Desktop-VR system forced the development of an interface relying on traditional devices such as the mouse and keyboard. However, the hardware independent logic mechanism behind the interface (described in Section 7.4) together with the modular architecture of the system allows expansion to more complex pointing devices such as 6-DOF mice and Virtual Gloves etc. The logic behind the interface is in fact independent of the devices used.

6.3 The *Scene Graph* Hierarchy Approach

In computer graphics the term *scene graph* is used to describe the abstraction of the structure used to support the elements necessary for the creation of a virtual world. "The scene graph contains a complete description of the entire scene, or virtual universe. This includes the geometric data, the attribute information, and the viewing information needed to render the scene from a particular point of view" (Sun Microsystems, 1999).

The idiom *scene graph* is a more familiar synonym of the term *Directed Acyclic Graph* (DAG), also known in discrete mathematics as *tree*. A *tree* is a specific type of *discrete graph*, a graph based on *nodes* and *edges*, since it is characterised by two specific features: it is *acyclic* and *directed*. The structure of a *tree* has a logical, parent-to-child arrangement, hence the attribute *directed*, and it cannot contain cycles, hence the adjective *acyclic*.

As a consequence the paths that connect each node to the root of the DAG are unique and this feature gives the characteristic layout resembling an upside-down tree (See Figure 6.3), which explains the adoption of the idiom *tree*.

The developers of Java 3D™ have stressed its *tree* structure by adopting a formal convention that divides the nodes of the DAG between *groups* and *leaves*. The former can have one or more *children* nodes while the latter cannot, thus they are usually the terminal nodes of the graph, like the leaves on a tree. Finally, the state of each node is influenced by the combination of its parent nodes up to the root of the graph.

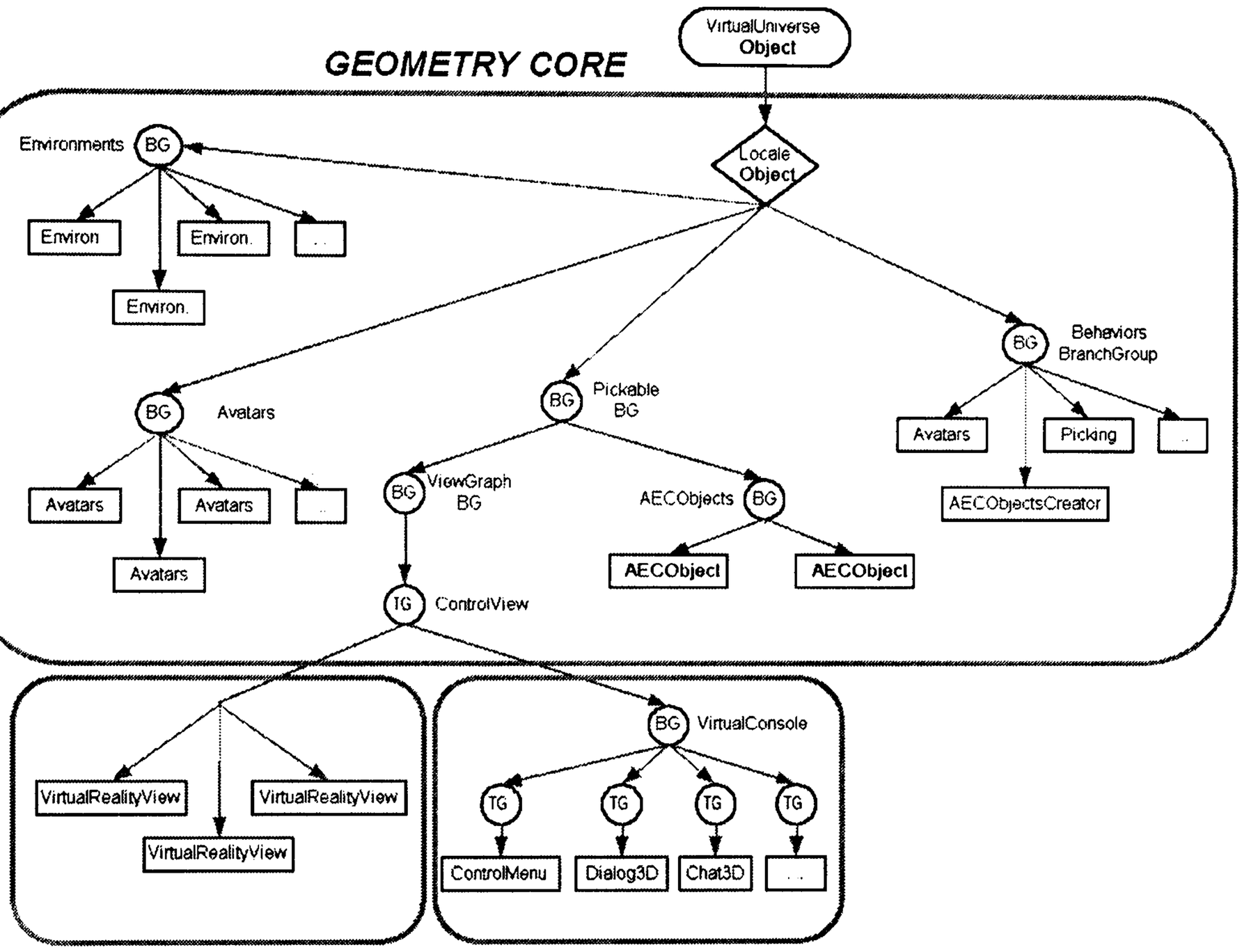


Figure 6.3: An overview of the DAG (Directed Acyclic Graph) of the 3D Unit

However, the rigid spatial convention implied by the use of a DAG is mainly adopted for the advantage of the programmer. In fact the mathematical representation within the computer memory usually differs substantially from the layout of the DAG. Nevertheless, the adoption of a *scene graph* encourages a logical organisation of the elements used for the construction of the virtual world. This logical arrangement can ultimately be exploited by the system to automatically reconfigure the DAG structure into an optimised layout through the compilation process.

As a consequence, from the implementation point of view, the organisation of the 3D-Unit (illustrated in Figure 6.1) can be shown through the overview of the DAG (shown in Figure 6.3). From this perspective it is possible to outline the three cores (described in Figure 6.1) through the content of the main branches.

The rest of this chapter describes the contents of the branches for the implementation of both the Visualisation and the Interface Cores. Due to its complexity, the remaining core will be described independently in the following chapter.

6.4 How the Scene is Shown on the Screen: the Visualisation Core

The visualisation subsystem is one of the most important parts of a VR application since it provides the means for the *renderer* to be able to display the content of the virtual world.

JCAD-VR's Visualisation Core administers the visualisation process through a set of classes that represent in a mathematical form the different physical features of the VR set-up. These objects are placed on a specific branch of JCAD-VR's DAG, called the *visualisation sub-tree*.

The data contained within the visualisation sub-tree is necessary to fill in the fields of a number of matrixes, the combination of which gives the *projection matrix* used by the *renderer* to visualise the content of the virtual world. In short, the *projection matrix* contains the information necessary for the *renderer* to calculate the correct point of view according to the user's position and orientation, the internal geometry of the view (See Section 4.4.2) and the physical configuration of the devices used by the VR system.

6.4.1 The Visualisation Sub-Tree

Figure 6.4 shows (in greater detail than Figure 6.3) the layout of the *visualisation sub-tree*, which is the section of the DAG hosting the implementation of the Visualisation Core.

From analysis of Figure 6.4, it is possible to see that the visualisation sub-tree contains the objects necessary to render both the point of view (POV) of the user and the objects used for the three dimensional GUI. In fact the 3D-GUI of JCAD-VR consists of a number of objects placed close to the user's point of view and therefore it actually belongs in the virtual world rather than being relegated to another area of the screen as in most WIMP (Window-Icon-Menu-Pointer) applications. Instead the logical mechanism behind the 3D-GUI, is handled by the Interface Core, an independent part of the system which is explained in Section 6.5, later in this chapter.

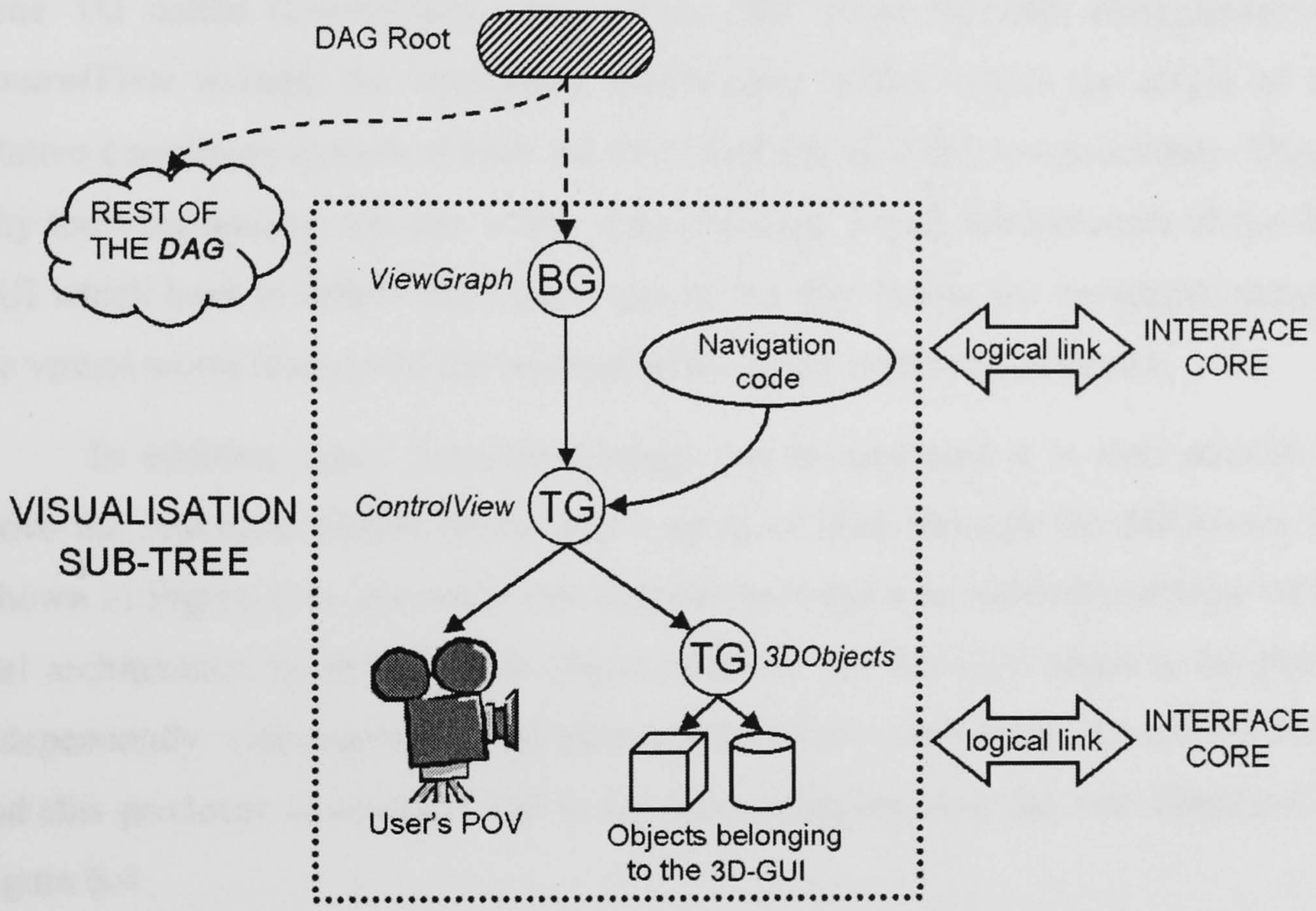


Figure 6.4: A simple overview of the visualisation sub-tree

The first node of the visualisation sub-tree (See Figure 6.4) is the *ViewGraph* BranchGroup (BG), a special type of group node in that it can be the root of another sub-branch, unlike a leaf node. The BranchGroup node has a number of unique features: first of all the sub-branch attached to it can be removed or added at run-time and secondly this sub-branch can be compiled independently from the rest of the DAG. The compilation, as mentioned before, is the process of optimisation that turns the logic oriented architecture of a DAG into a more efficient performance oriented organisation.

Moving further down the hierarchy of Figure 6.4 it is noticeable that all the objects necessary to both the POV and the 3D-GUI are under the same parent node: the *ControlView* TransformGroup (BG). The TransformGroup is a special node which handles the spatial information necessary to determine position, rotation and scale of all its children according to a relative reference system. This is done through the information supplied by a 4x4 double-precision floating-point matrix.

In short, each object of the 3D-GUI and of the POV is located in the virtual space according to the same relative coordinate system since it is placed under the same TG called *ControlView*. This means that when the user manipulates the *ControlView* through the navigation mechanism, he/she moves the origin of the relative coordinate system of both the POV and the 3D-GUI synchronously. This is why the visualisation sub-tree is the ideal container for all the elements of the 3D-GUI which have to follow the movements of the user during the navigation through the virtual world along with the representation of the visualisation device.

In addition, since TransformGroups can be cascaded it is also possible to move the interface relative to the user's point of view through the *3DObjects* TG (shown in Figure 6.4). However this diagram portrays a simplified overview of the real architecture. In reality, each object used for the 3D-GUI needs to be placed independently. Consequently each object requires an autonomous TransformGroup and this produces a structure that is far more complex than the one illustrated in Figure 6.4.

6.4.2 The Abstraction of the System's Set-Up

The adoption of a DAG gives the basis for a new low-level abstraction of the visualisation process. The developer can calibrate and manipulate the virtual environment through a number of objects that represent the abstraction of the physical VR set-up. In the case of JCAD-VR, the low-level abstraction can take advantage of the instruments offered by the Java 3D™ API, the most important of which are:

- The ViewPlatform: this represents the point of view of the user in the virtual world.

- The View: this handles the parameters necessary to define the *internal geometry of the view* (See Section 4.4.2) such as Field Of View (FOV), the type of view – if monoscopic or stereoscopic - etc.
- The Canvas3D: this represents the panel of the *2D screen space* (See Section 4.4.1) onto which the images will be rendered.
- The Screen3D: this contains the information relative to the physical properties of the display.
- The PhysicalBody: this contains the information regarding the user's body, necessary to calibrate the system when a tracking system is employed.
- The PhysicalEnvironment: this contains information regarding the physical environment where the VR system operates, necessary to calibrate the system when 6-DOF devices are used.

Therefore, through the management of these objects the programmer can implement the most convenient visualisation techniques. From this point of view all the visualisation techniques (such those based on the camera metaphor described in Section 4.5.1.3) are higher-level abstractions of the visualisation process provided by the developer for the convenience of the user. Through this adoption of a metaphor closely resembling real-life knowledge, the user can experience an immediate and natural interaction with the system.

In JCAD-VR these particular objects are all relegated to the visualisation sub-tree. In this way the classes containing information regarding devices placed in the physical world are clearly isolated from the rest of the VR environment within the DAG.

The advantage of such an approach for the developer is that they can be treated as a group, from the logical point of view, whose global function is to represent the POV within the virtual environment (as was shown in Figure 6.4). This feature gives the Visualisation Core a deal of flexibility since various visualisation configurations can be easily implemented through limited alteration of the original code.

6.4.3 A Flexible Architecture

The encapsulation of the visualisation functions within a specific portion of the DAG, provides a powerful mechanism to dynamically change the configuration of the visualisation sub-system of JCAD-VR. Indeed, one of the main requirements of the system was to be able to support both traditional monitors and much more complex configurations such as tessellated screens for semi-immersive environments.

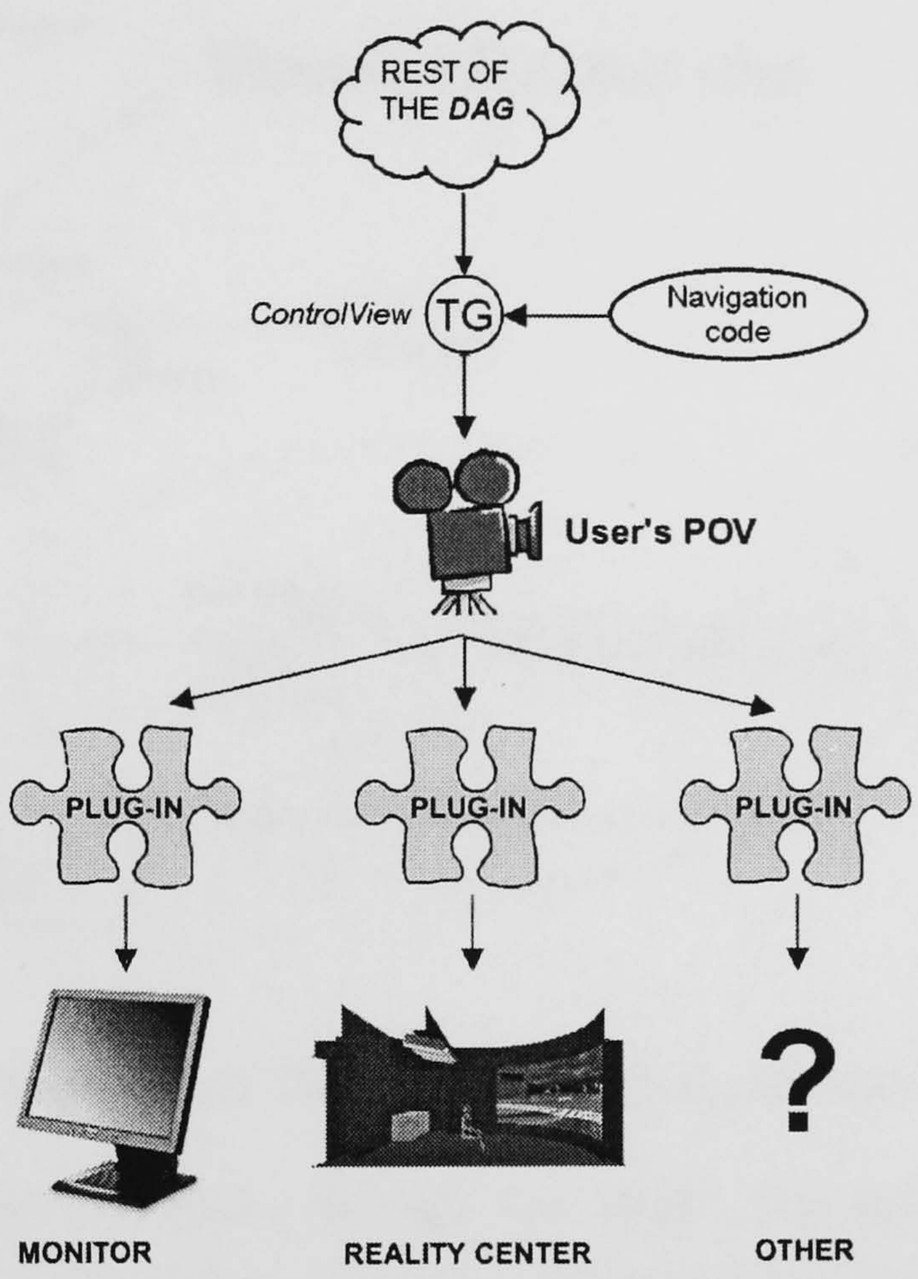


Figure 6.5: The plug-in approach used in the visualisation module

The isolation of the classes within the DAG handling the POV, has allowed the development of plug-in like architecture. This mechanism limits the alterations required for each different configuration to the portion of code necessary to handle the specific visualisation device (See Figure 6.5).

This approach, has proved flexible yet efficient and has delivered a system that can visualise images through very different viewing devices, ranging from traditional monitors to the Reality Center™ installed at the ABACUS unit of the University of Strathclyde.

6.4.3.1 The Single-Screen Mode

Figure 6.6, which gives a more detailed view of the DAG (illustrated in Figure 6.5), shows the implementation of the most simple example of a traditional monitor. For the sake of comprehensibility, the illustration does not include the code necessary for the 3D-GUI. The objects below the *ControlView* TG are used to acquire the information regarding the environment and, at the same time, to visualise the image relative to the point of view of the user on the screen.

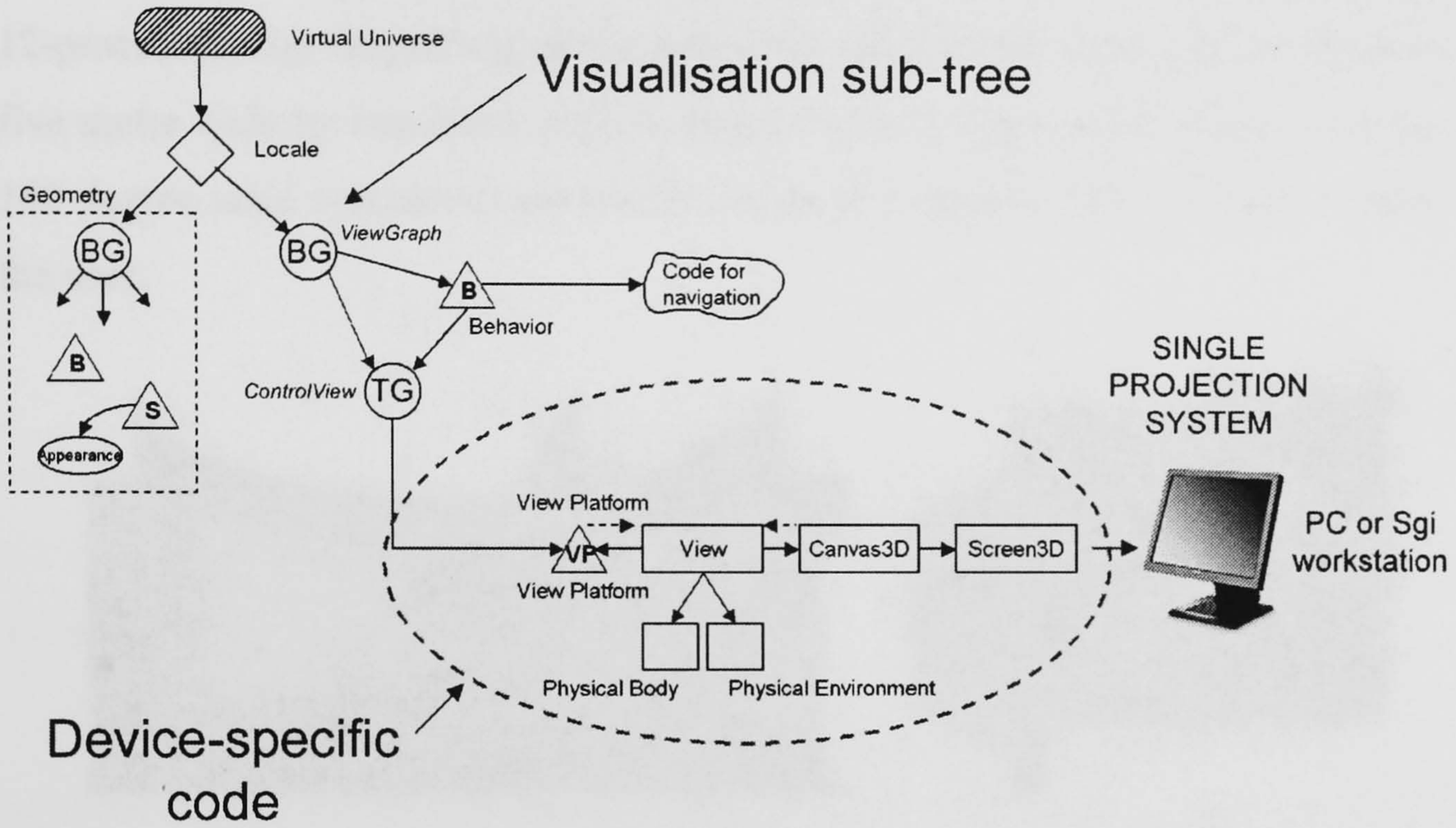


Figure 6.6: The single-screen mode DAG

When the user navigates through the world, the system manipulates the *ControlView* through the Interface Core (See Section 6.5) and as a result the objects below the TG are moved according to the wish of the user. In terms of camera metaphor, the result of this action corresponds to the user moving the virtual camera around the space. The result is due to the fact that the user moves the ViewPlatform and this is the object representing the user's location within the virtual environment. Once the correct point of view is calculated, the renderer can update the content of a virtual canvas placed on the user's monitor by generating the image corresponding to the new point of view.

Through a panel (See Section 6.5.3.4) the user can also manipulate at run-time some of the parameters of the View object representing the internal architecture

of the virtual camera, for example aspects such as the Field Of View (FOV) or the back and front clipping distance.

6.4.3.2 The Multiple Screen Mode

JCAD-VR is configured to be launched on the Sgi supercomputer which runs the Virtual Environment Laboratory (VEL) (University of Strathclyde, 2002) installed at the ABACUS unit, University of Strathclyde (See Figure 6.7).

When the system starts in multiple-screen mode it can take advantage of the 12-processors Sgi Onyx2's graphics power to visualise the virtual environment on a five metre wide by two metre high tessellated screen where three projectors create a 160 degree semi cylindrical panoramic image that entirely fills the field of view of the user.

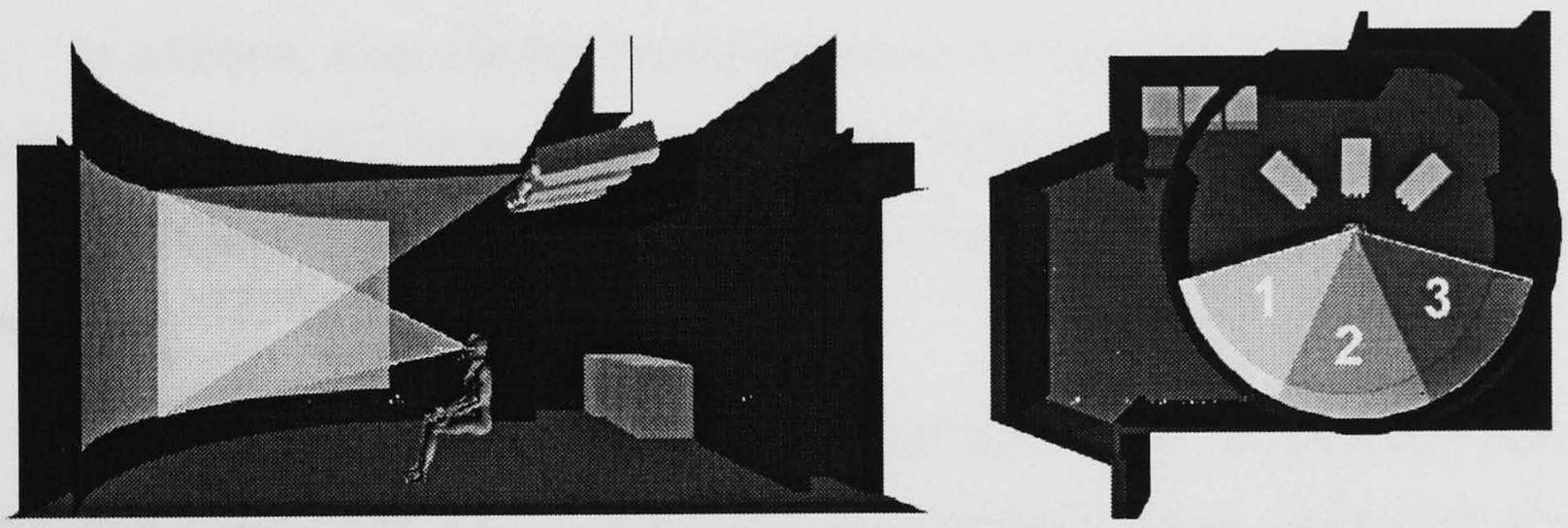


Figure 6.7: Cross section and plan view of the Reality Center™ at ABACUS, University of Strathclyde, Glasgow

From Figure 6.8 it can be seen that the architecture of the new DAG (compared to the single-screen structure of Figure 6.6) required only a limited number of adjustments to be able to handle the new configuration.

The difference between the two graphs is limited to the section of the DAG that contains the objects defining the set-up of the VR system. The previous example was characterised by one ViewPlatform-View-Canvas3D-Screen3D set while the new Reality Center™ configuration requires three sets, one for each projector.

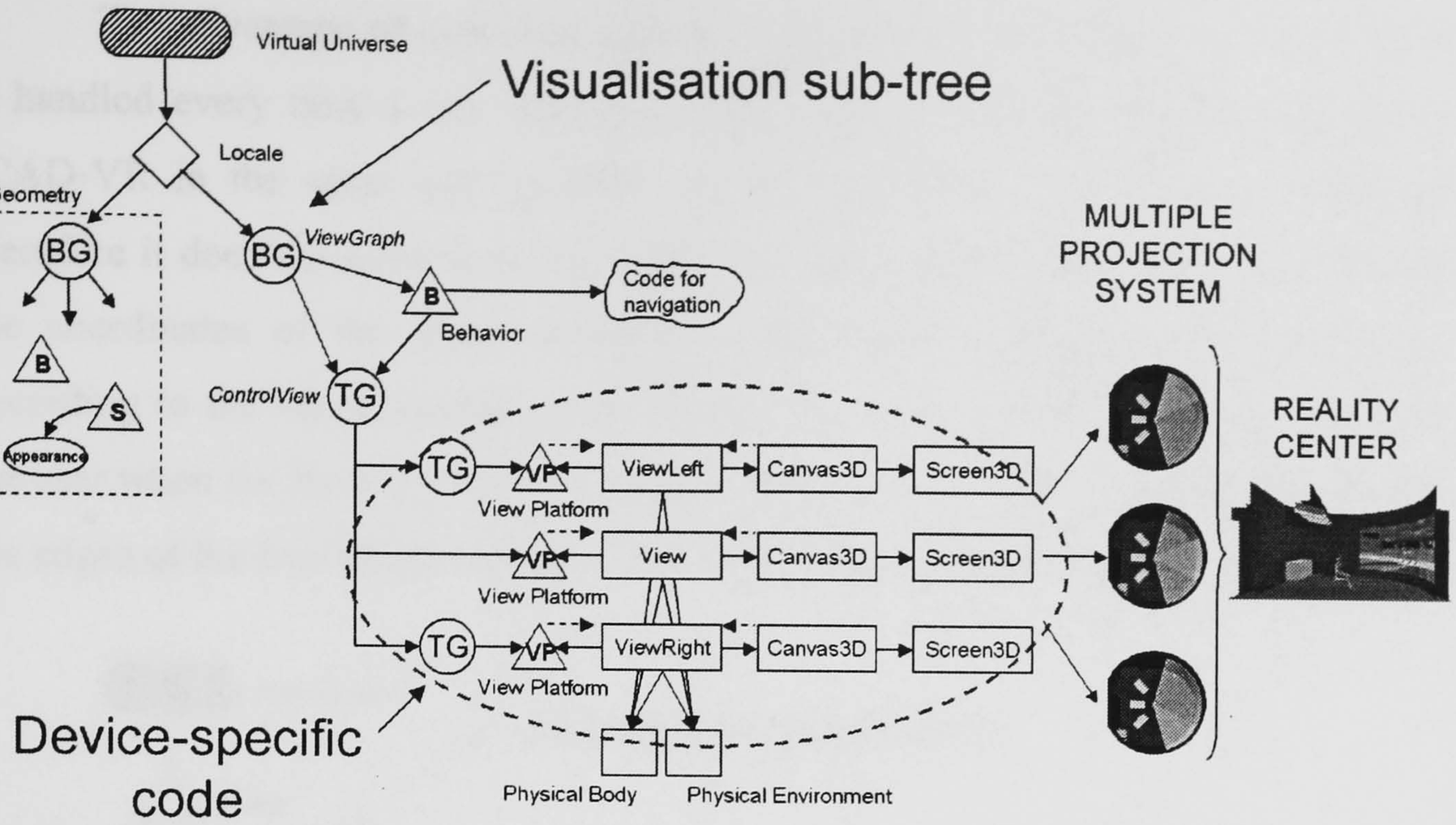


Figure 6.8: The multiple screen mode DAG

In addition, since the two lateral projectors are translated and rotated from the central one, the DAG includes two additional TransformGroups placed above the two ViewPlatforms used by the side screens to provide the renderer with the correct information.

A different and simpler approach could have been adopted however. The configuration described above uses three ViewPlatform-View groups for only one point of view. Ideally only one ViewPlatform-View set per point of view should be used (as described in Figure 6.9). In order to match the physical properties of the VR set-up, the calibration of the virtual environment could be achieved through the configuration of the relevant parameters of the three Screen3D classes, each representing a display. Each Screen3D object could be used to define the position and the orientation of the physical screen through a 4x4 matrix that replicates the internal structure of a TransformGroup.

Due to a bug in the release of the Java 3D™ API available for Sgi at the time of the development (See APPENDIX D), use of this approach caused a persistent flickering effect across two of the three screens of the Reality Center™ and therefore the previous solution, although less efficient, had to be used.

The advantage of a modular approach is also reflected in the way the 3D-GUI is handled every time a new device is implemented. The 3D-GUI is rendered by JCAD-VR in the same way as other objects belonging to the virtual world and therefore it does not need to be re-coded. The developer is only required to specify the coordinates of the initial location of the 3D-GUI since this might change according to the visual configuration adopted. For instance, for the convenience of the user when the Reality Center™ configuration is chosen, the interface is moved to the edges of the two lateral screens rather than being left cluttering the central one.

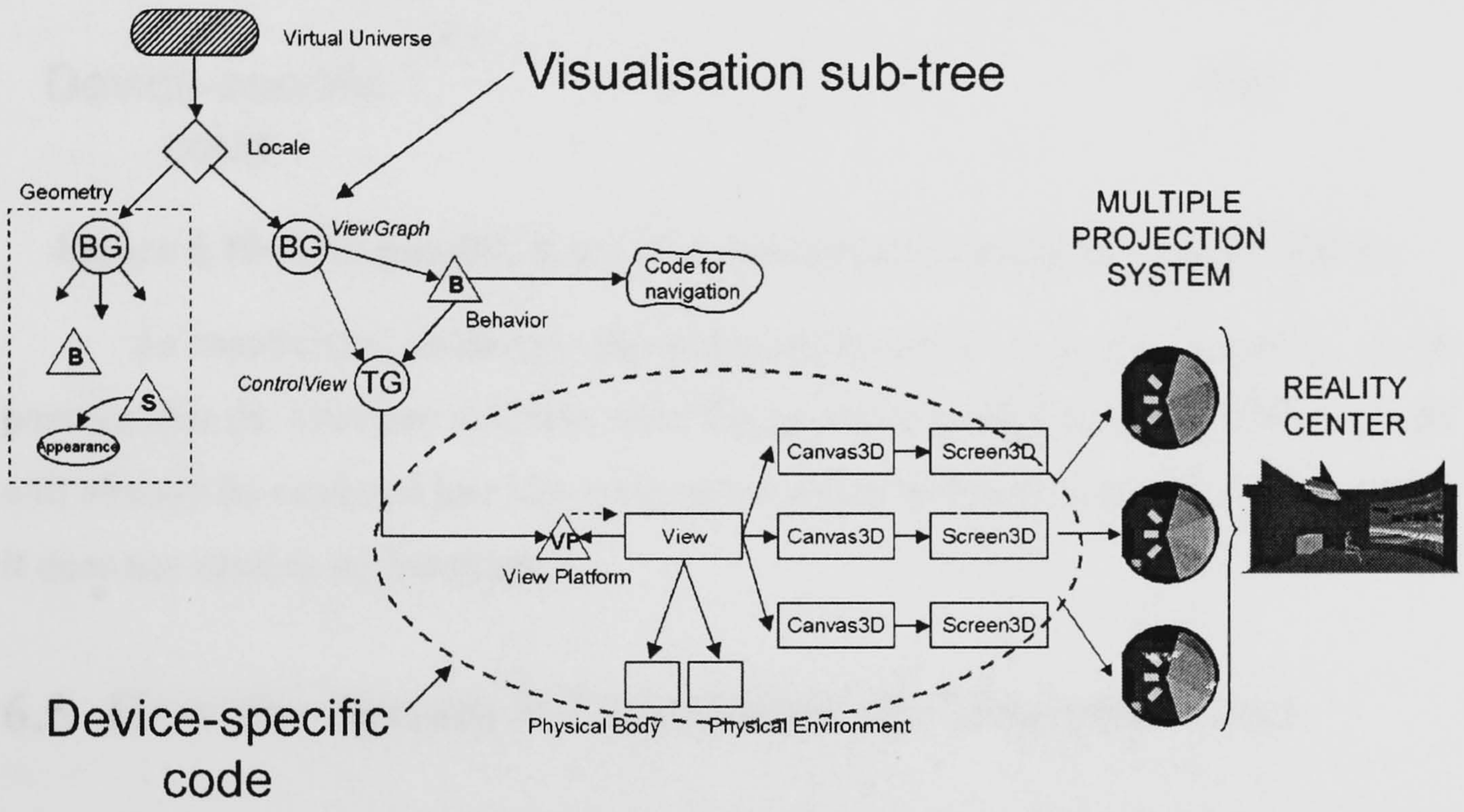


Figure 6.9: The ideal DAG for a multiple-screen mode

6.4.3.3 Flexible Extension to Other Devices

The modular approach shown in the previous paragraphs gives a great deal of flexibility to the Visualisation Core. The system can be easily adapted to the requirements of other specific set-ups through the manipulation of a limited part of the DAG.

Consequently, devices such as CAVEs, Head-Mounted Displays or Virtual Tables could be supported without interfering with the general structure of the system. Each new device would only require the re-writing of the specific section of the DAG containing the device-specific code to allow the renderer to generate the correct image, (as shown in Figure 6.10).

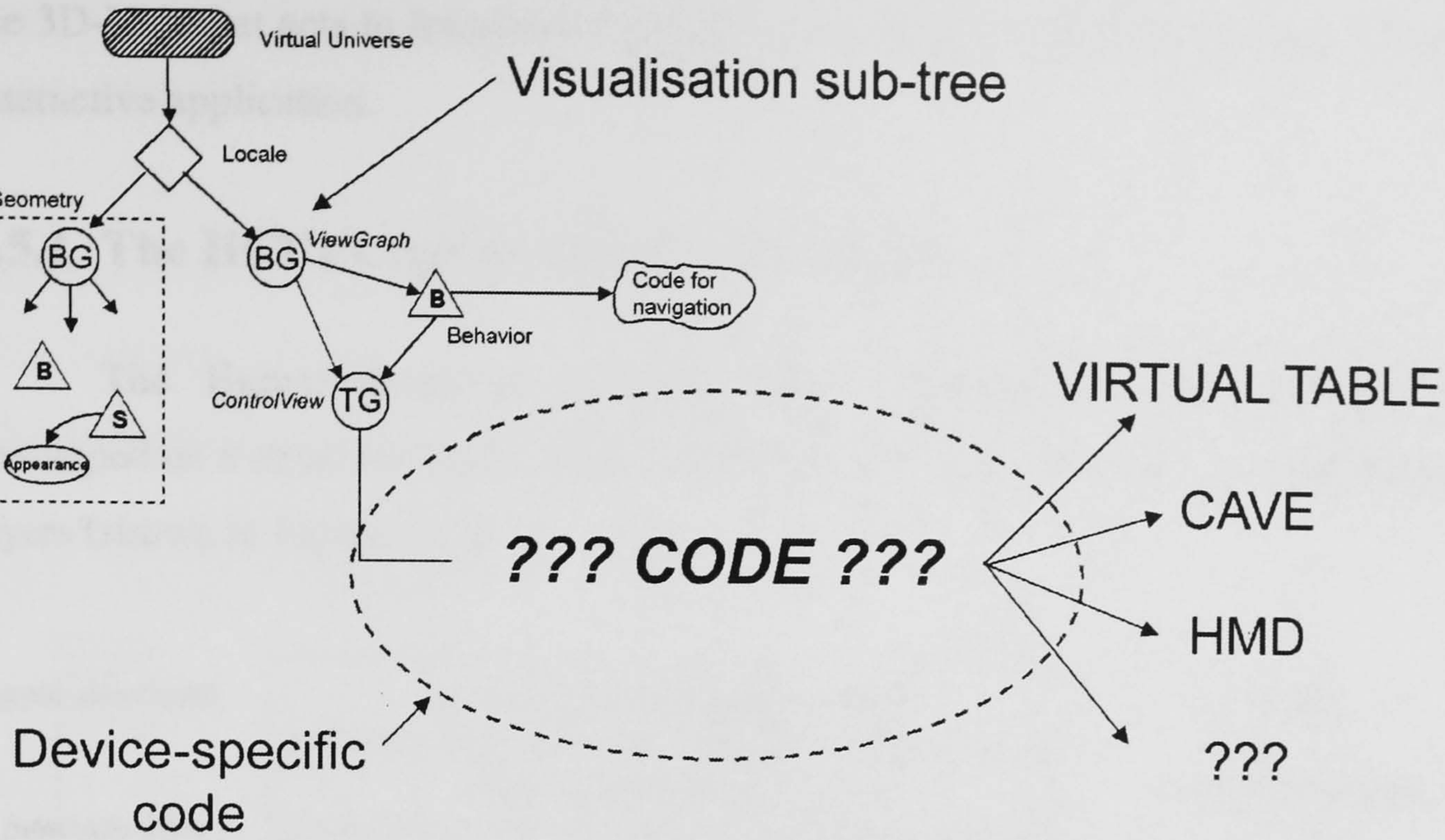


Figure 6.10: The possible DAG of a customised visualisation configuration

As mentioned, ultimately the programmer needs to provide only the initial position that the interface will take once the system is loaded, since the 3D-GUI itself will already be rendered just like every other object in the environment and therefore it does not need to be rewritten.

6.5 How the System is Controlled: the Interface Core

The previous sections have illustrated how the visualisation sub-tree contains the nodes representing the visualisation devices as well as the objects necessary to create the 3D-GUI. The next sections will describe the logic behind the interaction with the 3D-GUI which is implemented through an independent part of JCAD-VR called the Interface Core.

The Interface Core is the section of the system that is responsible for the interaction between the user and the system. Its role is to transform the user's input activity into a sequence of meaningful actions or commands. This is accomplished through a set of complex mechanisms that interact with the other sub-systems dealing with the visualisation, the geometry of the world and the connection with the network. Therefore, the Interface Core represents the bond between the modules of

the 3D-Unit that acts to transform a collection of independent modules into a single interactive application.

6.5.1 The HCI's Logical Abstraction Layers

The Human-Computer Interface (HCI) implemented in JCAD-VR is developed on a structure based, from the logical point of view, on several abstraction layers (shown in Figure 6.11).

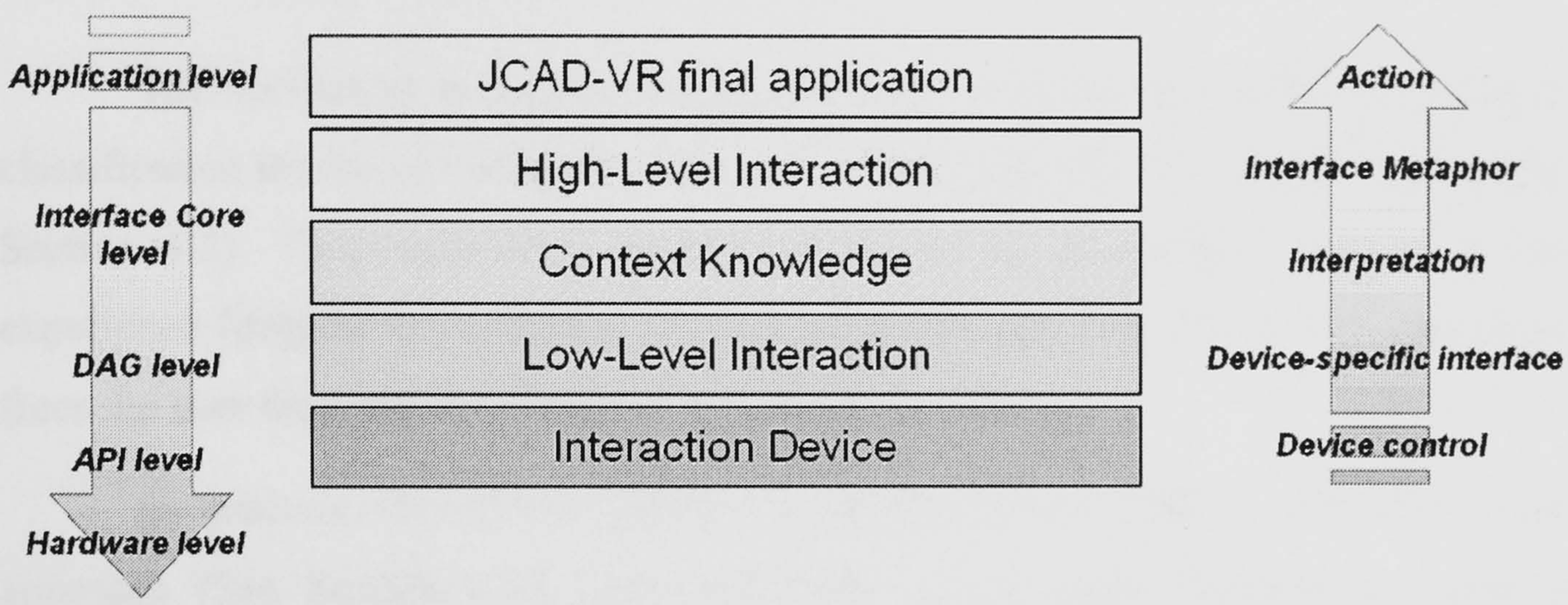


Figure 6.11: The levels of interaction in JCAD-VR

The structure is a development of the classification in three levels proposed by Barrilleaux (2001) (See Section 4.3). In Figure 6.11, Barrilleaux's organization into *primal*, *analytical* and *virtual levels* is replaced by a five level architecture logically stacked on top of the device adopted.

Starting from the hardware level, moving over several software generalizations, the abstraction increases until the uppermost level is reached, which contains the JCAD-VR application.

The previous sections have already illustrated JCAD-VR's HCI's lower level abstraction, based on a number of classes representing the devices adopted in the VR settings. Through the use of the tools available from the API, the Visualisation Core abstracts JCAD-VR's set-up and gives a flexible DAG.

However, at this level the system is not aware of the interaction techniques or metaphors adopted by the HCI. However, at this level, JCAD-VR does have control

upon the interaction techniques used for the *navigation*, *interaction and access* (See Section 4.5).

Through its knowledge of the state of the virtual world, the Interface Core implements the higher layer abstractions of the interface (illustrated in Figure 6.11). This provides the proper tools to interpret the user's commands into meaningful actions, based on the metaphor adopted by the system.

6.5.2 The HCI Interaction Techniques

The interaction techniques adopted in JCAD-VR can be divided according to classification into three categories: *navigation*, *manipulation* and *system control* (See Section 4.5). They have been chosen to deliver an interaction close to a real experience (despite the adoption of a Desktop-VR solution) that at the same time frees the user from the strict rules of the physical world.

In practice, JCAD-VR follows what Barrilleaux (2001) calls the *mixed* approach (See Section 4.3). The constraints to the user's freedom are chosen according to the nature of the application, which has been designed to deliver an environment tailored to architects working in the early stages of the design process.

6.5.2.1 Navigation

JCAD-VR, according to Barrilleaux's (2001) classification in terms of personae, is characterised by a *first person* navigation approach (See Section 4.5.1.1) where the user directly controls the point of view.

This approach provides a higher level of presence compared to *second* and *third person* navigation. Due to the choice of a Desktop-VR or semi-immersive environment set-up it does not have the drawbacks typical of *first person* immersive environments previously reported by Bolas (1994) (See Section 4.5.1.1).

In terms of the user's degree of freedom, JCAD-VR encourages a great deal of free expression when interacting with the environment. The system has not been developed as a simulator (See Section 4.3) but nevertheless it promotes quick navigation through the virtual world, regardless of walls, slabs or other objects. This is the intention of the application as is this way it leaves the architect free to reach an

advantageous point of view regardless of the law of physics, since neither gravity nor *collision detection* is implemented.

Nevertheless, to avoid causing the user any potential confusion by unrealistic degrees of freedom, navigation has been constrained slightly by a few specific rules, as suggested by several authors (Bolas, 1994; Hanson et al., 1997).

Therefore the user, although free to navigate the virtual world, in JCAD-VR is provided with a limited subset of movements. The approach followed tries to replicate a real life experience where the user's movements are mainly limited to translations in a two-dimensional horizontal plane and to rotation around the vertical axis. In JCAD-VR the user is also able to move on a vertical plane but this action has been totally dissociated from the act of walking by the user having to explicitly select the proper command.

For the convenience of the user, it is also possible for them to tilt their virtual head on a horizontal axis. However, rotation of the virtual head does not affect the direction of their movement since the user will always move over a horizontal plane. In practice when using JCAD-VR the vector representing the user's movements always lies on a horizontal plane regardless of the direction of the LAD-DV (See Section 4.4.2).

From the implementation point of view the navigation effect is achieved through the manipulation of the two TransformGroups (illustrated in Figure 6.12, which is a detailed view of the graph in Figure 6.5). For the system to be able to move the user's point of view in a *first person* mode it first retrieves the direction of sight of the virtual head through the LAD (look-at direction) vector (See Section 4.4.2) with its position, and then it adds the relative movement necessary to go to the new position.

As illustrated JCAD-VR adopts a constrained navigation mode to replicate a real-life experience and although it lets the user tilt the virtual head, it always constrains translations to a horizontal plane. To ensure that the user always moves on a horizontal plane, JCAD-VR uses two cascaded TransformGroups. The lower one is used to tilt the user's virtual head while the upper one is independently used to translate the POV onto a horizontal plane and to rotate it around a vertical axis.

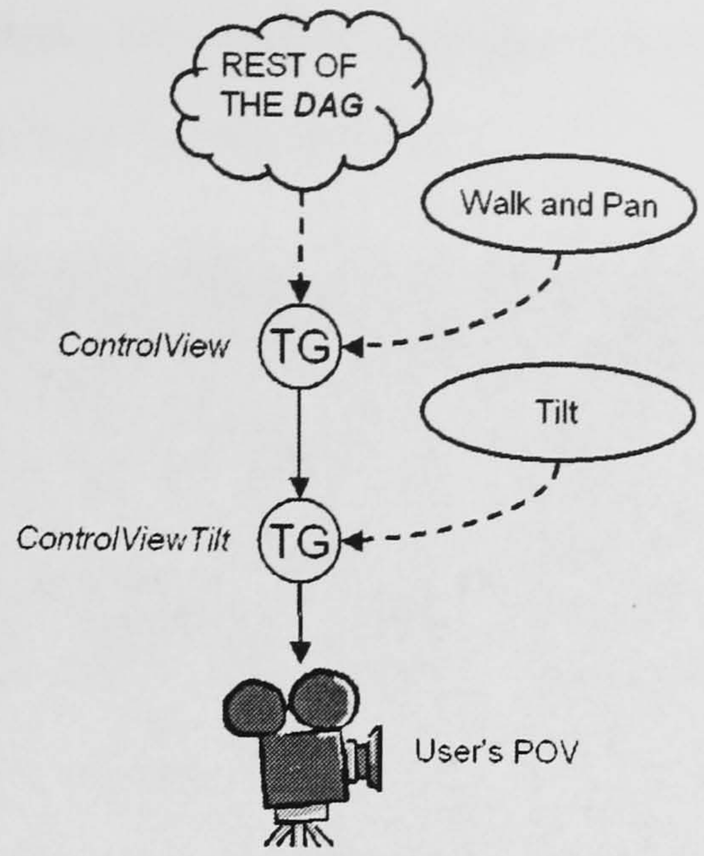


Figure 6.12: Details of the navigation process

This approaches has an advantage in that it is easier to deploy. It also avoids having number of conversions for every frame from the local to the absolute coordinate system (and vice versa) that would be necessary if a single TransformGroup were used.

6.5.2.2 Manipulation

In terms of *control personae*, JCAD-VR adopts a *third person* approach (See Section 4.5.2.1). A set of 3D-widgets which are shown near the objects once they are selected, gives access to the manipulation functions.

This approach combines the advantages of the *second person* approach which promotes a natural experience, with the user naturally manipulating objects in the virtual world but at the same time it forces the user to explicit about the action of manipulation. To emphasise the action, and at the same time to assist the user during manipulation, JCAD-VR makes use of three arrows to define the local coordinate system of the object (See Figure 6.13). The arrows appear beside the object once it is selected together with the nearby icon used to choose the type of action.

The adoption of a *third person* manipulation technique has two main advantages. First it constrains the manipulation to one direction at a time, therefore making manipulation easier when a traditional mouse is used, and at the same time it provides the relevant feedback on the type of manipulation being done. Therefore,

3D-widgets are simultaneously the means to manipulate an object and the feedback mechanism that informs the user on the action.

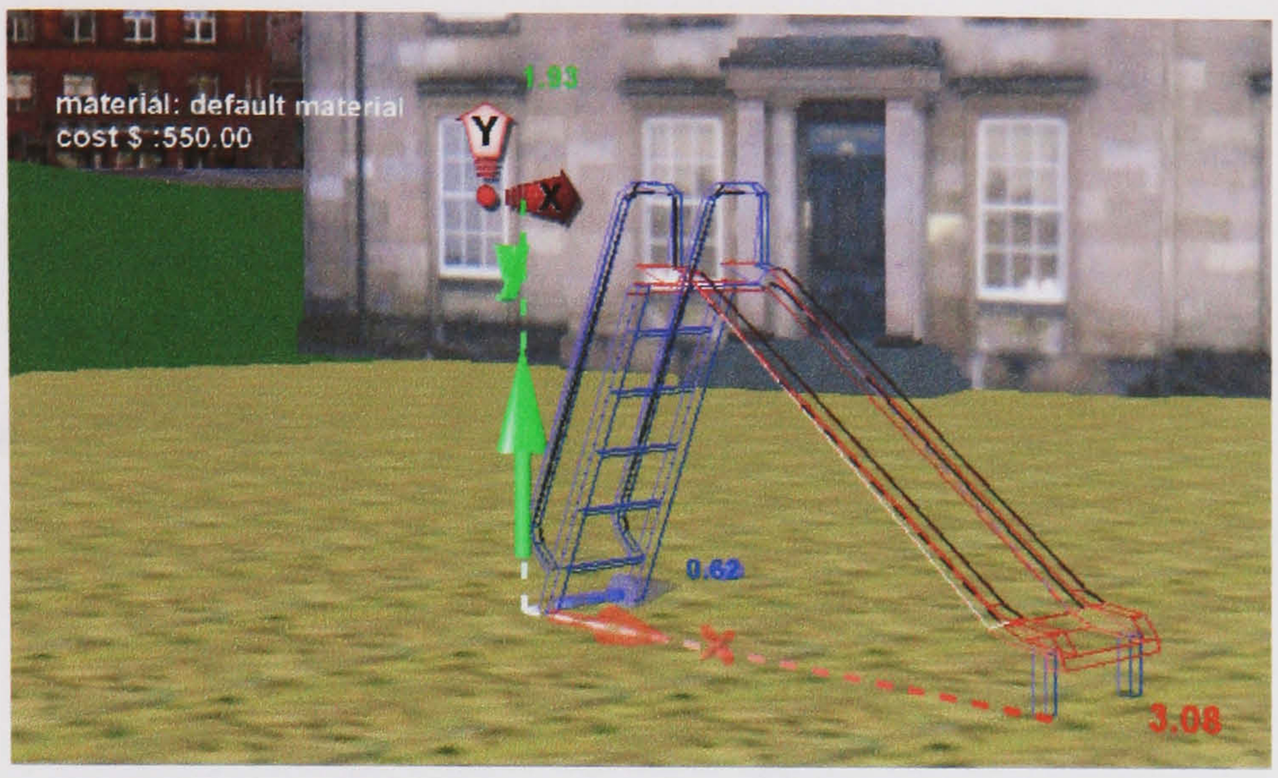


Figure 6.13: A detail of the 3D-widgets used for the manipulation of objects

The user selects the type of modification they want by clicking on the relevant icon. In this way the user activates the translation, rotation or scaling modes in turn. Finally the user selects the arrow defining the direction and drags the pointer until the desired effect is reached.

Concurrently the system displays the most important information about the object in a panel beside the object. The information in the panel is updated in real time, as is the general dimensions of the x, y and z directions, through three 3D-labels placed at the extremities of the objects. For the convenience of the user, both the panel and the labels are always facing their point of view.

This approach is very visual, it does not require any typing therefore it does not force the architect to think in terms of mathematical entities. In addition it does not require any specific hardware, since commands are interpreted through mouse movements and clicks, and therefore it can be used with standard workstations. The adoption of more complex hardware, such as 6-DOF mice, could be implemented on top of the existing interaction technique, since the latter is hardware-independent.

From the implementation point of view the manipulation of objects is achieved by accessing the section of the DAG where the object is placed. Figure 6.14 illustrates the relevant portion of the DAG where the geometry of the object is placed. Once the object is selected, the relevant 3D-widget appears through the

SwitchInterface Switch, a special type of class that allows run-time activation/deactivation of its children nodes.

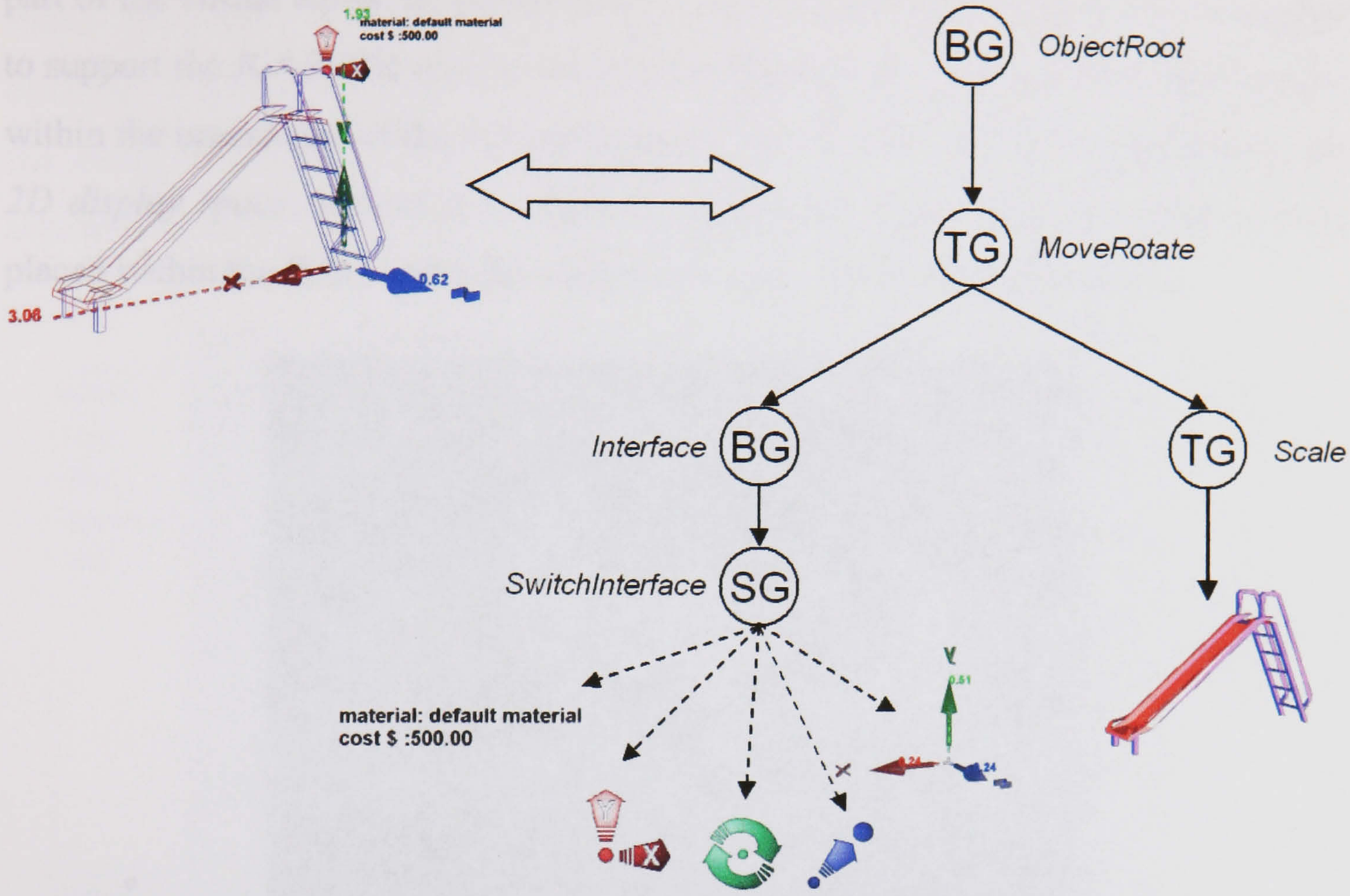


Figure 6.14: A schematic view of the DAG of a 3D object in JCAD-VR

Once the user has decided on the type of modification they want to apply to the object, for example translation, rotation or scale then he/she selects one of the arrows and makes the change. The system interprets the action and executes the command, altering the values of the matrixes contained inside the two TransformGroups (shown in Figure 6.14).

For each object, the implementation of rotation or translation is separated from the scaling process since two TransformGroups are used. Specifically the *Scale* TransformGroup is placed on the section of the tree that contains the geometry of the object so that the modification of its values does not affect the 3D-widgets. This choice causes the 3D-widgets to rotate or translate every time the object is modified, and at the same time it ensures that the 3D-widgets always remain at the same size. Particularly, it prevents them from being altered every time the object is scaled, thus avoiding an aesthetically unpleasant distortion of the 3D-widgets after a non-uniform scaling operation.

6.5.2.3 System control

One of the main goals of the system was the development of an interface as part of the virtual world. In answer to this requirement a 3D-GUI has been developed to support the JCAD-VR *system control* (See Figure 6.15). The 3D-GUI does not live within the boundaries of the *2D screen space* (See Section 4.4) or its counterpart, the *2D display space*. Instead it is made of objects that live in the *3D virtual world*, placed within the DAG under the visualisation sub-tree (See Section 6.4.1).

Figure 6.15: A detail of a 3D-menu being moved

3D-GUIs, as already pointed out in Section 4.5.3, can be more difficult for people to use if compared to traditional WIMP applications. For this reason JCAD-VR follows a hybrid approach, considered by several authors as the most effective choice (Bowman et al., 2001), as it can improve the three dimensional nature of the virtual environment yet retain the usability of two dimensional GUIs.

Therefore in JCAD-VR traditional WIMP elements are turned into their three-dimensional counterparts. Users can then access functions and at the same time handle menus and icons, in the same way that they did with other objects in the environment. For the convenience of the user 3D-Menus can also be scaled, rotated or moved (shown in Figure 6.15).

6.5.3 The functions of the HCI

As soon as the user starts JCAD-VR four different 3D-menus appear on the screen placed at the corners of the rendered area. Each menu is opened or closed by clicking on the first icon which illustrates, in an iconographic way, the functions of the menu. Through the main menus the user can control four groups of features: creation of geometries, navigation, communication and utility functions.

Figure 6.16: The 3D-menus

6.5.3.1 The Creation Menu

The *creation menu* gives access to a number of geometries, from simple geometrical primitives, to AEC-specific objects like slabs, walls, windows, doors, stairs and an expandable library of objects.

In the case of windows, doors, stairs, primitive shapes and the object library, the creation menu gives access to further 3D-panels (shown in Figure 6.17).

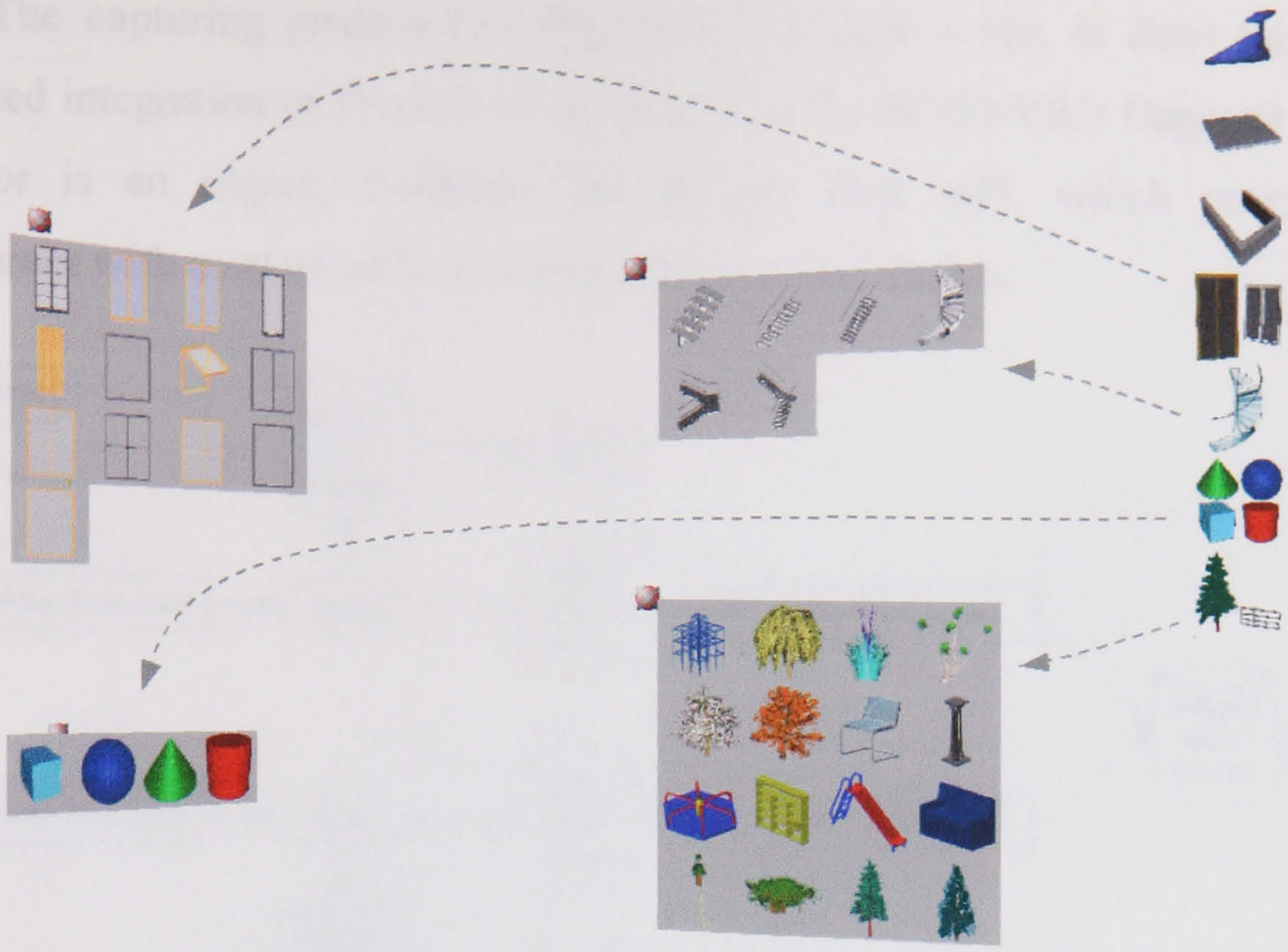


Figure 6.17: The 3D-panels activated through the creation menu

Like any other object, each panel can be placed in the virtual world according to the need of the user, and when closed its position will be stored, to be replaced once reactivated.

6.5.3.2 The Navigation Menu

The *navigation menu* allows the user to select between walk, pan and tilt movements. For their convenience there is an icon to bring the user back to the origin of the virtual world. This function would be useful if the user got lost in the virtual environment.

6.5.3.3 The Communication Menu

The *communication menu* holds the panels where the chat text is rendered as well as images from the videoconference module (See Figure 6.16). For the sake of visibility, the chat panel shows only the last messages sent or received by the user while the full content of the chat is accessible through the *utility menu*.

In the *communication menu* special attention has been given to the videoconference panel. Its development has required close integration between objects belonging to two different APIs: the Java 3D™ and the Java Media Framework (JMF) (Sun Microsystems, Inc., 2002d).

The capturing process (as illustrated in Figure 6.18), is done through an articulated integration of Processors developed in the JCAD-VR's Capturer class. A Processor is an object, available through the JMF API, which provides the programmer with a set of tools to manipulate an input stream.

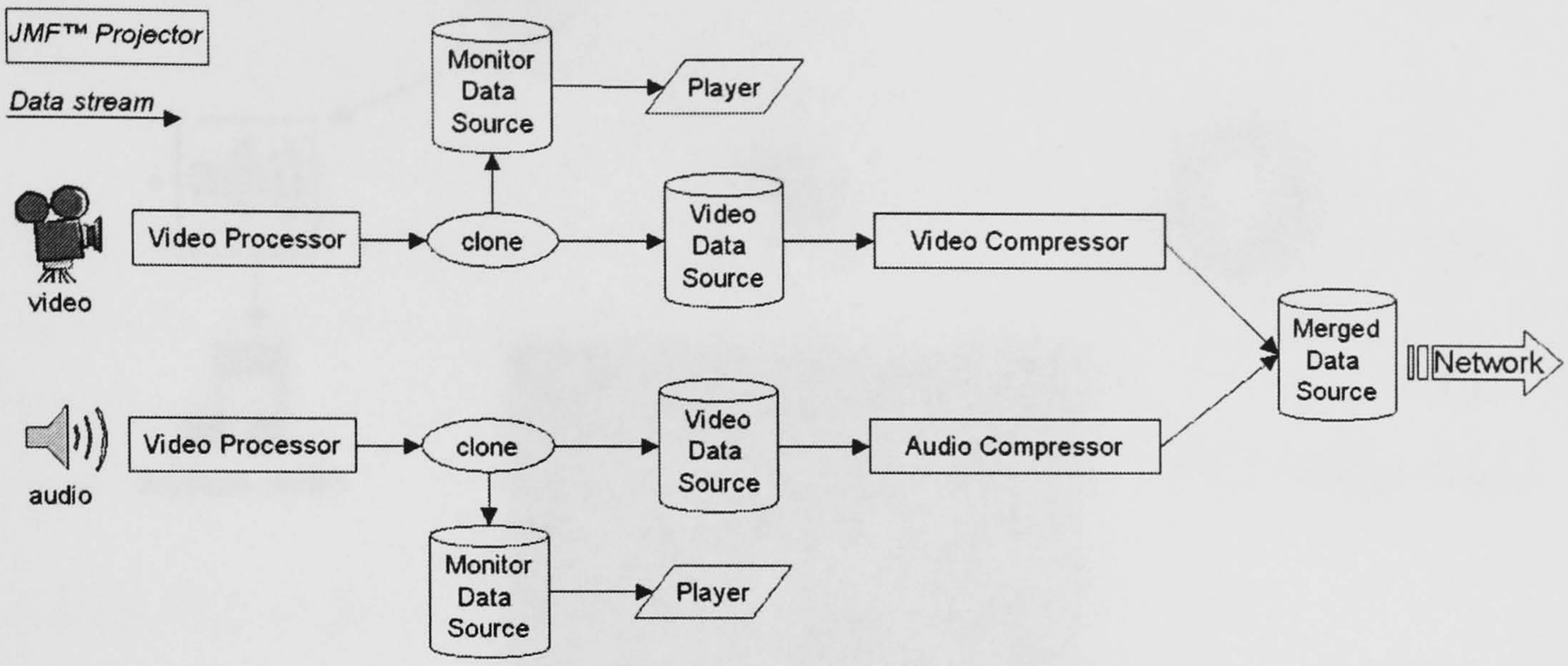


Figure 6.18: The videoconference capturing process

As illustrated in Figure 6.18, the data stream coming from the video camera and the microphone is sent to the first processor that converts it into a DataStream object. The system creates a copy of each DataStream and sends it to two JMF Players, to playback audio and video respectively from the user's set-up. This is done for the convenience of the user who is able to monitor the content being streamed to the network.

The two original DataStreams are then used as a data source for the other two Processors, which compress the video and audio respectively into more efficient forms. This operation is done to reduce the load of data broadcast throughout the network.

Before sending the two streams to the network the system merges the compressed audio and video into a single multiplexed data stream. This is finally sent through the server in the form of RTP (Real Transform Protocol) streams.

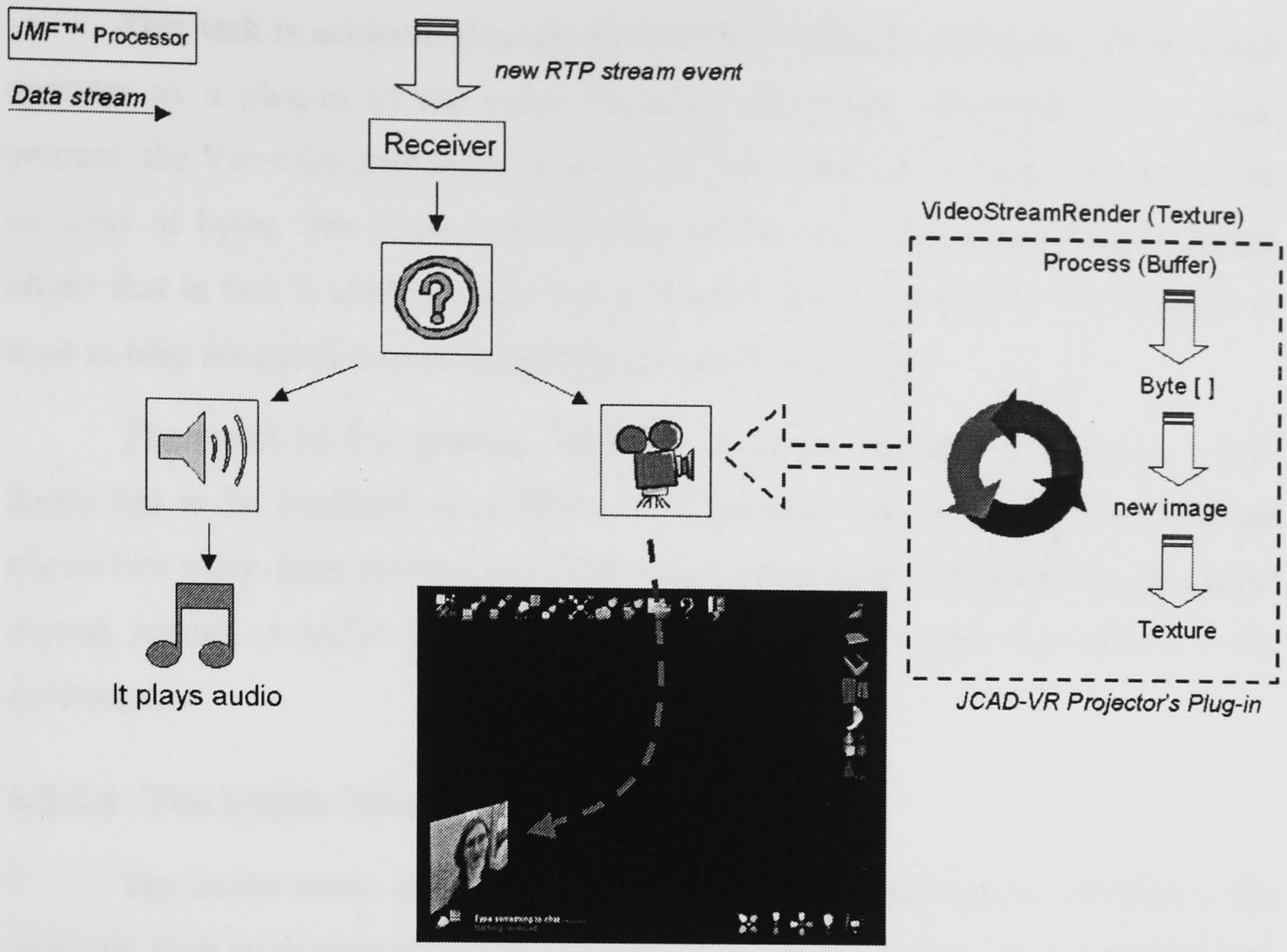


Figure 6.19: The videoconference playback process

At the same time the system listens to the RTP streams broadcasted by other users present in the virtual environment through the JCAD-VR's Receiver class (See Figure 6.19). The first time an RTP stream belonging to a new user is identified, the Receiver class sends a message to the 3D-GUI. This in turn activates a new 3D-panel next to the existing ones, onto which the content of the videoconference will be rendered. In addition a new label is created, and it is displayed on top of the video panel, illustrating the login name of the transmitter.

As soon as the new 3D-panel is created the content of the new RTP stream is passed to a Processor. This first Processor which acts as a de-multiplexer, analyses the content of the streams and also separates the video from the audio channel thus generating two new data streams.

The audio data stream is directly used as the input of a Processor that decompresses the content and then plays it through the loudspeakers of the system. The video meanwhile is sent to another Processor that converts the data stream into a texture.

This task is achieved through JCAD-VR's VideoStreamRender class, which operates as a plug-in of the video Processor previously mentioned. In a cyclic process, the VideoStreamRender receives the video data stream and converts it into an array of bytes. The content of the array is then utilized to create a new Image object that in turn is used to create a new Texture object. Finally the new Texture is used to map the panel created at the beginning of the process.

The result of this process, which is cyclically repeated every time a new frame has to be rendered, is a 3D-panel, integrated into the virtual world which shows live video from another user. The panel, being truly three-dimensional can be moved, rotated or scaled as desired by the user similarly to any other object of the environment.

6.5.3.4 The Utility Menu

The *utility menu* gives access to various sets of commands. Through it the user can: save or retrieve the content of the virtual world, activate the communication panel including the whiteboard and the full text of the chat, delete the content of the virtual world, take single screen shots or make continuous recordings of the virtual world, access advanced settings and finally they can quit the application.

Some of the functions, such as the load/save facility and the visual settings panel (See Figure 6.20), due to their special and limited use, are provided through standard WIMP elements.

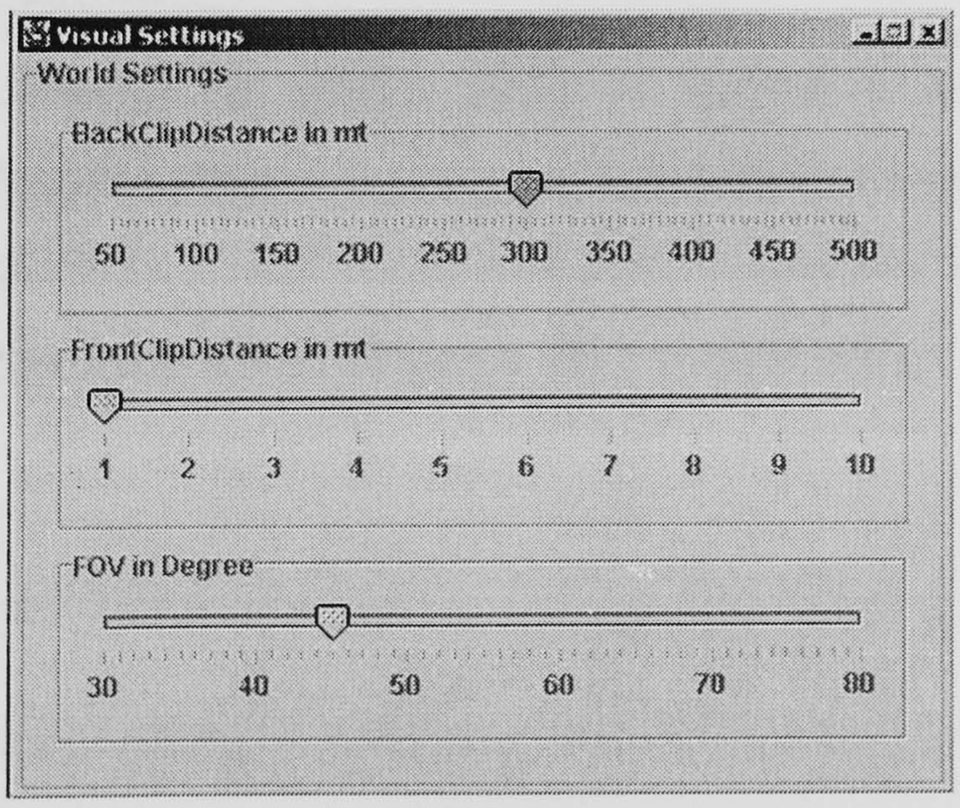


Figure 6.20: The visual settings panel

6.5.3.5 The Material Chooser

The *material chooser* is a simple tool which is included in JCAD-VR to allow the quick choice of textures. The user can activate the material chooser by selecting the relevant icon that appears beside the selected objects.

The idea behind the development of the material chooser is to provide the user with a quick way to make a choice of material through the simple use of textures rather than using other iconic forms. Therefore the material chooser should not be considered as an advanced material editor helping the user by defining the final texture map of an object, but it should be seen as a simple and quick tool that supports the user's ideas in a visual way.

In an ideal scenario the architect would be able to apply a concrete looking texture to a surface to illustrate that the final material will be concrete and not red bricks for example. The map adopted is not the final texture however but it is rather a visual aid to show that a certain object should be made of a certain type of material.

As noted, JCAD-VR is neither a CAAD application nor a modelling package and for this reason functions involved in setting texture are kept to a minimum. Therefore texture maps are applied automatically at a default size and no means is provided for scaling or arranging them.

From the implementation point of view, the material chooser is essentially based on another tree-like structure. The user starts from a root level, where a set of 3D-folders appears, generically labelled wood, metal, etc. As shown in Figure 6.21, once the user has selected the proper folder its content is shown through cubes textured with the relevant material. If the other sub folders are available the user can continue browsing the library until the lowest level is reached. At any time the user can precede up the tree structure clicking on the top-right folder.

Figure 6.21: A view of the material chooser

The user can also customise the content of the texture library simply. The tree structure of the material library is created through the analysis of the content of the "materialsLibrary" a sub-directory of the folder where JCAD-VR is installed. The labels used to name the cubes and the 3D-folders are automatically derived from their filename counterparts. The system loads all the images in JPEG format and uses their filenames to create the elements of the material library. The naming convention has been implemented so that the system reads the filenames, removes the ".jpg" extension and turns underscore symbols into spaces. The changes to content or sub-folders structure of the "materialsLibrary" directory, take effect the next time the system is loaded.

The texture included in the current release of the system are simple and photo-derived maps. It is possible through the mechanism described however, to substitute the content of this library with a more abstract set of images. This would prevent users from considering the textures selected from the library as final choices, rather than just as suggestions.

The ability to customise the library currently has a major limitation. During a collaborative session as soon as material was selected an event was dispatched throughout the network to every other user. The network message carries an ID number created during the loading of the library. If two or more users have different

libraries the IDs generated by the respective systems might not coincide and this can lead to misinterpretation during transmission and ultimately cause the different systems to show different textures on the same object. Therefore, to ensure consistency across the network, every user must have the same material library. Precisely, the content and the structure of the "materialsLibrary" folder and its sub-folders must be identical for every user in the environment.

This limitation could be overcome through the development of a different algorithm that generates the material IDs in a more effective manner and, at the same time, provides the means for the transmission of missing textures to other users. However so far this function has not been developed.

6.5.4 Implementation of the HCI

As illustrated throughout this chapter JCAD-VR's 3D-GUI is based on the fundamental assumption that the interface becomes part of the virtual world. As illustrated this approach has two different advantages.

From the theoretical point of view the entire system is "placed" within the virtual world. Objects, menus and icons are treated in the same way. The 3D-GUI becomes part of the world.

From the practical point of view this gives a high degree of flexibility to the configuration of the system. In the case of a new visualisation device being used, the HCI does not need to be re-programmed since it is automatically rendered together with the rest of the virtual world. For the convenience of the user, the developer has to configure the relevant classes (as described in Section 6.4.3) and decide where the 3D-GUI will be initially placed inside the environment once the system is loaded.

New pointing devices can be implemented taking advantage of the pre-existing structure of the 3D-GUI as a means for interaction with the system, since the pointing device is independent from the metaphors used.

The features described above are achieved through the adoption of a complex design pattern illustrated in the following sections.

6.5.4.1 Everything is a "Pickable" Node

JCAD-VR can treat the 3D-GUI like one of the objects of the virtual world thanks to an articulated mechanism based on one of the major features of Object Oriented (OO) programming languages: the *inheritance* mechanism.

In the *inheritance* process a class *extends* its *base* or *super* object, thus automatically importing its features. The inheriting class can also implement further routines, or methods, thus extending the function of its *super* class.

In the specific case of Java™ this approach is further emphasised since every objects extends another class. The *inheritance* process can be made explicit through *extension* of a specific object or, more often, it is implicit since in Java™ every object that does not extend a class automatically extends the primal *Object* class.

The process of *inheritance* contributes towards the separation, within the programming code, of the functions of an application from the details of the implementation. This is achieved through an approach that promotes the reuse of existing code.

In a real world analogy (See Figure 6.22) the programmer, could for instance develop a class called *Vehicle*, with two routines or *methods*, named *switchOn* and *switchOff*. Through the *extension* process a new class called *Car* could *inherit* the features of a *Vehicle*. This way every *Car* object would automatically be provided with the two methods *switchOn* and *switchOff*; however, at the same time, *Car* could also include other methods such as *turnOnLights* and *turnOffLights*. Likewise, a new class called *Convertible* could extend the *Car* class thus inheriting all its methods, including those inherited from *Vehicle* and those specifically developed inside *Car*. Additionally *Convertible* could include more specific methods. This analogy could be extended to include objects like *Truck* or *Motorbike*, which would inherit their properties directly from *Vehicle*, or objects like *Saloon* or *Hatchback, which* would inherit properties from *Car*.

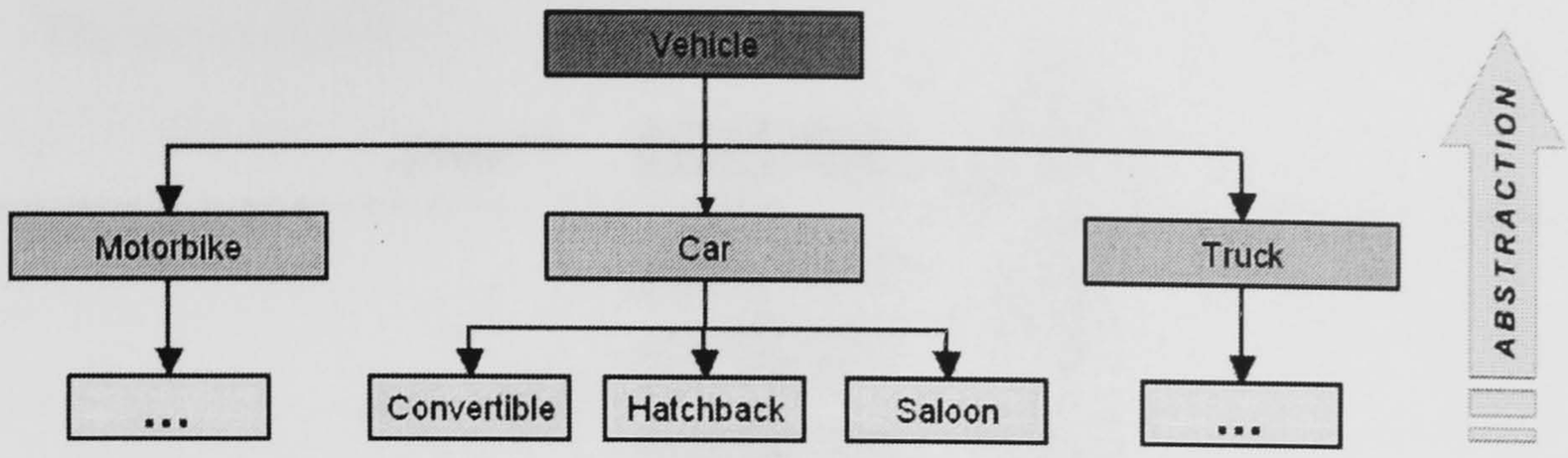


Figure 6.22: The inheritance mechanism

It is noticeable from this analogy that the inheritance mechanism promotes the reuse of existing code through the automatic extension of functions developed in previously developed classes.

However, developing a code that makes clever use of inheritance has another, perhaps even more important, advantage. Referring to the previous analogy once more, it could be possible for the programmer to go back to the *Vehicle* class, after all the aforementioned classes were developed, and change one of its properties by for instance including a new method: *brake*.

As a consequence of the inheritance structure all the objects extending the *Vehicle* class could access the new function without the need for the programmer to rewrite the *brake* feature for each object. More importantly, it would be possible to manipulate the mechanisms of the *Vehicle* class by replacing for instance, the internal combustion engine with an electric motor. The change would be automatically applied to all the other objects. Finally, if specific circumstances required it, the programmer could replace the electric motor of the *Convertible* with a petrol engine, through the *casting* mechanism, independently from the type of engine being used by all the other vehicles.

It is evident that through *inheritance, casting* and *polymorphism*, which is the mechanism that permits further level of customisation of methods extended from a *base* or *super* class, the developer can achieve an extremely high level of flexibility within the code.

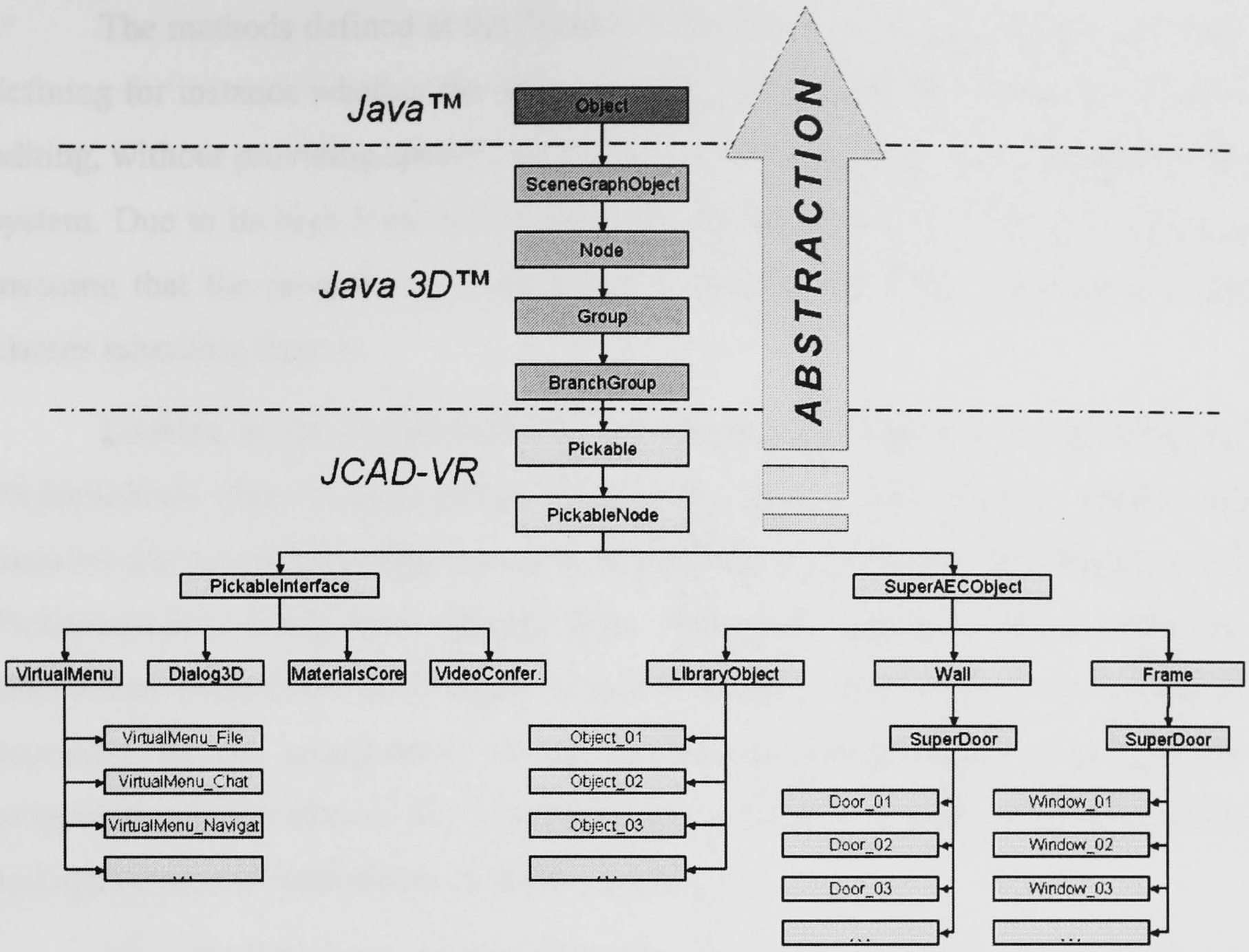


Figure 6.23: The inheritance architecture in JCAD-VR

Adoption of this powerful structure is shown in the structure illustrated in Figure 6.23, which illustrates the inheritance architecture of JCAD-VR. At the top of the diagram there is the previously mentioned *Object* class, the ultimate super class of every object created in Java™. Proceeding down the structure the next four levels of inheritance are characterised by the Java 3D™ API layer. At the top of the JCAD-VR level, there is the *Pickable* class, the *super* class of every other graphic object developed in JCAD-VR. This class is the super class of every "pickable" object in the virtual world, every object that can be selected and manipulated inside the system.

The inheritance from the *Pickable* class allows JCAD-VR to treat the elements of the 3D-GUI as the same as the objects in the virtual world since they all share the same features: they can be "picked". To do so the *Pickable* class provides all the methods necessary for the handling of objects through the *picking mechanism* (described later in Section 6.5.4.2).

The methods defined at the *Pickable* class level are of the most general type, defining for instance whether the object is being edited or if the system has finished editing, without providing specific details of how the object has to be handled by the system. Due to its high-level architecture, the *Pickable* class is defined as *abstract*, meaning that the programmer cannot use it directly but it has to use one of the classes inheriting from it.

Looking at the Figure 6.23 and moving one level down the hierarchy, the *PickableNode* class is encountered. This is the super class of every object, and includes the set of 3D-widgets used to manipulate its geometry. The details of the *PickableNode*'s DAG have already been illustrated in Figure 6.14, where the mechanism behind the 3D-widgets was shown. This class defines the geometry necessary for the management of the 3D-widgets and through its methods the programmer has access to the TransformGroups used to modify the configuration and appearance of each object in the environment.

From the illustration of these first two classes it is evident how JCAD-VR can manipulate elements of the 3D-GUI in the same way as every other object belonging to the virtual world. In fact, through the inheritance mechanism every 3D-menu, 3D-panel, wall or cone etc. can be ultimately treated by the system as a *PickableNode* and therefore manipulated through its 3D-widgets.

Referring again to Figure 6.23 and moving down another level, it is possible to see how the inheritance structure eventually separates the objects belonging to the 3D-GUI from those used to create the geometries, whose detailed implementation will be discussed in the following chapter.

Regarding the implementation of the 3D-GUI, the *PickableInterface* is the super class of every 3D-menu, 3D-panel or 3D-icon present in the world. This class features only one method that is used by the system to activate the object. The main difference between the elements of the 3D-GUI and the other objects is that the former have some form of functionality or command embedded in them that can be invoked by selecting the object with the pointer. In order to take advantage of the flexibility supplied by the inheritance mechanism, the method of the *PickableInterface* class that expresses the command is declared *abstract*.

As with to the case of the *abstract* class, the programmer can declare the existence of a method, without defining its implementation, through an *abstract* method. This mechanism ensures that every element of the 3D-GUI is embedded with a command used to accomplish a certain function, but it leaves the implementation details of a specific function to each class extending *PickableInterface*.

The flexibility and abstraction brought by having several levels of inheritance permits JCAD-VR to treat the same object in different ways according to the context. In this is way an object of the 3D-GUI can be treated as an element of a 3D-menu if the user is selecting it to command an action, or as a simple 3D-object if the user wants to move it for instance.

At the base of every interaction there is the fundamental process of selecting the objects, known in technical literature as the *picking* process. The next section describes the details of the picking process implemented in JCAD-VR.

6.5.4.2 The "Picking" Process

The "picking" process in computer graphics is the mechanism necessary to select objects in the virtual space.

In JCAD-VR a selection process based on simple *ray-casting* or *laser pointer* techniques (Bowman et al., 2001; Forsberg et al., 1996) has been implemented. The technique was chosen in preference to the ones described in Section 4.5.2.2, since it is very effective, and yet more simple to develop as a systems dealing only with solid geometries rather then lines or points.

In the *ray-casting* technique, a ray passing from both the user's virtual eye and the mouse pointer, is then projected into the environment and the object closest to the user which is intersected by the ray is chosen as the "picked" one.

This method is normally supported at the Java 3D™ API level and does not require further implementations. Unfortunately a bug in the release of Java™ used to implement JCAD-VR (See Appendix D) prevented the standard picking mechanism being used when JCAD-VR was running in multiple-screen mode. When more than one screen was attached to the system the mechanism did not correctly return

information on which canvas the pointer was located at the time the picking action was requested. This information is fundamental to create the ray cast into the scene.

As a consequence the relevant routines were rewritten in JCAD-VR's PickingEngine class. The PickingEngine adopts a two-stage algorithm. During the first stage the system checks for intersections between the correct ray, generated compatibly with the multi-screen configuration, and the *bounding shape*, an invisible volume defining the rough dimension of every object in the virtual world. The information on the objects whose bounding shape has been intersected by the ray is stored in an ordered array where the element n. 0 is the closest to the user's virtual eye.

In order to increase the efficiency of the searching algorithm JCAD-VR's entire DAG has been structured so that all the objects inheriting from *Pickable* belong to a specific branch. In this way the search for the pickable node is constrained to a limited section of the DAG (illustrated in Figure 6.24).

During the second stage, the system scans the objects present in the array mentioned above, and checks for any intersections between the geometry of the object and the ray. The first object whose geometry intersects the ray is chosen as the one to be picked.

The approach described is relatively fast but not always precise. The picking process is imperfect in the unlikely circumstance of two objects being in the trajectory of the picking ray and the bounding box of the farther one intersecting the picking ray before the bounds of the closest object.

Ideally, during the second stage, the system should check for intersections between the picking ray and the geometries selected in the first step. It should then calculate each intersection point and eventually choose the geometry whose intersection point is the closest to the user's virtual eye. Nevertheless, due to the infrequency of the unlucky circumstances described, the simpler approach has been taken because it performs much better in terms of efficiency and speed.

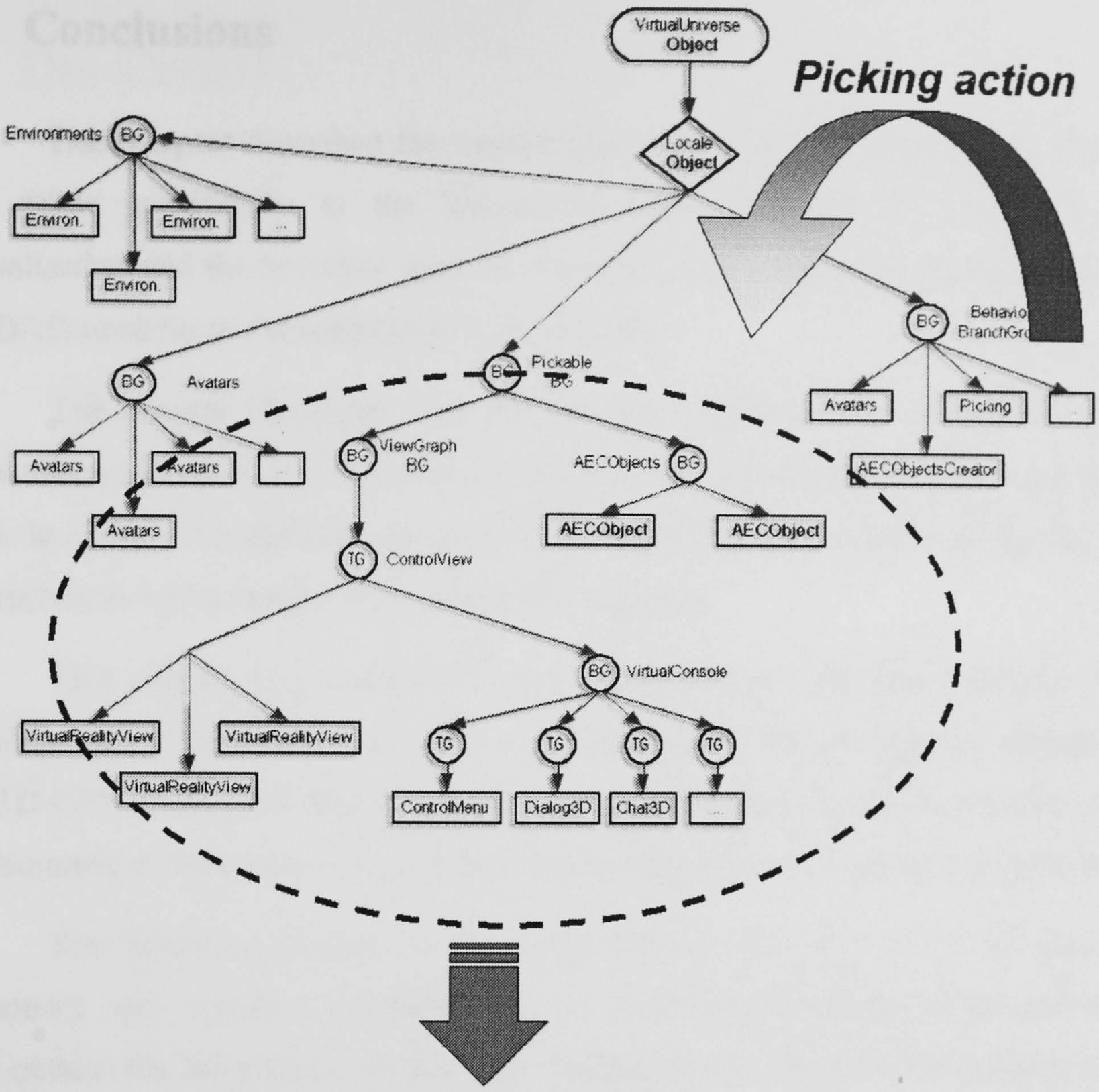


Figure 6.24: The BranchGroup where all the *Pickable* objects are located

The ray casting mechanism is not only used when selection of objects is required. It is also fundamental for when the user needs to create objects in the space. In JCAD-VR the user creates objects in a very visual way, first by selecting the type of geometry to be created from the creation 3D-menu and then through drags and clicks of the mouse (See Section 7.3.2). Since a two dimensional device such as a mouse is used to localise the exact point in the space the system forces the user to define points as the intersection of a planar surface and a ray passing from their virtual eye and the pointer.

6.6 Conclusions

This chapter described the implementation of the two cores of the 3D-Unit that relate specifically to the Human-Computer Interface of JCAD-VR: the Visualisation and the Interface core. This has been illustrated through description of the DAG used for the development of JCAD-VR.

The chapter illustrated how the flexible architecture of JCAD-VR would allow the system to be expanded over a range of devices with minimal impact on the code. In addition the different parts of the 3D-GUI have been described together with the technical issues tackled during their development.

This chapter also outlined the special inheritance structure followed in the development of the system leading to a mechanism that can manipulate elements of the 3D-GUI in the same way as normal objects. Details of the picking process, which is ultimately at the root of every form of interaction with the system, was provided.

The following chapter describes the third an last core of the 3D-Unit, the Geometry core, which is responsible for the creation and editing of geometries. It will outline the inheritance architecture responsible for the different features of the geometries and the details of the internal mechanism developed to create and edit the geometry.

7 The Content of the World: the Geometry Core

7.1 Introduction

The previous chapter described the implementation of the two cores responsible for the Human-Computer Interface (HCI) of JCAD-VR: the Visualisation Core and the Interface Core. This chapter illustrates the details of the third and last part of the 3D-Unit: the Geometry Core, the section of the client application that deals with the content of the virtual world.

As already mentioned in the previous chapters, JCAD-VR is a design tool especially designed to assist architects in the early stages of the design process. During this stage traditional Computer Aided Design (CAD) systems do not provide the user with the proper tools. They are often specifically for the engineering stage and consequently their use is inappropriate in earlier stages of the design process. In fact CAD applications usually bind the designer's freedom through the imposition of a formal and constrained syntax. This forces the user to think in terms of mathematical values rather than letting them experiment quickly with a variety of different solutions.

In contrast, the Geometry Core of JCAD-VR has been designed to provide the user with a tool to create simple geometries in a rapid and intuitive way. It therefore provides a tool which creates shapes in a very straightforward and visually sympathetic way using simple mouse commands, through the abstraction of the rigid mathematical representation of traditional CAD systems.

The following sections will describe the details of the Geometry Core. They will illustrate the processes of creation and editing of geometries and explain the internal architecture of the 3D-shapes supported by JCAD-VR.

7.2 Issues Related to Java 3D™

7.2.1 The Relevant Section of the DAG

Figure 7.1 illustrates an overview of the Directed Acyclic Graph (DAG) used in JCAD-VR. Through this it is possible to show the function performed by the Geometry Core.

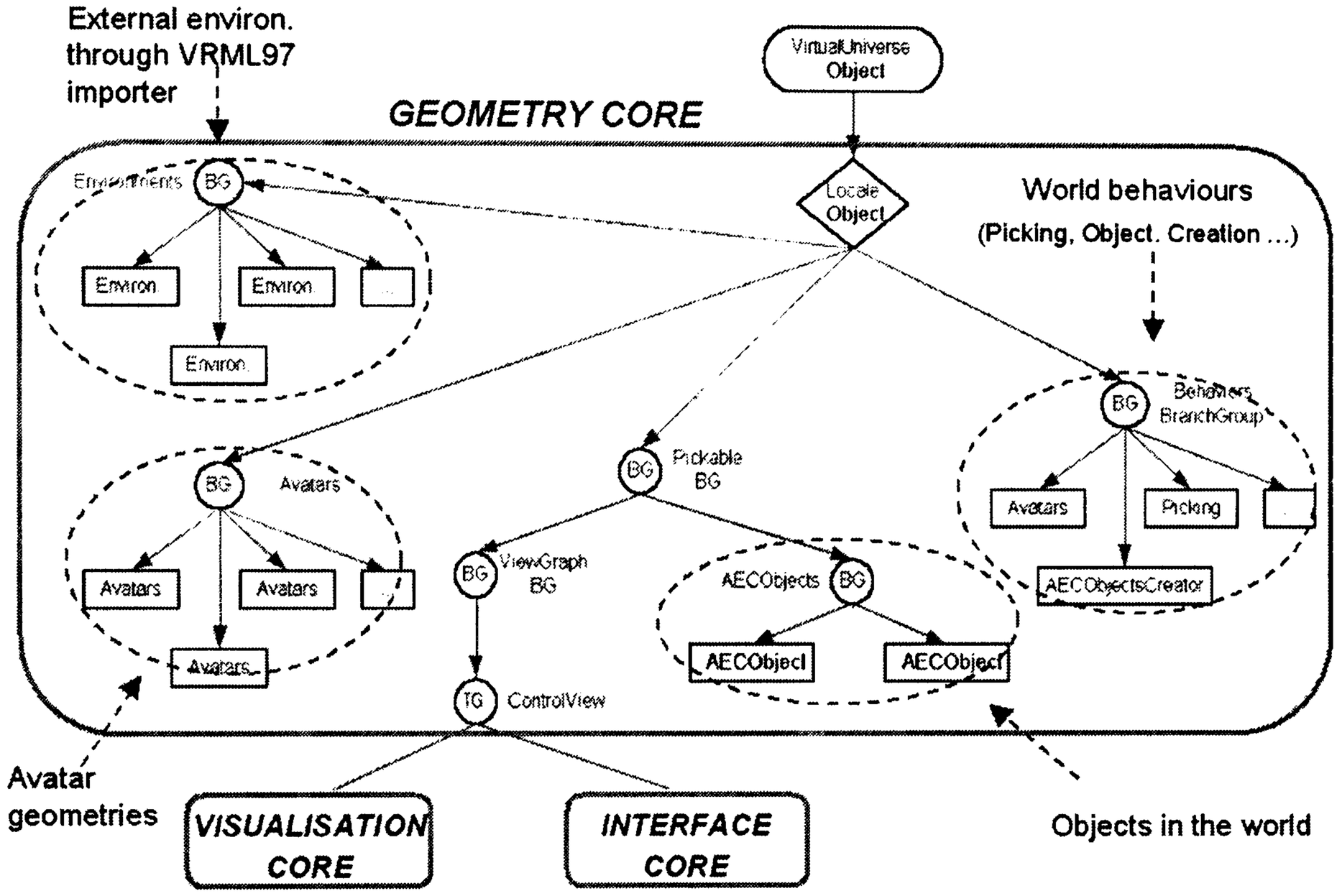


Figure 7.1: The section of the DAG implementing the Geometry Core

From Figure 7.1 it is possible to see that the Geometry Core handles the geometry of every object visible in the virtual world. This includes:

- The environment used as the context of the design, loaded from a 3D-model built using CAD software.
- Geometries of avatars whose position is upgraded every time the sever notifies the movement of a user in the virtual world.
- The *behavioural* components of the system, responsible for decoding the user's commands.

- The objects created by the user.

This chapter will show the details of each sub-branch of the DAG and later sections will outline the advantages of this layout. Moreover special attention will be paid to the illustration of the *behavioural* features of the system. This is achieved through a set of nodes whose task is to interpret the user's mouse movements and clicks into meaningful actions according to the state of the system.

7.2.2 The Object Inheritance Structure

The previous chapter outlined the advantages of the multiple level of abstraction given by the use of *inheritance* in the general management of the programming code. Specifically, within the Geometry Core, every class used to create 3D-shapes *extends* a single super object called SuperAECObject (See Figure 7.2).

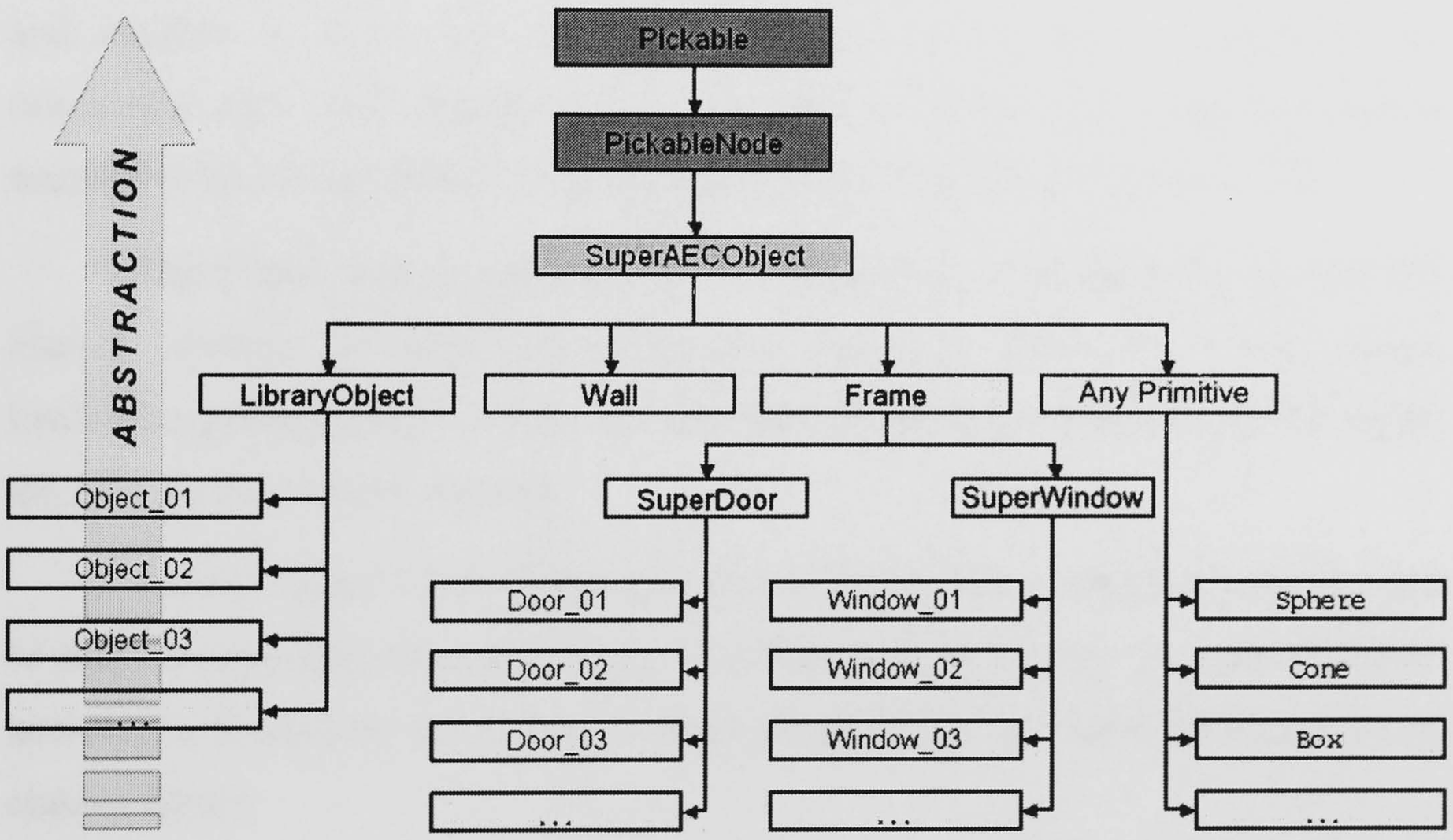


Figure 7.2: The inheritance structure of the objects created by JCAD-VR

The SuperAECObject class outlines the general features shared by every 3D-shape created in the virtual world. This is done through a number of routines, or *methods* that deal with the technical details of each specific geometry.

In particular this class defines the details of how an object manages its geometry during the construction process. Most of the methods used for the manipulation of each shape's geometry are defined at the SuperAECObject class level. However, this class does specify their implementation and consequently every class extending from it has to specify their content.

This approach provides a shared framework and a sequence of operation which is common to every object and which can be used to handle the creation of the shape at a more general level. Each particular object then implements the routines extended from SuperAECObject according to the requirements of their geometry.

As illustrated in Figure 7.2 the SuperAECObject class is extended by every 3D-shape created with JCAD-VR. These include both geometric primitives, e.g. cones, boxes and spheres and architectural entities, e.g. walls, doors, windows and slabs. From Figure 7.2 it is possible to see that SuperAECObject is actually the super class of another important class called Frame, which is the super class of every door and window in JCAD-VR. This class contains all the routines necessary for interaction with Wall objects and in particular it accesses the database used to manage windows and doors (this will be illustrated in Sections 7.5.1 to 7.5.3).

The Frame class is also the parent of SuperDoor and SuperWindow, the two classes extended by every door or window created in JCAD-VR. These classes handle the general details of how the geometry of a door or window is placed within the existing structure of the wall.

Finally Figure 7.2 illustrates the LibraryObject class, which is the parent class of every element present in the library. This class does not specify any special feature however it is used by the system to keep track of all the shapes belonging to the objects library.

7.3 How the User Creates 3D-Shapes

In JCAD-VR the mechanism which controls the creation of objects within the virtual world is without any doubt the most important part of the Geometry Core.

In JCAD-VR the user can create both geometric primitives, e.g. cones, boxes and spheres, and architectural entities e.g. walls and slabs. The difference between the former and the latter is fundamental. Although from the visualisation point of view a wall is identical to a box in fact, its implementation is substantially different. While a box is just regarded as a simple shape without any further qualities, the wall owns specific properties. First of all it can be the parent of other sub-objects like windows and doors. Its geometry is also made of three different surfaces: the internal and external faces and a core. These geometries are independently reconfigured when doors and windows are attached to it (as illustrated in the later Section 7.5.1).

As already mentioned the creation and editing of objects in JCAD-VR is a very dynamic and visual process since users do not just type in values but use the pointer to instruct the system on the object's type and size. In addition, since the object is being built and adjusted interactively while users move the mouse, this immediate feedback encourages users to experiment with different design configurations.

In JCAD-VR (as illustrated by Figure 7.3) the creation process is a three-step procedure. First the geometry is created with default values, then it is dynamically adjusted by the user, and then it is finalised and made accessible to the other participants of the collaborative session.

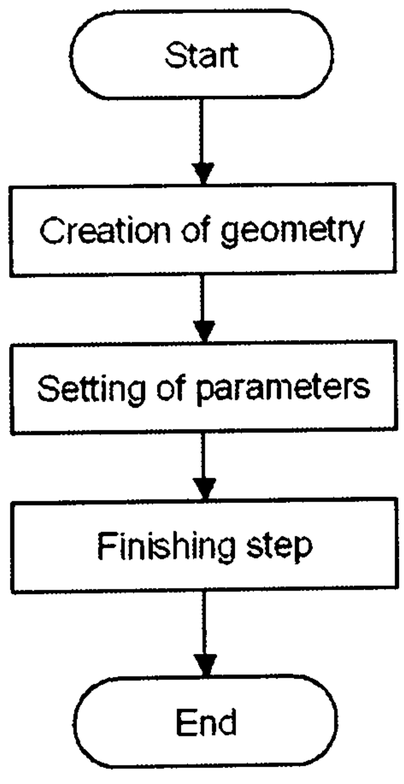


Figure 7.3: The three phases of the process in creating 3D-shapes

While the first and the last stages are instantaneous the second (the setting of the geometry) relies on the users interaction to take place. The user moves the pointer

and the system adjusts the object accordingly until the desired dimension is achieved and the user confirms their choice through a click on the mouse. The user can quit the second stage at any time. If this happens the system assigns standard values to the remaining undefined dimensions.

7.3.1 An Example of the Creation of a Geometric Primitive

In practice users create 3D-shapes by selecting the type of geometry required from the relevant 3D-menu. They then use the pointer to create the geometry through a sequence of drag-and-click operations.

Figure 7.4 shows an example of the interactive creation of an object (a cone). The circles in the image show the location of the pointer on the screen every time the left button of the mouse was pressed.

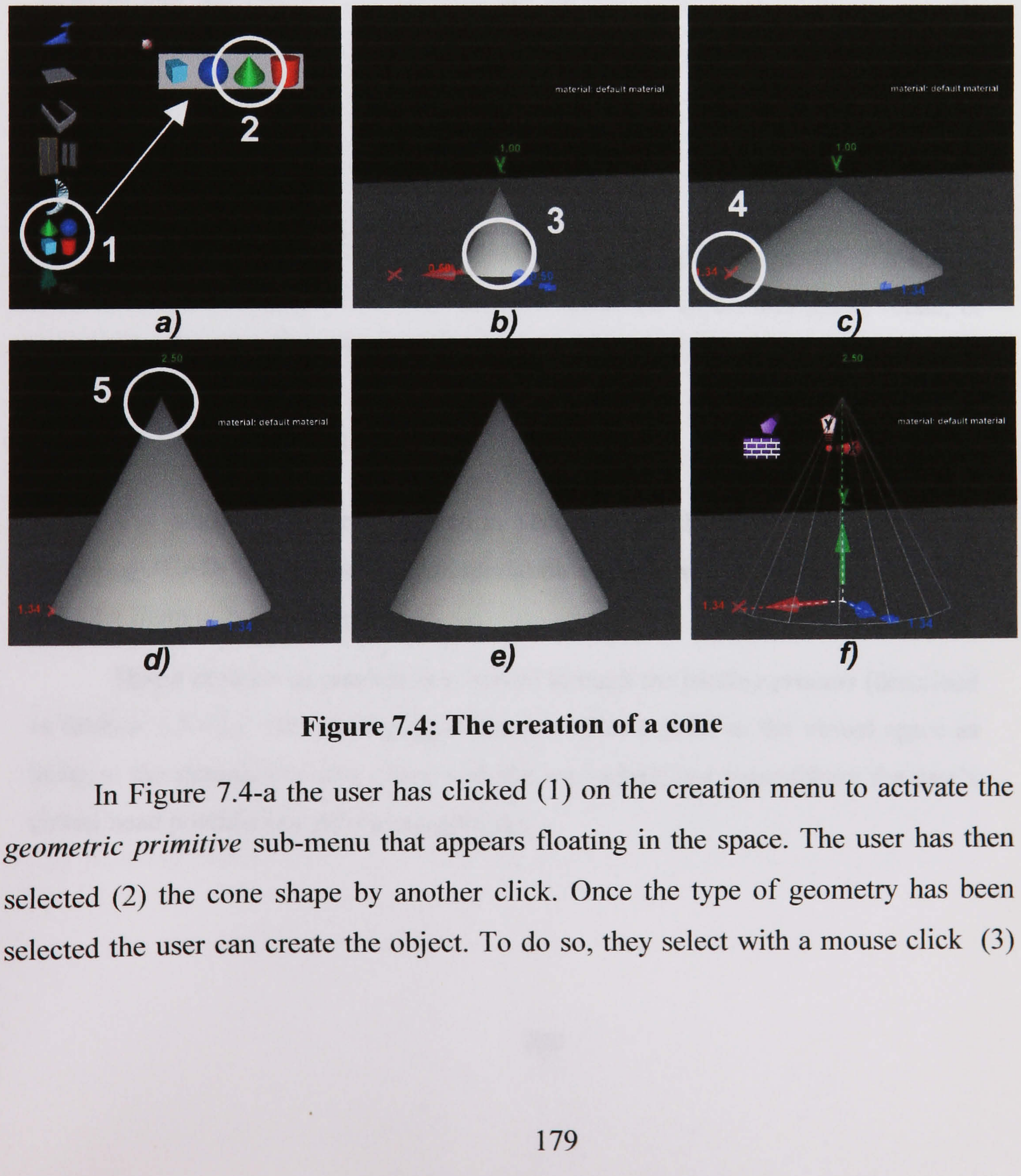


Figure 7.4: The creation of a cone

In Figure 7.4-a the user has clicked (1) on the creation menu to activate the *geometric primitive* sub-menu that appears floating in the space. The user has then selected (2) the cone shape by another click. Once the type of geometry has been selected the user can create the object. To do so, they select with a mouse click (3)

the point on the "ground plane" where the cone is to be placed and the object was automatically placed in the space with its default values (Y=1.00, X=0.50, Z=0.50) (Figure 7.4-b).

Once the object is placed in the space the user can decide the dimensions of the object being created through a sequence of mouse commands. First they moved the cursor to define the first dimension of the cone, the base radius. This was interactively adjusted according to the position of the pointer until a mouse click (4) confirmed its final value (Figure 7.4-c).

Then the user moved the cursor to define the second dimension, the height of the cone. As in the previous case this can be interactively adjusted at the user's wish, until the mouse button was pressed (5) to confirm its value (Figure 7.4-d).

The system, is aware then that the object had been fully defined, and it instantly finalised the object. At this stage (as shown in Figure 7.4-e) the 3D-widgets disappeared, the object's information was stored in the internal database and a message was sent to the server. Once the message was received with the information about the cone the server broadcasted the information to the other users whose systems then upgraded the content of their virtual worlds to reflect the new state of the object.

Once the object is finalised it can be accessed again and edited in a similar way (as shown in Figure 7.4-f) any user can select the object and move, rotate, or scale it by using the 3D-widgets (as described in Section 6.5.2.2).

7.3.2 A Mouse-Driven Process

The simple example illustrated in Section 7.3.1 shows that the user creates shapes in JCAD-VR through the interpretation of their mouse actions, without the need for them to type in values.

Shape creation in practice is achieved through the *picking process* (described in Section 6.5.4.2). The picking algorithm identifies a point in the virtual space as being at the intersection of a plane with the *ray* (which has passed from the user's virtual head position and the mouse pointer).

During creation the *picking process* extracts a set of points from the position of the mouse on the screen. In the example shown in Section 7.3.1, the system placed the cone in the space through the point identified by the user's mouse click (See Figure 7.4-b). This point is calculated as the intersection of the picking ray with the horizontal plane and the cone is placed so that the centre of its base coincides with this point.

Once the geometry has been created at a default size, the user can customise a number of settings, in the example of the cone the base radius and height was changed. To do so The *picking algorithm* calculates the intersection between the ray and the horizontal plane for every frame while the user is moving the pointer. This point is used by the system to set the base radius as the first parameter for every frame. In particular its value is set in relation to the distance between the point just calculated and the centre of the base.

The result is that as the users move their pointer they have the impression that the base of the cone changes size according to their pointer's position. Once the user has decided on the final value he/she confirms it by clicking on the left button of the mouse. In the previous example the system set the final value of the radius of the cone and passed on to the following step where the user set the height.

As in that case the height of the cone is calculated through the intersection of a *ray* with a surface. This time the surface used is a vertical plane passing from the centre of the base of the cone. As a result when the user moves the pointer the height of the cone is interactively changed and appears to follow the pointer's movement.

7.3.3 The Event/Listener Pattern

In the case of the programming code involved every time the user moves the mouse or clicks on one of its button an *event* is generated.

An *event* is a special type of object used to automatically transmit some data between different parts of the code. More precisely, in Java™ a class can "fire" an *event* to a number of *listener* classes that receive it automatically. The *event/listener* pattern is the Java™ specific implementation of a more general *Observer/Observable* programming pattern. This is also known as the *Model/View/Controller* architecture

which was introduced by the Smalltalk programming language. This programming technique provides support for asynchronous one-way *wiring* through *event* notification. The *event/listener* pattern allows precisely the *dynamic binding* of separate elements of the code through the use of special objects, called *events*, which deliver some types of data between parts of the code. The advantage of this pattern is that the data transfer is not achieved through a direct *hard wiring* of routines, but is done automatically by the system at run time whenever a new event is generated.

For instance if two classes (A and B) are registered as *listeners* to a third class (C) every time class C "fires" an *event*, A and B will be automatically notified. The notification mechanism automatically triggers a routine in both classes. As a result class C does not need to directly invoke the routines within A and B. Instead it simply "fires" an *event* and every class registered with it as a "listener" will automatically respond by triggering the relevant method.

In JCAD-VR, every time a mouse movement or the pressing of one of its buttons is detected, the system "fires" a special *event* called MouseEvent. This contains all the relevant information about the action such as the position of the pointer on the screen, the time and the type of action.

This information can then be used by the system to interpret the user's commands into meaningful actions. In JCAD-VR the precise interpretation of the user's commands takes place using a four level interpretation mechanism, which will be described in the following section.

7.4 The Four Level Interpretation Process

The previous sections showed that the process of creation or editing of 3D-shapes depends completely on the interpretation of the mouse commands. The user can similarly access other functions, like navigation or interaction with menus by moving the mouse or clicking its buttons.

JCAD-VR requires the use of a three button mouse or alternatively a mouse with two buttons and a roller configured to work as the central button. The left button is used to start an action, for example selecting a menu, opening a door, selecting one

of the 3D-widgets or confirming a point in the space. The right button is used to select an object and to edit it. Finally, the central button or roller is used to start the navigation.

Every time the user interacts with the mouse a specific type of *event* (See Section 7.3.3) called MouseEvent is generated by the system. Unfortunately the version of Java™ for Windows platform used during the development of this system (See Appendix D) did not properly support the signals coming from the central button of the mouse. This version of Java™ specifically did not generate the correct MouseEvents when the central button of the mouse was pressed. However *events* were correctly generated every time a drag or move action was requested.

As a consequence, when JCAD-VR is run on the Windows platform the user cannot press the central button of the mouse to start the navigation. Instead, the user is obliged to select the relevant icon from the navigation menu and then begin navigating while keeping the central button pressed. This problem does not occur on the Sgi version where the system fires the correct MouseEvent every time a click from the central button of the mouse is detected.

Every time a MouseEvent is received the system has to decode it to transform it into a meaningful command. In JCAD-VR the interpretation of the user's commands is based on the four level process shown in Figure 7.5.

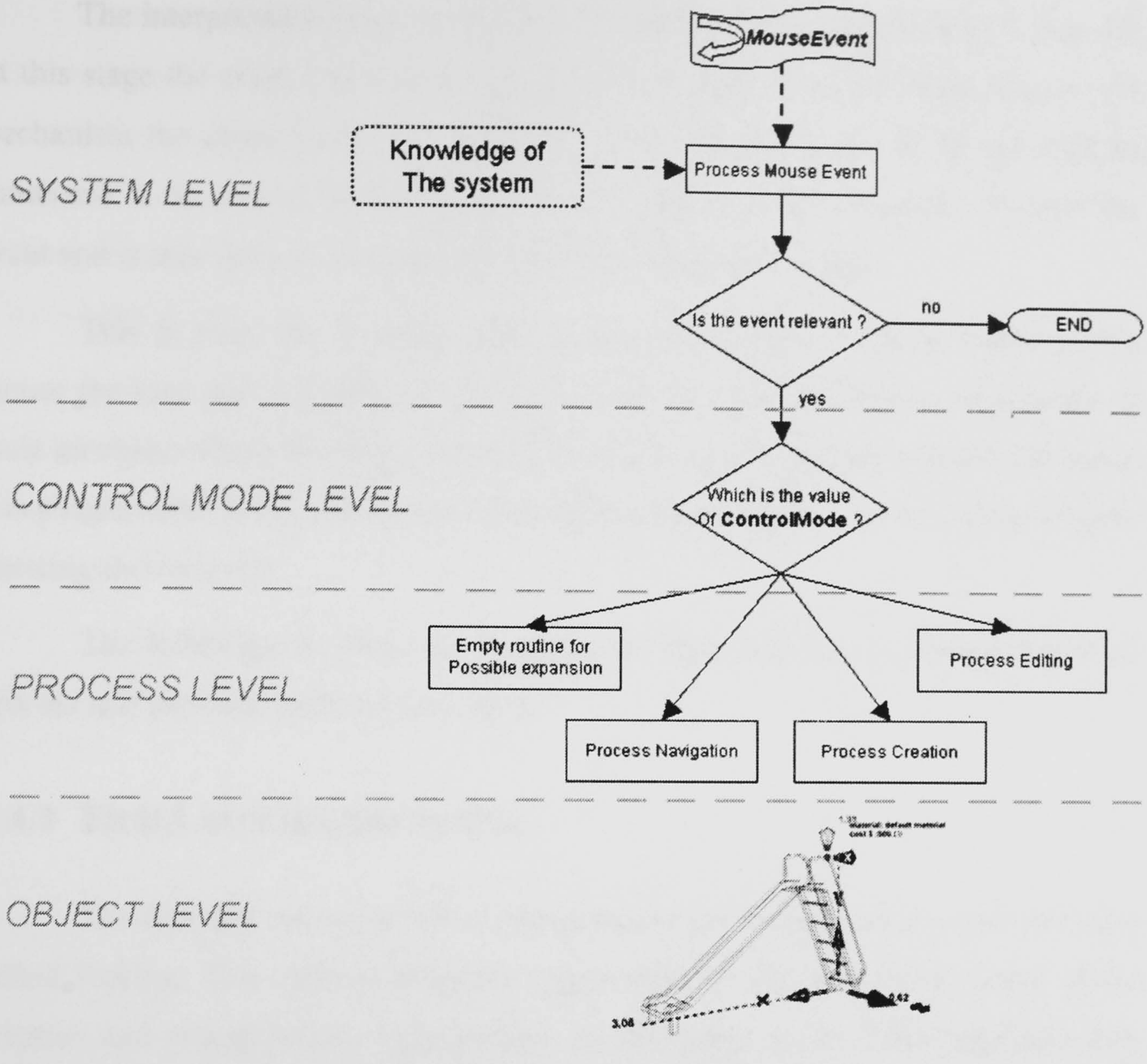


Figure 7.5: An overview of the algorithm responsible for the management of MouseEvents

Every time a class belonging to a level receives a new MouseEvent, it uses the information contained in the event to change its state accordingly and then it passes the *event* to the following level for further interpretation. The effect of the event on the system is distributed in this way. The classes that handle the general logic of the interaction use the event to set the state of the system. The classes used to create shapes use the event to change the state of objects.

More specifically the first level has to evaluate whether an event has to be processed or ignored. The second level then operates as a router that identifies the nature of the event and it passes it to the relevant process on the third level. There the event is handled by different sub-routines depending whether it carries information on the creation or editing of objects or on the navigation of the environment.

The interpretation process continues until the fourth and last level is reached. At this stage the event reaches the object and it is fully decoded. According to this mechanism the object itself is responsible for the interpretation of the information contained in the event. At the fourth level the object itself ultimately receives the event and it acts upon its routines and variables to change its state.

This is how, for instance, through the pressing of the same button of the mouse the user can minimise or maximise a menu, open or close a door, rotate or scale an object. Once the object receives the event it performs the fourth level action it is programmed to do: a fixture opening or closing its panel, a menu disappearing or showing its icons etc.

The following sections will illustrate the logic structure of the interpretation process and provide details of each level.

7.4.1 First Level Interpretation

The first and second levels of interpretation are implemented within the class called Picking. This class is primarily responsible for the interactive nature of the creation and manipulation of geometries. As illustrated in the following pages this class identifies the type of interaction intended by the mouse action through an articulated algorithm that uses the picking mechanism described in Section 6.5.4.2.

The interaction is provided by extending the Java 3D™ class responsible for the *behavioural* features of an environment. This will be referred to as the Behavior class, rather than the Behaviour class to conform with the existing technical literature.

The Behavior class provides the means for the automatic execution of commands in response to a triggering condition. This could be as a result of the state of the virtual world, like the rendering of a certain number of frames or the elapse of a specific number of milliseconds. A triggering condition can also be generated, as in the case of the Picking class, by a user's action in a command inferred through a mouse or keyboard. More complex triggering conditions can be the results of many conditions linked together.

As soon as a triggering condition is met the Java 3D™ *scheduler* invokes the relevant Behavior class. This in turn automatically executes the command contained in a specific routine. The command can cause a change in the structure of the DAG, (for example changing attributes or the position of a node) as well as modification of part of the code that is independent from the content of the virtual world (the dispatching of a message through the network for example).

In this way, a single class can elegantly and efficiently decode different types of triggering conditions rather than using several *event/listener* mechanisms. The interpretation process becomes closely attached to the structure of the 3D-world itself since a Behavior node can be placed within the DAG of the system. In the case of the Picking class, every time the user interacts with the mouse a new MouseEvent is generated and when this is notified it automatically triggers the interpretation process.

As shown in Figure 7.5, at the first level the algorithm receives the MouseEvent and it decides through the information it has on the state of the system whether the event just detected is relevant to the process being performed. At this stage repeated or irrelevant commands are filtered out. If the user has already selected an object for instance, every further event requesting the selection of the same object is ignored. If the MouseEvent is considered relevant instead, the system has to decode its content and therefore it passes the event to the second stage of the algorithm: the *control mode* level.

7.4.2 Second Level Interpretation

The Picking class is also responsible for the second level. At this stage (which takes place in a method called *processMouseEvent*) the class acts like a router sending the relevant information to the correct third level routine. This is done according to the state of the system specified in the *controlMode* variable which is used to monitor if the system is already engaged in a creation, navigation or editing process.

The system checks whether the MouseEvent is coherent with the present *controlMode* and if this condition is met it sends the event on to the third level to the

relevant process that will provide the means for further decoding of the action. If instead the action is found to be incompatible with the present state, the system interprets the event as a command from the user to change mode. The algorithm then changes the value of the *controlMode* variable and passes the MouseEvent on to the relevant process as shown in Figure 7.5.

If for instance the user presses the central button of the mouse during the creation of a shape, which is done with the left button of the mouse, then this is interpreted as a command to switch from creation to navigation mode. The system then interrupts the creation of the object and assigns the default parameters to it. It then changes the *controlMode* variable and routes the event to the third level for the navigation sub-routine and further decoding.

7.4.3 Third Level Interpretation

At this level (as shown in Figure 7.5) three different routines are implemented for the navigation, creation or editing of objects. Navigation has already been illustrated in the previous chapter although from a different perspective. Therefore the next sections will only give the details of the two algorithms that are relevant to the Geometry Core: the processes dealing with the creation and editing of objects.

7.4.3.1 The Interpretation of Creation Commands

As already shown in from Figure 7.4 in JCAD-VR the user can create shapes through clicks of the left button and movements of the mouse. However, the left button has several other functions for instance it can be used to create geometries, activate a command through a menu or to open or close a door.

As a consequence, every time a MouseEvent is generated after the left button of the mouse is pressed the system needs to retrieve more information in order to be able to decode the event correctly (See Figure 7.6).

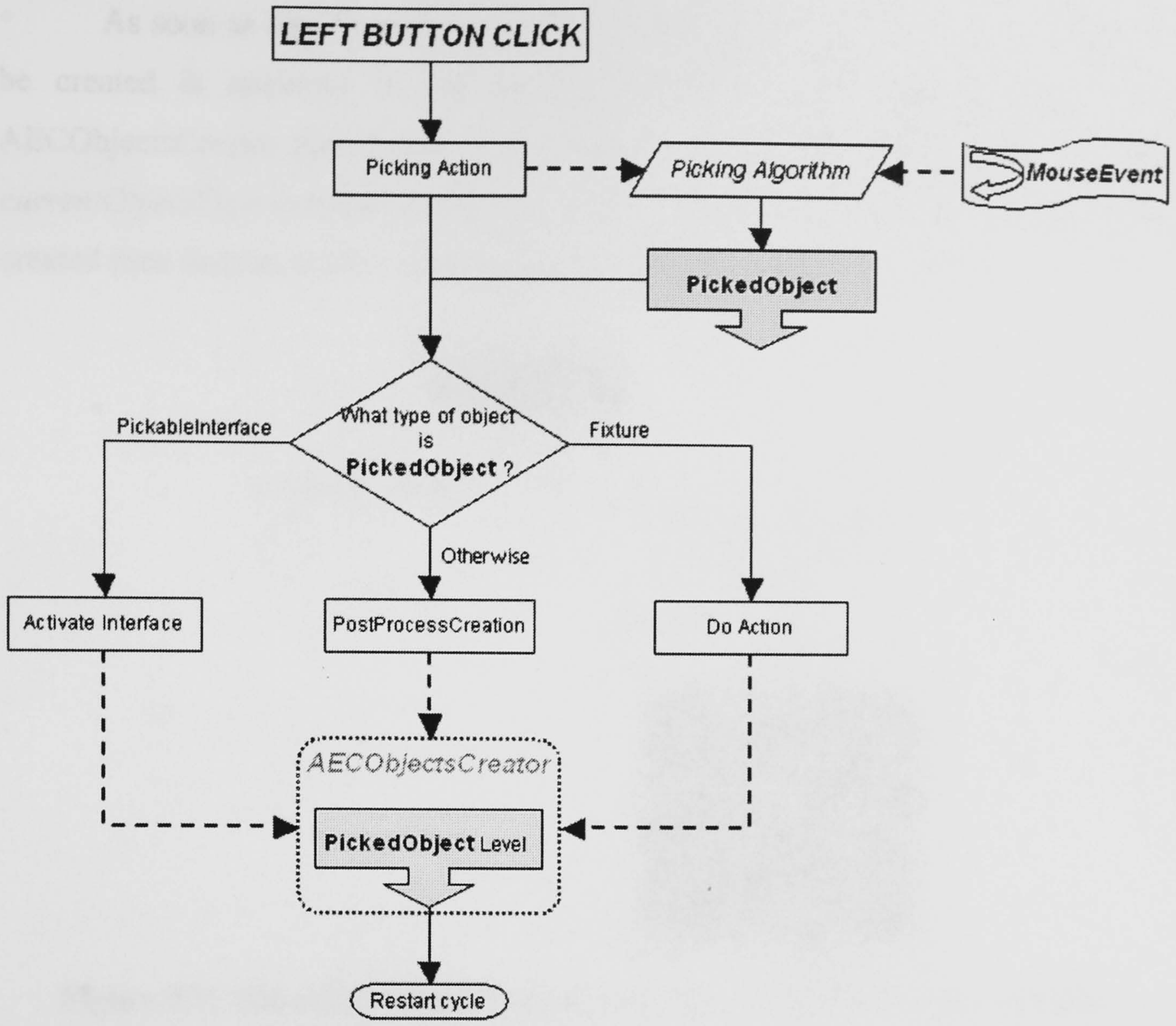


Figure 7.6: The algorithm responsible for the creation process

This system invokes a *picking process* (See Section 6.5.4.2). This allows it to retrieve information regarding the type of object being selected. If the user has selected a menu the system has to execute the relevant command. If they have selected a door or window it has to pass the event to the object so that the panel can be opened or closed. If neither of these two conditions is met the action is considered to be part of a new or an existing creation process. In this case the MouseEvent, which had generated the picking process, is passed to the class that manages the creation of objects: the *AECObjectsCreator* class.

The *AECObjectsCreator* is the backbone of the creation process. It is triggered once the system has recognised that the MouseEvent deals with the creation of an object. Its *currentObjectType* variable defines the type of object being created. The value of this variable is set every time the user selects a 3D-menu to create an object.

As soon as the 3D-menu is selected, the information on the type of object to be created is retrieved by the picking mechanism and it is passed to the AECObjectsCreator (See Figure 7.7). Then the AECObjectsCreator class sets the *currentObjectType* variable to the correct value identifying the type of object being created (See Section 6.2.2.1 of the companion thesis Ucelli, 2002).

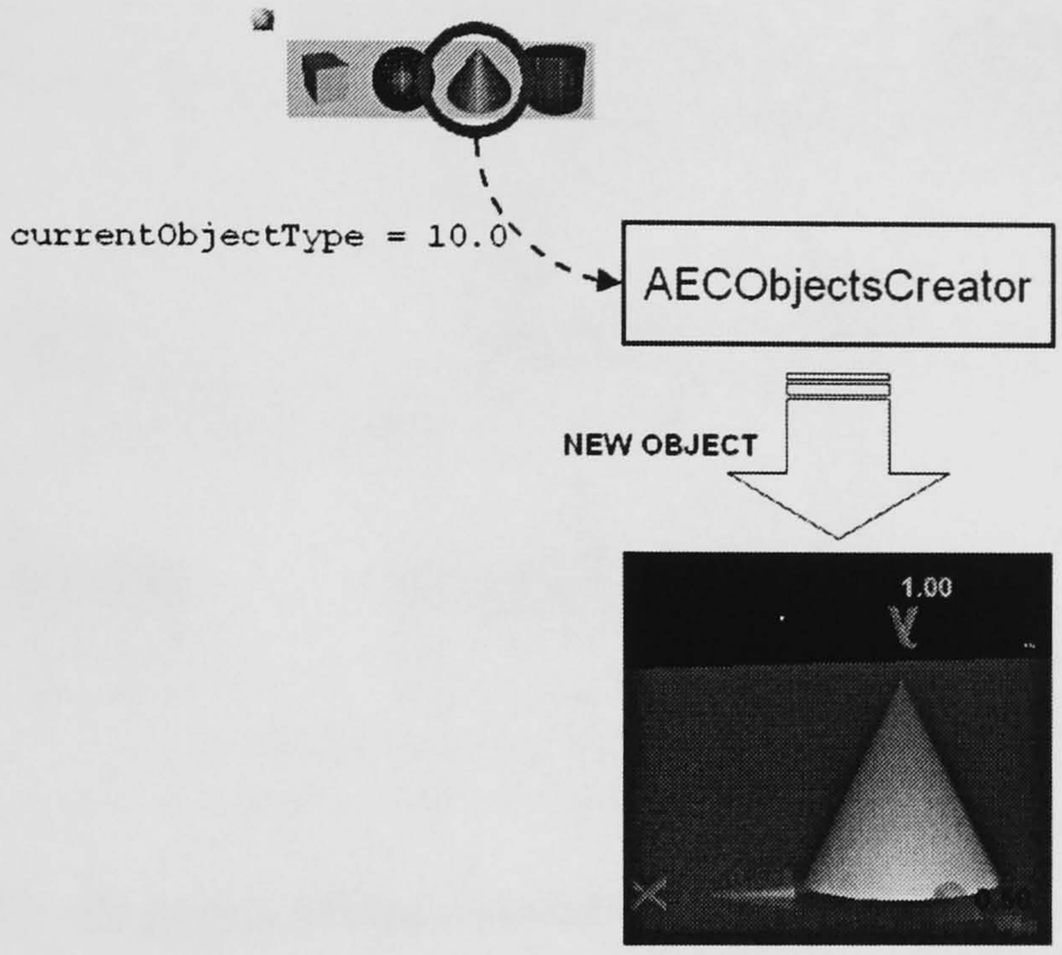


Figure 7.7: The AECObjectsCreator and the *currentObjectType* variable

When AECObjectsCreator is invoked it retrieves the state of the object being created to decide what action is to be taken. If an object is currently being created the information received is used to set its state accordingly. If the AECObjectsCreator finds instead that there is no object being created then it will start to create a new one.

However, although AECObjectsCreator manages the overall creation process the new object is not generated within this class. Instead the AECObjectsCreator class makes use of a different class called AECObjectFactory. The UML diagram in Figure 7.8 illustrates this mechanism and highlights the relationships between the classes involved in the creation process.

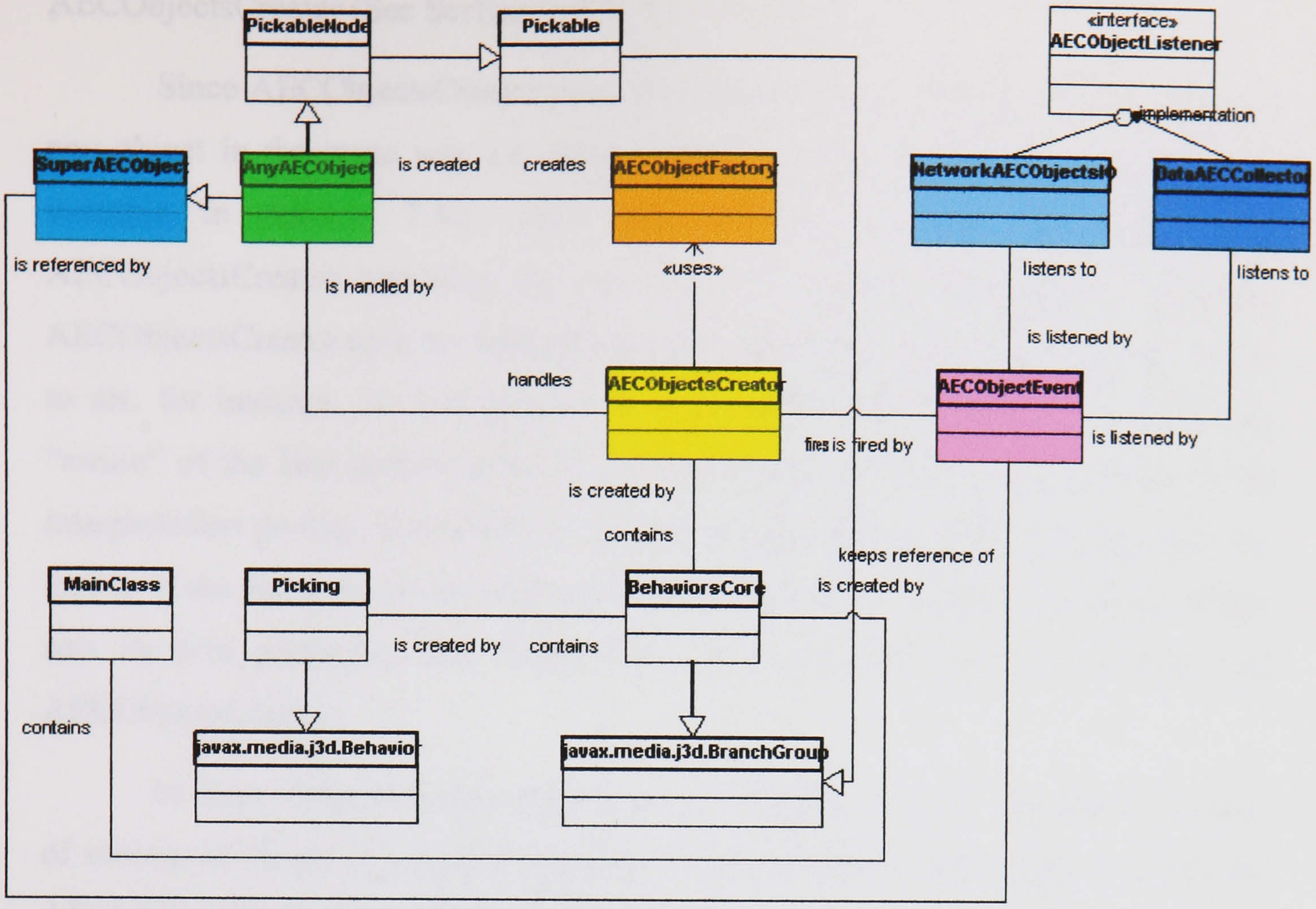


Figure 7.8: UML diagram of the classes involved in the creation of objects

When the AECObjectsCreator (in yellow in Figure 7.8) needs to create a new object it invokes the AECObjectFactory. To do so it passes the specific object type specified by the *currentObjectType* variable to the AECObjectFactory. The AECObjectFactory (in orange in Figure 7.8) creates the object (in green in Figure 7.8) and returns it to the AECObjectsCreator to continue the creation process.

The AECObjectFactory creates the specific object requested by the AECObjectsCreator (e.g. a Cone) however it returns this in the form of a generic SuperAECObject (in cyan in Figure 7.8). This is possible thanks to the inheritance structure (illustrated in Section 7.2.2) according to which every object created in JCAD-VR is ultimately a SuperAECObject.

This programming technique returns an object in the form of one of its super classes. This is then called *upcasting* since the object is "cast" (from the logical point of view) into one of the elements further up the inheritance structure.

The use of the *upcasting* technique permits the AECObjectsCreator to manage the creation process generally, regardless of the specific type of object handled. In fact, independently from being a Door, Window or Cone every class is

ultimately a SuperAECObject and as such they can be handled by the AECObjectsCreator (See Section 7.2.2).

Since AECObjectsCreator receives a generic SuperAECObject it can handle any object in the same way i.e. independently of its specific nature. In the cone example in Section 7.3.1, when the user changes the base radius, the AECObjectsCreator operating the action is not aware of the object type. The AECObjectsCreator uses the information received to generically command the object to set, for instance, its first parameter to a specific value. The system becomes "aware" of the first parameter being the base radius only at the fourth level of the interpretation process (described in the following section). When the MouseEvent arrives at the fourth level, the object (the Cone class in the previous example) finally sets its first parameter (the base radius) to the specific value passed by the AECObjectsCreator.

In short, this abstraction allows the AECObjectsCreator to handle the process of setting the length of a wall or the base radius of a cone in the same way. For the AECObjectsCreator both variables are in fact just the first parameter specified by their super class.

A further advantage of using the *upcasting* technique is the system's "awareness" of the specific nature of each object. Every object, if required, can be turned into its specific type through a *downcasting* process. This term means to cast an object down the hierarchy structure.

As soon as the AECObjectsCreator receives the object in the form of a SuperAECObject, it places it inside the virtual world. As shown in Section 7.3 the creation itself is a three stage process. The object is first placed in the virtual world with its default dimensions, the user sets its parameters and then the object is finalised.

After the 3D-shape has been placed inside the environment, every further MouseEvent received is used to set the parameters of the object until it is finalised. To do so the AECObjectsCreator retrieves from the object a variable called *numberOfParameters* that specifies the number of parameters for each object that the user can set. Then the AECObjectsCreator class continues to route the MouseEvent

to the object until the final number of parameters of the specific object is reached. The specific action of setting a parameter is done at the fourth level since the details of it are specified within the code of the objects. This will be illustrated in Section 7.4.4.

When AECObjectsCreator has set all the parameters specified by the *numberOfParameters* variable it finalises the object. The user can quit the process of creation at any time. In fact, if an action incompatible with the current process is decoded at the second level (See Section 7.4.2) the Picking class orders the AECObjectsCreator to finish the creation. In this case, independently of the number of undefined parameters, the AECObjectsCreator goes directly to the finishing stage. The object's remaining parameters are assigned default values and it is finalised through the call of its *finish* method as illustrated in Section 7.4.4.

When the AECObjectsCreator finalises an object it fires an AECObjectEvent (in pink in Figure 7.8). This *event* contains the numerical representation of the object (See Section 6.2.2.1 of the companion thesis Ucelli, 2002). Through the network this information is also used to inform the other users in the session of the creation of the new object (as illustrated in Section 7.4.3.3). Furthermore the AECObjectEvent contains a unique ID number, called *key*, which is univocally created from the memory location of the object to identify that object.

The AECObjectEvent fired by the AECObjectsCreator is then received automatically through the listener/event mechanism (described in Section 7.3.3) by the NetworkAECObjectsIO and DataAECCollector class (in light and dark blue in Figure 7.8). These classes handle the transmission of data from/to the server (See 7.4.3.3) and the storing of information inside the internal database respectively (See Section 6.2.5 of the companion thesis Ucelli, 2002).

7.4.3.2 The Interpretation of Editing Commands

The process of editing objects (as shown in the previous chapter) is based on their interaction with a set of 3D-widgets (See Section 6.5.2.2). With a right click on the mouse the user selects the object and starts manipulating it by dragging one of the

arrows representing the axis of its local coordinate system. The user can decide to translate, rotate or scale the object through the 3D-icon shown beside the object.

At the third level, from the logical point of view, the interpretation of the MouseEvent when editing objects is a simpler process than the creation equivalent. Once the user has selected the object, the system does not in fact need to retrieve any further information about it and the system starts editing the object regardless of its type.

As shown in the previous chapters (See Section 6.5.4.1), JCAD-VR's inheritance structure makes it possible to deal with elements of the 3D-GUI in the same way as the other objects populating the virtual world. Since every object in the environment ultimately extends the PickableNode super class, they can be edited in the same way regardless of their type. This class (as illustrated in Section 6.5.4.1) provides the means for an object to be recognised by the picking mechanism and to be consequently manipulated. Therefore the algorithm does not need to enquire if the object is for instance a 3D-menu or a Door.

At this level the MouseEvent generated by the pressing of the right hand button of the mouse (as shown in Figure 7.9), triggers a picking process which is used to acquire a reference of the object that the user wants to edit.

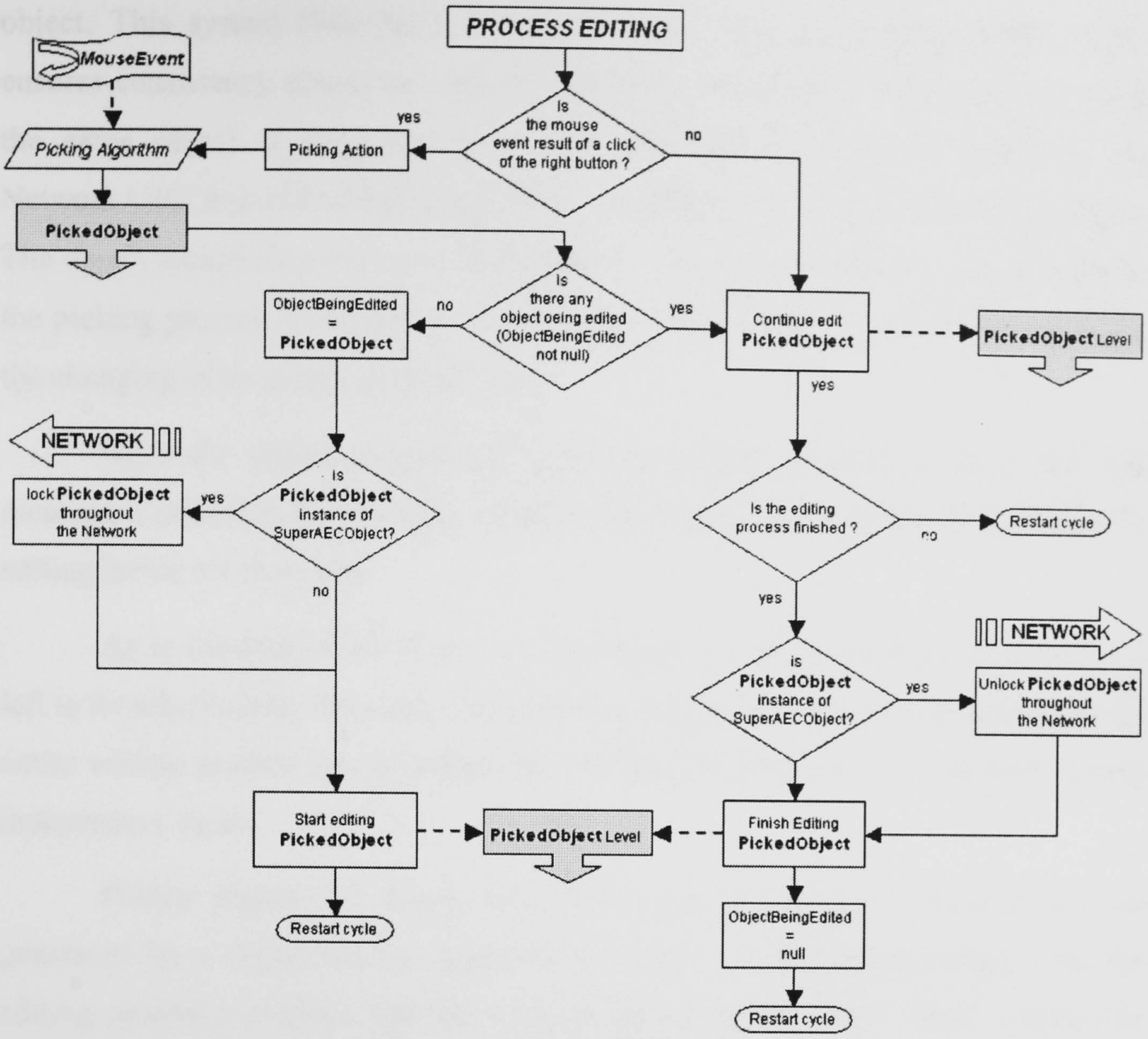


Figure 7.9: The algorithm responsible for the editing process

The picking algorithm is configured in order to constraint the search of the possible objects to the part of the DAG containing the relevant objects. This is done to increase the speed of the algorithm (as explained in Section 6.5.4.2).

Once the picking process has returned the object, the process of editing starts immediately. No special routine needs to be written in the case of the object being a menu, since this is treated in the same way as any other object by the system.

As shown in Figure 7.9 the system then checks if the object selected is the same as the previously edited one. This is done to understand whether the editing command refers to an object already selected or if the user wants to select a different object.

If a new object is selected and if this is also an instance of a SuperAECObject, then the system fires an AECObjectEvent to *network-lock* the

object. This system (See Section 6.2.2.2 of the companion thesis Ucelli, 2002) ensures consistency across the network and avoids that more than one user editing the same object at the same time. The AECObjectEvent is received by the NetworkAECObjectsIO class, which sends a message to the server to lock the object. The object locked then becomes "unpickable", and as a consequence it is ignored by the picking process. Furthermore the object's locked status is made evident through the changing of its colour to bright red.

Once the object is *network-locked* the editing mechanism can start. The message is passed to the fourth level and to the object itself which then performs the editing inside its own class.

As in the case of creation, once the object has been selected manipulation is left to its sub-routines. This mechanism makes the general structure simpler since the entire editing process can be abstracted and the functions can be relegated to each independent object.

Finally Figure 7.9 shows that if the system receives a MouseEvent not generated by a right-click or regarding an object already being edited, then the editing process continues and the event is passed to the object itself. Finally the system checks if the process has finished.

If the editing of the object is completed and if the object is an instance of a SuperAECObject, then the system fires an AECObjectEvent to the NetworkAECObjectsIO class to release the *network lock*. The server then broadcasts a message to release the *network lock* from that object and it again becomes "selectable" by every user in the session.

Although not illustrated in Figure 7.9 an analogous, yet simpler, process takes place when the user selects the object and hits the "delete" key to remove it from the virtual world. If the object is part of the 3D-GUI the command is ignored. If the object is instead an instance of a SuperAECObject then it is removed from the virtual world. An event is then fired and the NetworkAECObjectsIO sends a message to the server. This broadcasts the information to all the other users' systems to upgrade the status of the virtual world.

7.4.3.3 NetworkAECObjectsIO: how Other Users Create Shapes

The previous sections have illustrated the mechanism behind the creation and editing of objects with JCAD-VR. It has also highlighted how this action can affect the other participants in the session through the client/server network. If a user creates, edits or deletes any of the shapes in the virtual world the AECObjectsCreator fires an AECObjectEvent. This is automatically received by the NetworkAECObjectsIO class, which converts it into a message and sends it to the server (See Figure 7.10 a-b). This eventually distributes the message to the other users.

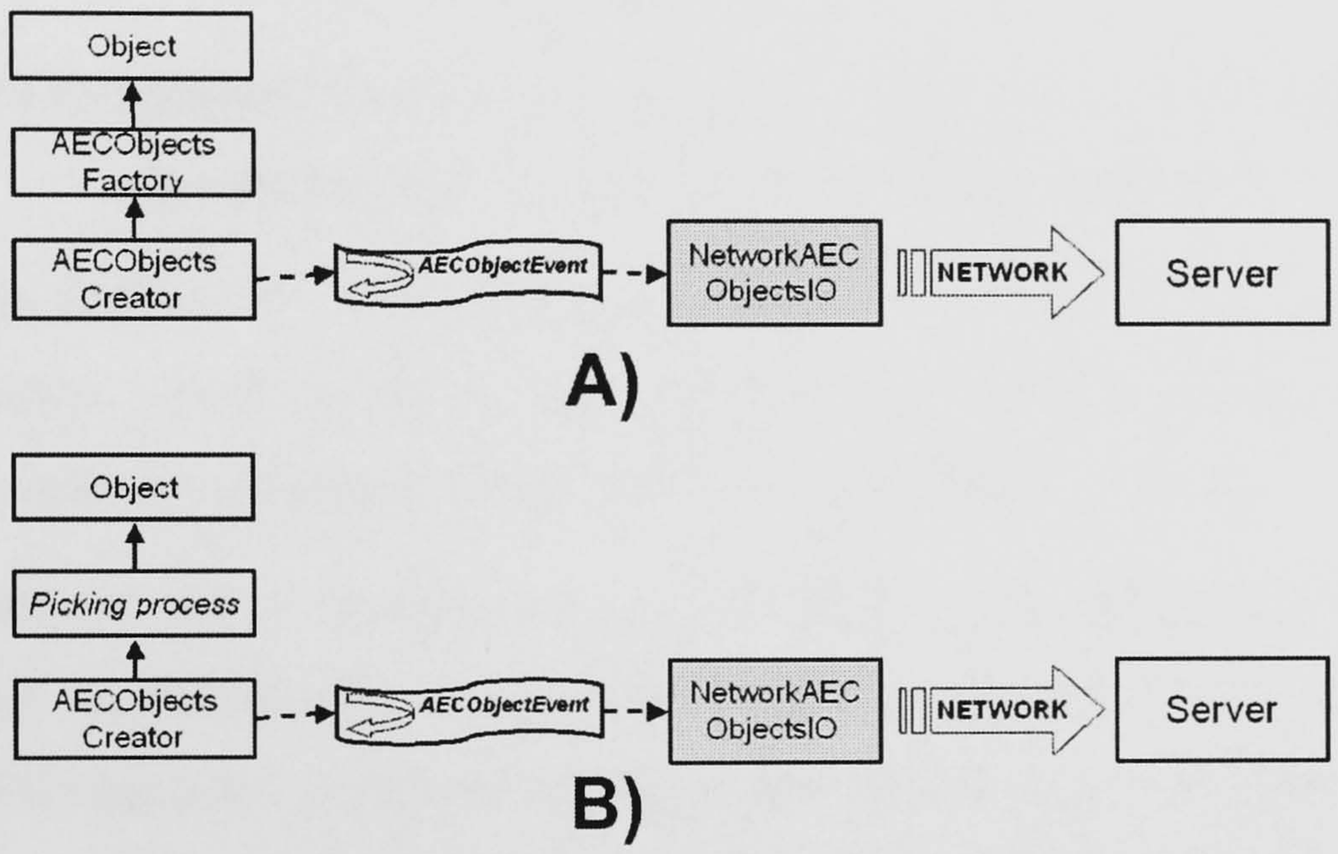


Figure 7.10: Communication from NetworkAECObjectsIO to the server during the creation (A) and editing stage (B)

Symmetrically, whenever a message is notified by the server the NetworkAECObjectsIO class decodes the content and acts on the environment. In the case of a new object being notified (See Figure 7.11-a) the class invokes the AECObjectFactory to create the new object, similarly to the routine described in Section 7.4.3.1 and illustrated in Figure 7.10-a.

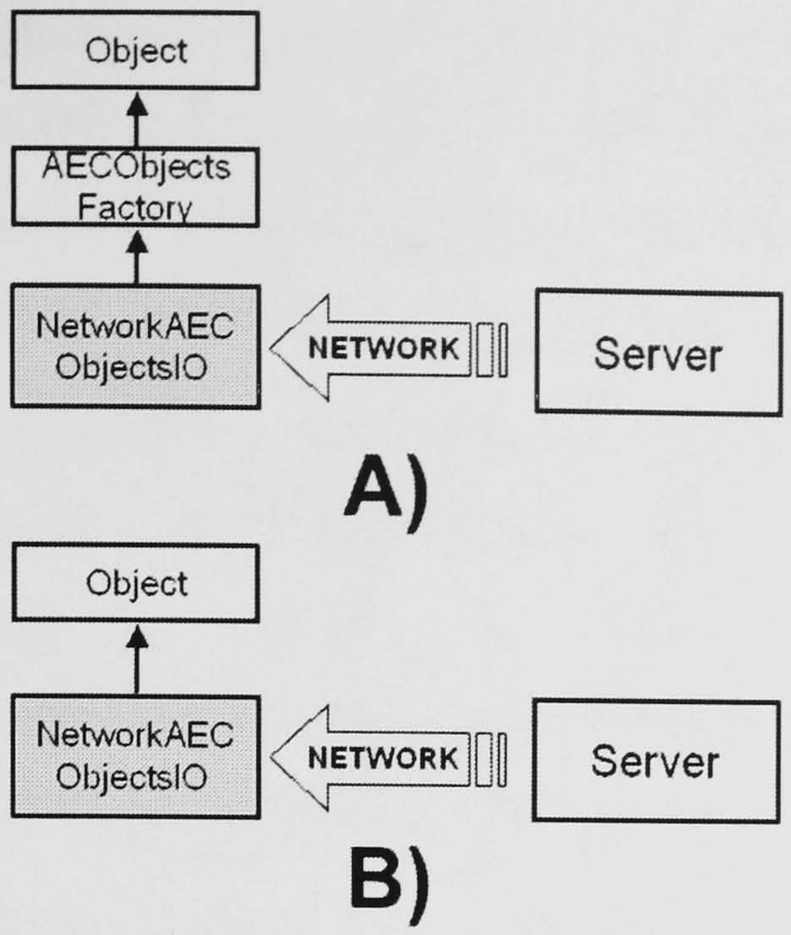


Figure 7.11: Communication from the server to the NetworkAECObjectsIO class during the creation (A) and editing stage (B)

As in Section 7.4.3.1 the object is created with default values. Then by using the information contained in the message, NetworkAECObjectsIO automatically invokes the relevant methods of the object to set up its specific values.

Therefore, the three stages of the creation process (illustrated in Figure 7.3 and detailed in Section 7.4.3.1) are automatically executed by the system. The NetworkAECObjectsIO replicates the algorithm executed by AECObjectsCreator. Therefore, from the implementation point of view, the process followed by the system in this case is identical to when the user creates an object (illustrated in Section 7.4.3.1). The sole difference is that in this case the user's interaction is bypassed. The information on the final state of the object, as well as the *key* used to uniquely identify the object, are already known by the system.

In the case of the message received from the server containing information regarding an object being edited, deleted or locked then the process becomes much simpler than the one described in Section 7.4.3.2. The NetworkAECObjectsIO class directly accesses the relevant object (See Figure 7.11-b) without any need for the picking process. This is done through a routine which checks all the objects present in the world and retrieves the one whose *key* is equal to the one contained in the message.

This process is made more efficient by the structure of JCAD-VR's DAG. Every object created by the system is placed in a specific part of the DAG (See Figure 7.1). This increases the efficiency of the algorithm which therefore only needs to analyse the limited part of the DAG where the objects are located.

In short, it is evident that the process described in this section is much simpler than the ones detailed in the two previous sections. This is due to the fact that the system does not need to interpret any actions from the user. The NetworkAECObjectsIO class can therefore pass the information strait to the object level where the data is used to change the state of the object.

This is done at the fourth level of the interpretation mechanism which will be illustrated in the following section.

7.4.4 Fourth Level Interpretation

As mentioned in the previous sections the interpretation of the user's mouse commands into meaningful actions is a four stage process. Each level uses the information contained in the MouseEvent to change the state of the system and then it passes the event to the following level for further decoding. The first three levels manage the process at a high-level: the system commands the setting of a property without being "aware" of the operations being done at the fourth level. At the fourth or *object level* each object interprets the data within the MouseEvent in order to act directly on its structure.

The next section describes the general mechanism used for the geometry of most shapes. The following ones will highlight the more specific approaches which are followed for the implementation of a specific object: the Wall class.

As noted the creation process is a three stage routine: the creation of the geometry, the setting of the parameters and the finishing process. This is illustrated in Figure 7.12 which also shows the most important *methods* involved in this process.

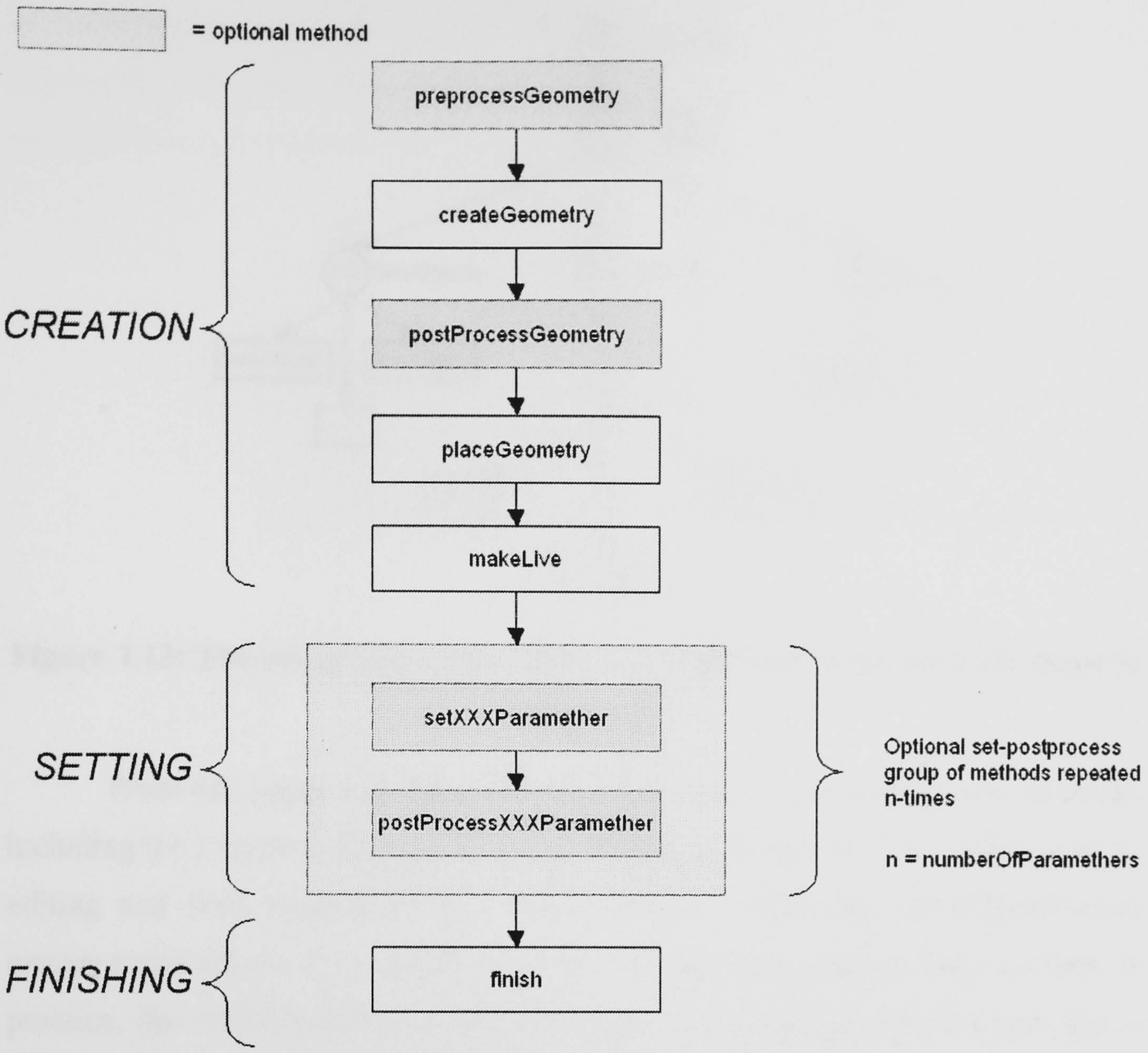


Figure 7.12: The methods involved in the creation of 3D-shapes

Section 7.4.3.1 illustrated the general process of creation of 3D-shapes in JCAD-VR. Every time the AECObjectsCreator receives an event regarding the creation of an object, it retrieves a new copy of a SuperAECObject from the AECObjectFactory class. The internal structure of a SuperAECObject is illustrated in Figure 7.13.

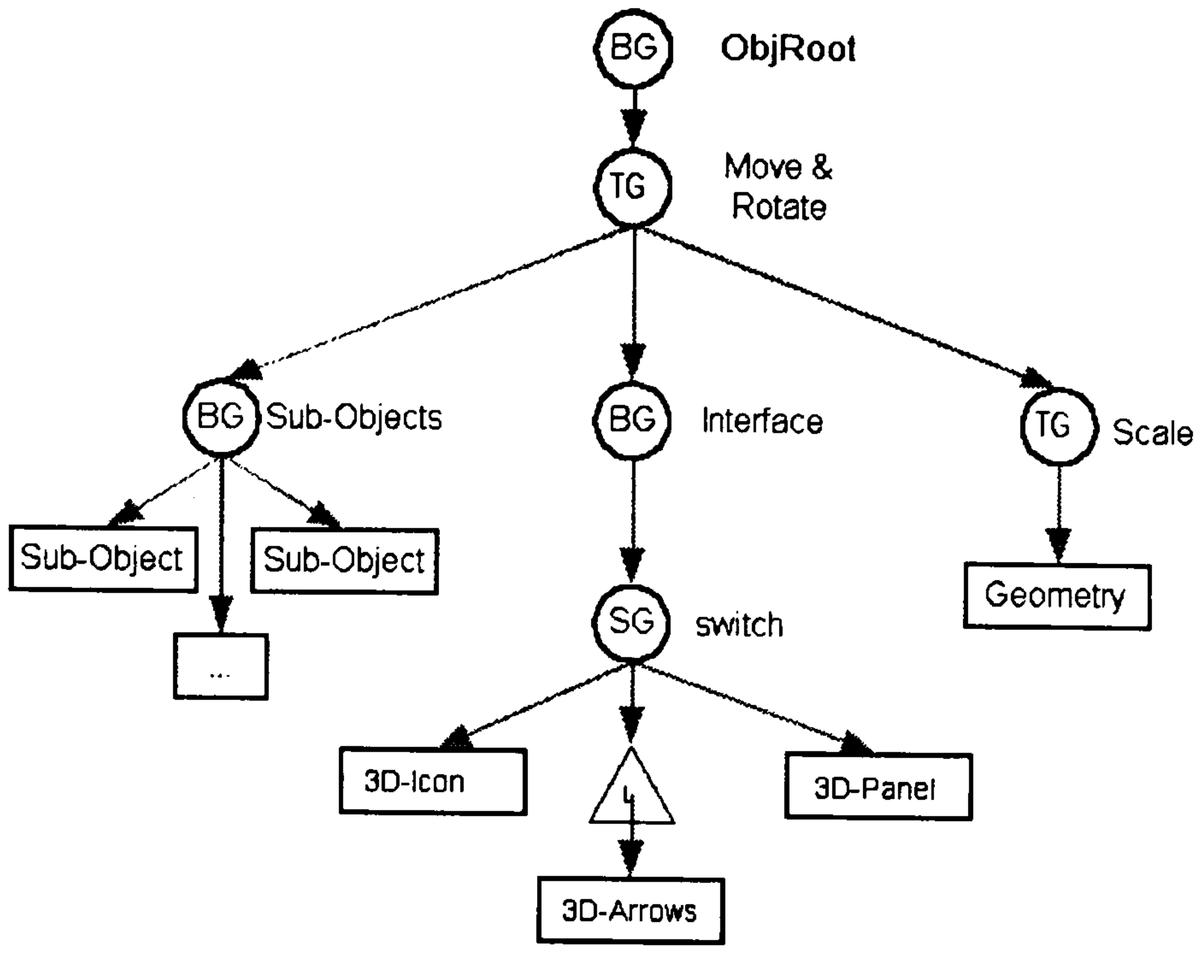


Figure 7.13: The sub-graph representing the nodes used to create a 3D-shape in JCAD-VR

From this image it is possible to see that the object is made of several nodes, including the TransformGroups (TG) used for manipulation, the 3D-widgets used for editing and most importantly the node geometry. When the AECObjectFactory creates a new object, this is returned without the geometry which is specified later. In practice, the AECObjectFactory prepares and creates the general structure that is common to every object, while the geometry instead is created and configured through interaction with the user.

Once the new object has been created the AECObjectsCreator passes the event to it for final decoding. This is when the fourth and final level of interpretation of the event takes place.

The interpretation is implemented within the methods inherited from the SuperAECObject class, the super class of every object created with JCAD-VR. As already outlined in Section 7.2.2, most of the methods of the SuperAECObject class are not detailed. They are defined to provide a general framework, a common set of routines that every shape in the world uses to handle its geometry. In fact, by having a common structure every object can be handled in the same way regardless of the specific nature of their shape. Each object then specifies the content of the methods according to the requirements of its geometry.

When creation starts the algorithm performs a number of operations. This is illustrated in Figure 7.14, which gives a more detailed view of the first group of *methods* shown in Figure 7.12.

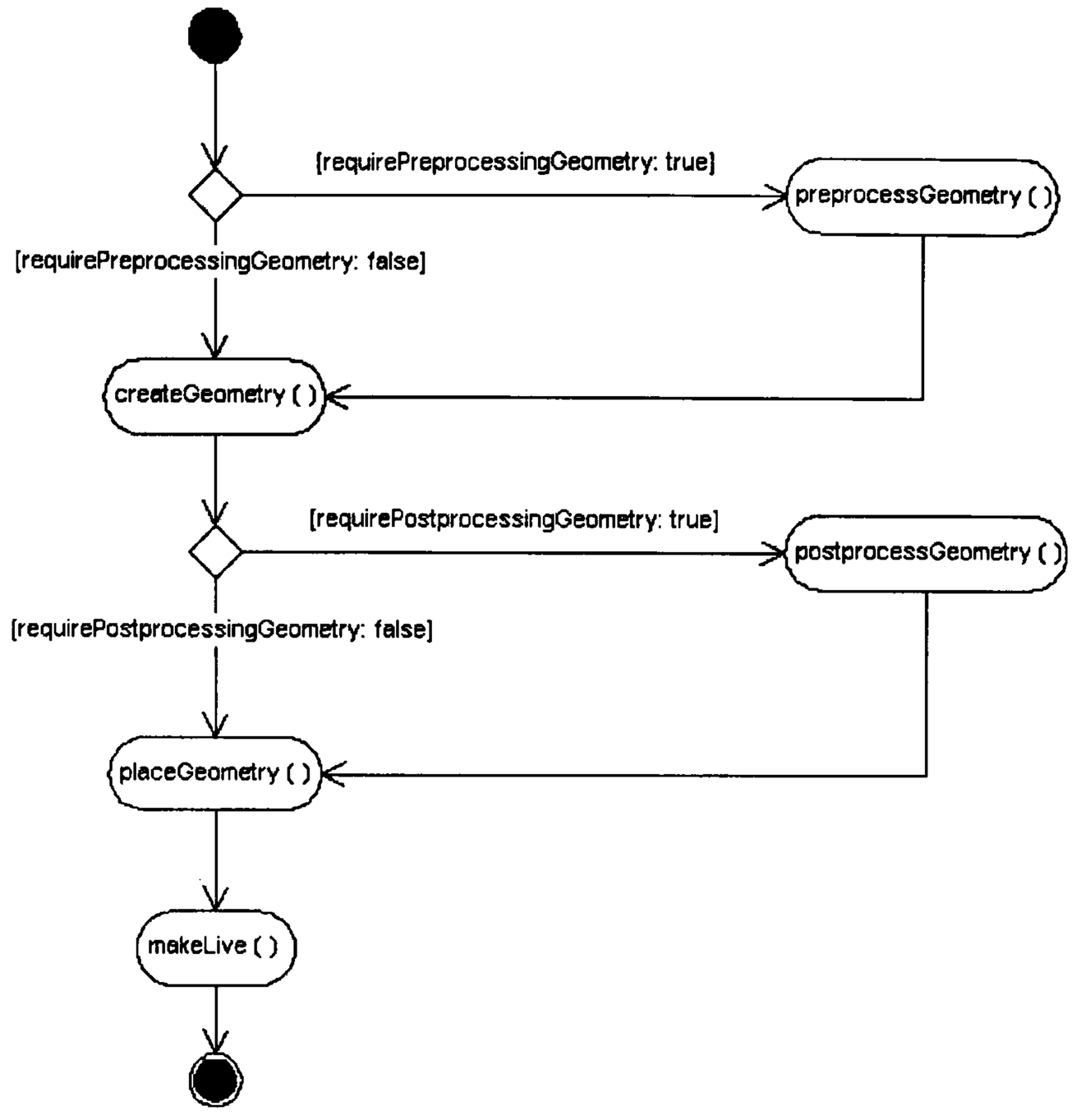


Figure 7.14: UML diagram of the SuperAECObject's methods involved in the creation of the geometry

If the object requires a particular pre-processing step before the geometry is created a special routine is called (See Figure 7.14). After this the geometry is created in the *createGeometry* method, which is implemented according to the object type. JCAD-VR's library objects will retrieve their geometrical information from pre-modelled VRML97 files through an external importer (See Section 7.6). Geometric primitives alternatively will use Java 3D™ classes to create their geometrical representations. In the case of the Wall class the details of the geometry will be specified within the code (this will be illustrated in the following section).

After the geometry is created, if the object requires specific post-processing operations, the relevant method is called (See Figure 7.14). The geometry is then placed in the space through the *placeGeometry* routine (See Figure 7.14). As

described in Section 7.3.2 (in the example of the creation of a cone), when the user creates a shape the system extracts a number of points resulting from the location of the pointer. The *placeGeometry* method then takes the first point defined by the user to place the object in the space (See Figure 7.4-b) and to set the fields of the *moveAndRotate* TransformGroup (TG). This *moveAndRotate* TG (illustrated in Figure 7.13) moves the object to the position in the virtual world chosen by the user through a click of the mouse.

Finally the system *calls* the *makeLive* method (See Figure 7.14) which is used to attach the diagram representing the object (illustrated in Figure 7.13) to the main DAG. In this way the object is attached to the virtual world becoming visible to the user. The first stage shown in Figure 7.12 is completed by this method.

The object with its default values is now placed in the virtual world (See Figure 7.4-b) and the user can set a number of parameters. As described previously, the number of parameters editable by the user is specified by the *numberOfParameters* variable. This variable is used by the system to keep track of the number of operations to be done on the geometry of an object to fully define it and therefore it is specified in the implementation of each object.

For instance every object in the library has a *numberOfParameters* equal to zero. This is due to the different nature of all the objects present in the library which makes it impossible to generalise on the number of parameters to be used. The object is placed "as it is" in the space, however the user can modify it later. Likewise doors and windows also have a *numberOfParameters* equal to zero. This causes the object to be placed into a wall with their default size.

Other objects have a *numberOfParameters* equal to one. This is the case if the object is a sphere: during its creation the radius is the only parameter that can be set. Some objects can have two parameters available, e.g. a cone where the base radius and the height can be set. Other objects can have three parameters, e.g. a wall, where the user can specify the length, width and height.

After the object is placed in the virtual world, if the *numberOfParameters* is greater than zero, the system starts setting the relevant parameters. This is done through a number of *methods* cycles that set a variable and perform the required

post-processing algorithm. This iterative mechanism is illustrated in Figure 7.15, which is a more detailed view of the second stage of Figure 7.12.

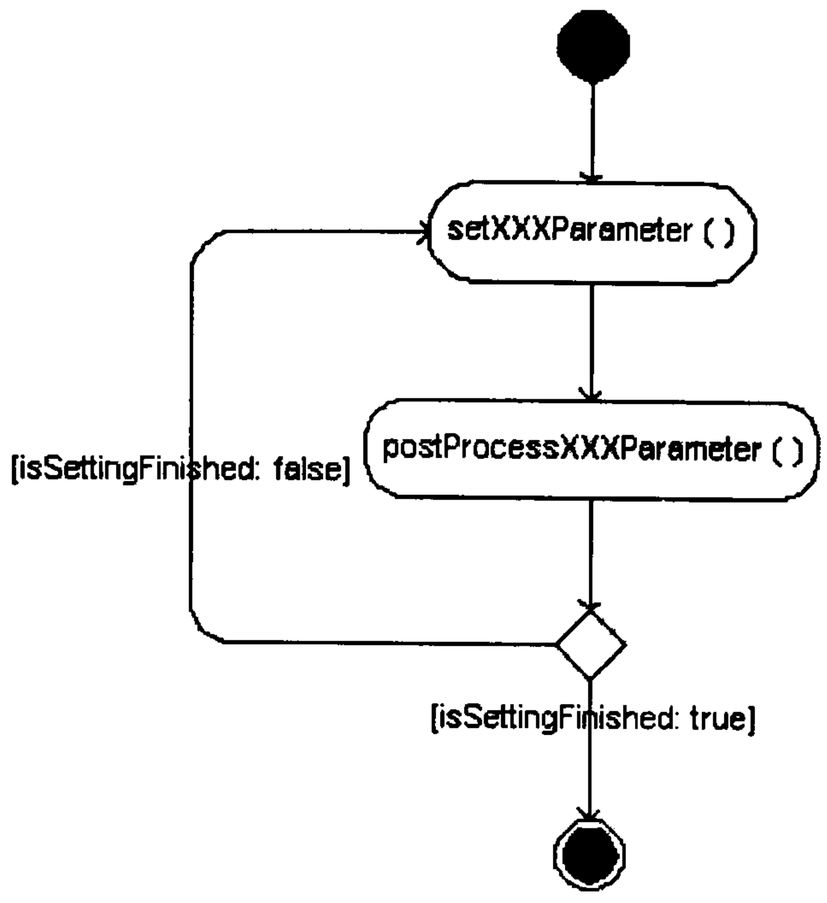


Figure 7.15: UML diagram of the SuperAECObject's methods involved in setting the properties of the geometry

In this image the "XXX" idiom is used for the method names to represent the first, second or third parameter, which is iteratively set. From Figure 7.15 it is evident that after setting the first parameter the system iteratively checks if the setting phase is finished. If it is not, it goes on to the following *setXXXParameter* method. This is repeated the number of times defined by the *numberOfParameters* variables.

Each time the *setXXXParameter* is called the point in the space passed from the picking algorithm is used to calculate the parameter. This value is immediately applied so that the user can see the effect in the following frame. These points (as explained in Section 7.3.2) are calculated by the system according to the position of the pointer. Therefore the user has the impression that the geometry's modifications are directly related to the position of the pointer directed by their mouse. This behaviour provides the user with valuable real-time feedback by being very responsive and that encourages the user to try different solutions during the design.

Each *setXXXParameter* is used to set one of the parameters of the object. In the general case the point which is calculated from the pointer's position is used to calculate the fields of the *scale* TransformGroup (See Figure 7.13). Most of the

effects on the various objects can be achieved through a uniform or non-uniform scaling process. Using the example of the cone and assuming the XY plane as the horizontal one, the base radius can be changed through a non-uniform X-Y scaling (Figure 7.4-c) of the cone which was created with the default dimension (Figure 7.4-b). The height of the cone can be changed through a non-uniform scaling process on the Z-axis (Figure 7.4-d). Likewise the radius of a sphere can be changed through a uniform XYZ scaling process. Once the modification on the TransformGroup is done the new dimension is stored in the internal database for further retrieval. Once the parameter is confirmed by a click of the left mouse button the system performs the *postProcessXXXParameter* method. This routine confirms the value and prepares the system for the next cycle.

As mentioned, the user can interrupt the process at any time by clicking a different mouse button. The system then skips the remaining *setXXXParameter* methods assigns default values to the remaining variables and performs the finishing stage.

Once all the *setXXXParameter* and *postProcessXXXParameter* methods are called the system moves to the final stage where the *finish* method is called. This method turns off all the 3D-widgets previously used for the creation process.

Once the object is finalised the user can still change its state by selecting it with a right click of the mouse. This triggers the mechanism just described where the user modifies the object through the two TransformGroups (TG in Figure 7.13). The user can then move, rotate and scale the geometry uniformly or non-uniformly. The system then upgrades the values of the internal database to reflect the new status of the object.

This simple approach allows the changing of the dimensions of most geometrical primitives. Likewise the user can manipulate any other objects in the virtual world. In addition elements of the library (a staircase for instance) can also be manipulated using this mechanism. However, it must be noted that this approach does not modify the "topological" parameters of the objects. For instance, if the dimension of a staircase was scaled the number of steps would not be altered. This is

because this approach simply acts on the TransformGroups above the geometry and does not access the internal features of the geometry.

The only exception to this mechanism is found in the Wall object. This object is in fact the only real fully parametric object in JCAD-VR. Its geometry is not reconfigured through the simple alteration of its proportions but by directly manipulating each vertex of the geometry. This will be described in detail in the next section.

7.5 The Implementation of a Fully Parametric Object

The previous section described the common case of the creation of a generic SuperAECObject. Most of the research effort involved in the development of this prototype was focused on the interaction mechanism described in the previous sections. Consequently most of the geometries supported in JCAD-VR are created through a mechanism that simply modifies the proportion of the geometry.

However, to demonstrate the potential of JCAD-VR architecture and to prove that the framework illustrated so far could be developed far beyond the present level of its development in this prototype, a fully parametric shape (a wall) has been developed. This section shows how the general approach illustrated in the previous paragraphs could be extended offering support to real parametrical geometry.

It was decided to use the example of a wall to create a fully parametric object due to its unique features. A parametric wall requires not only the handling of its geometry but also the management of the sub-objects within its structure (doors and windows). Furthermore, the mechanism illustrated in the following pages includes the intelligent handling of the geometrical constraints generated by the insertion of those fixtures. The system proposed uses an algorithm that provides the means for the automatic creation of doors and windows compatible with the dimensions of the wall or with the number and location of various other fixtures.

7.5.1 The Geometry of a Wall Object

The implementation of the geometry of a wall had to take into account the fact that the insertion of doors and windows will change the geometry, which has to be pierced to house these fixtures. This makes it impossible to use any pre-determined or external file for its geometry. This required the development of a routine that can adjust the shape of the object at run-time.

Through a mechanism similar to the one described in the previous section the methods that handle the creation and setting of the geometry use the information on the position of the pointer to create and upgrade the geometry at run-time. Unlike the method described in the previous section, this is not done through the manipulation of the *scale* TG (See Figure 7.13) but by accessing the coordinates of each vertex of the wall and then upgrading the geometry for every frame.

The complexity of the algorithm being used is evident when a fixture is created. To do so the system abstracts the geometry of the wall by using points in the space. As illustrated in Figure 7.16 a wall is defined through four points

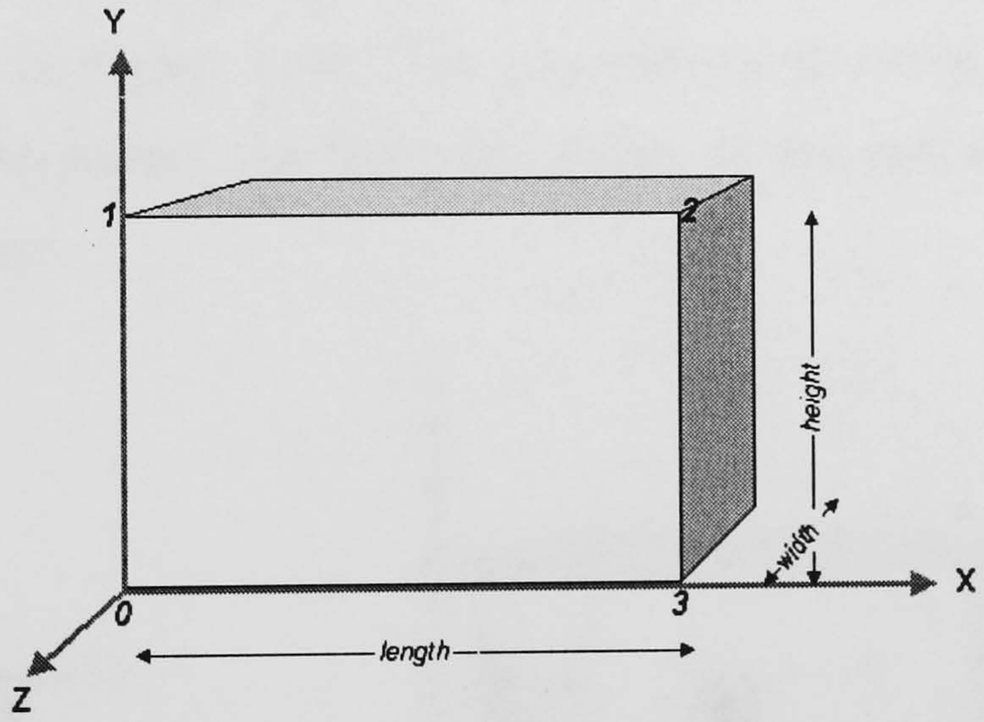


Figure 7.16: The abstraction of the wall's geometry through points in the space

The four points define the position of the wall in the space unequivocally, by its length and height. The remaining dimension, width, is provided independently through a number stored within the object. This is done because width, as will be shown later, plays a minor role in the dynamic reconfiguration of the geometry. These four points are then loaded into an array.

Whenever a new fixture is added, the geometry of the wall has to be recreated. To take into account the new layout, the window and door openings are represented through sets of four points each (as shown in Figure 7.17). As in the previous case these points are also loaded into an array, as a set of four, one for each fixture added to the wall.

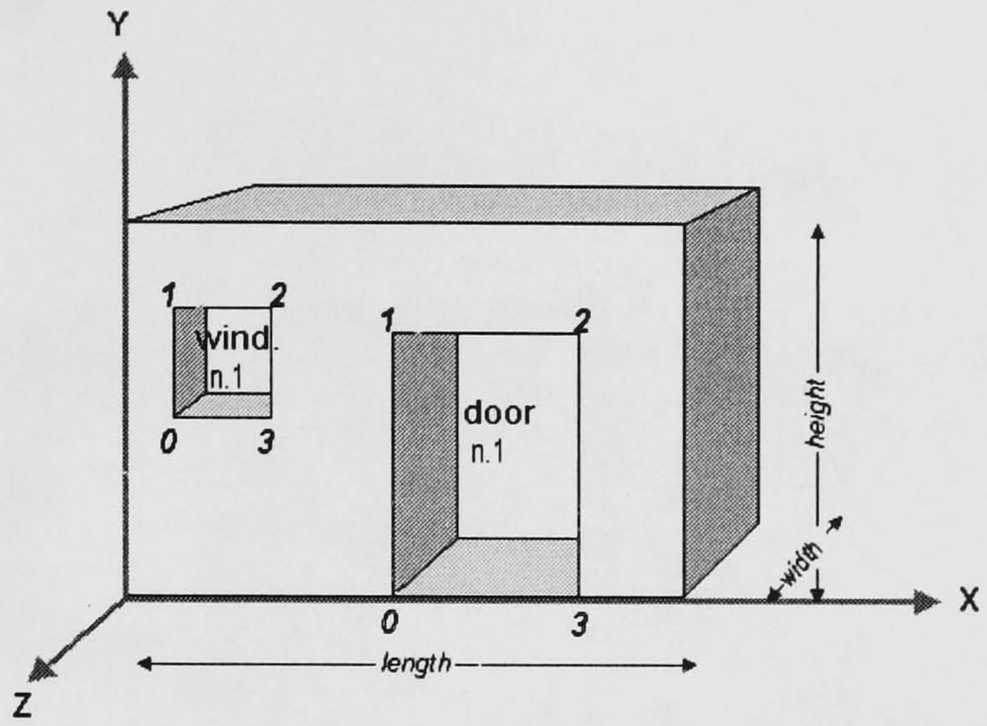


Figure 7.17: The abstraction of fixture openings through points in the space

The points, both those used for the openings and those used for the wall, are calculated relative to the same coordinate system which is the Wall local coordinate system (as shown in Figure 7.18). The user defines the origin of this reference system when he/she presses the left hand button of the mouse to place the new geometry in the space.

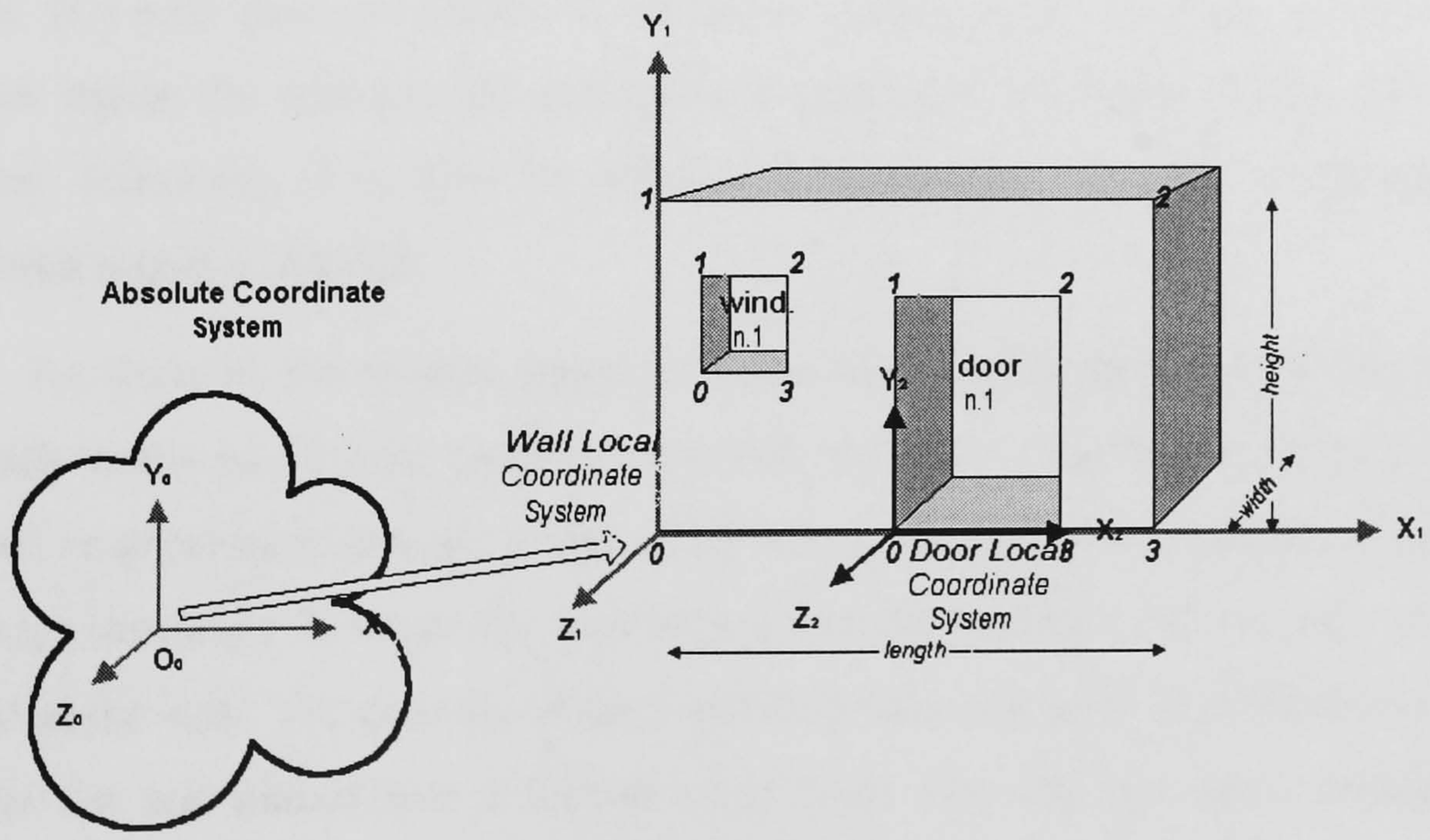


Figure 7.18: The relative coordinate system of the wall

Two lists (for doors and for windows) are used to store all the information regarding the wall and its fixtures in the arrays which contain the sets of points.

Precisely, the zero position of both lists is used to store the array containing the four points necessary to draw a wall. Every time a door or a window is created the points representing its opening are calculated and inserted in the relevant list as shown in Figure 7.19.

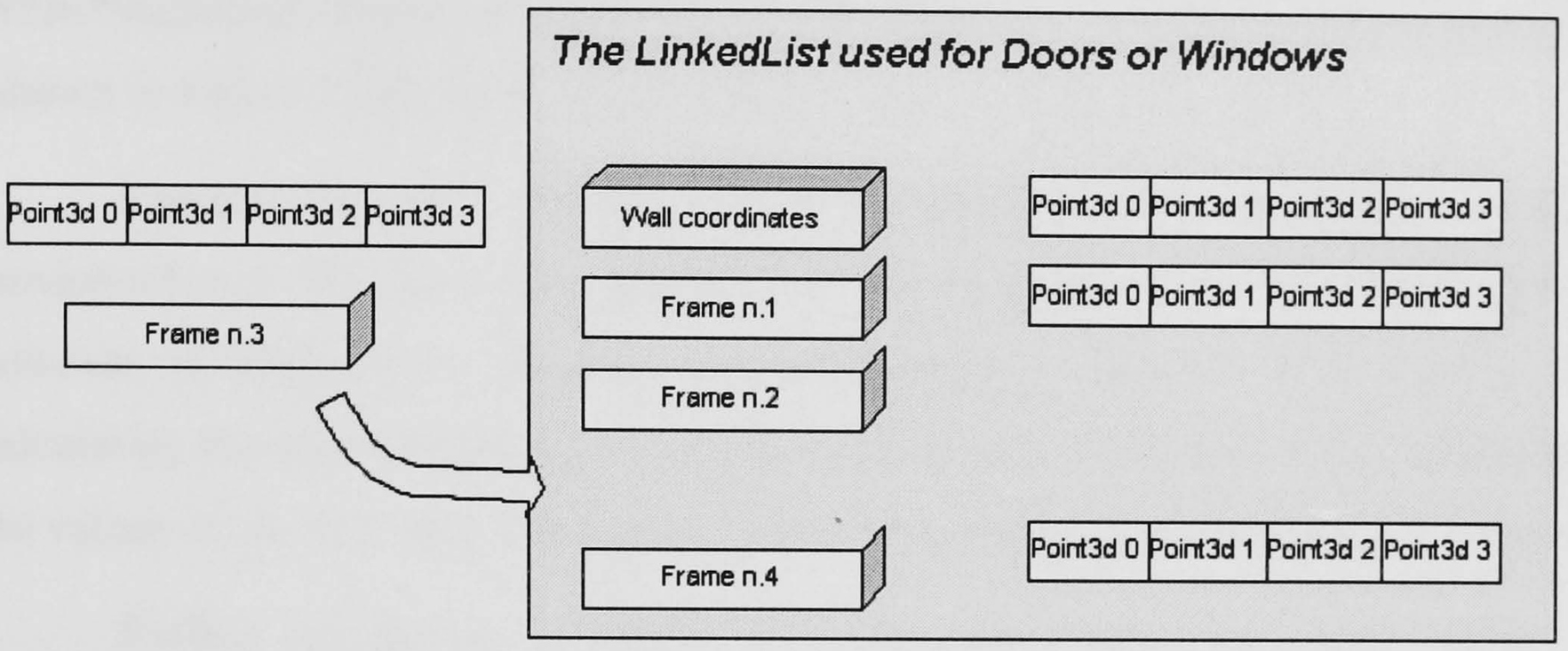


Figure 7.19: The insertion of a new set of points in the list

The arrays containing the sets of points are placed in the list in an ordered manner. Since both lists contain the array with the information on the wall at position zero, the new arrays are placed in ascending order from position one onwards so that the array at position one represents the closest fixture to the origin of the coordinate system. If a new door or window is created a routine places its array in the correct position inside the relevant list and shifts if necessary, the other arrays down one position. Likewise, if a door or window is removed, a routine rearranges the remaining arrays in the list.

As soon as the system needs to build the current geometry of the wall it cyclically retrieves the data from the two lists. This data, together with the width of the wall, is processed through an algorithm which calculates the number of vertexes and faces necessary to create the geometry given the number of doors and windows present in the wall. The position of each vertex is then extracted from the array in the relevant list and placed into a further array. This way the last array contains the ordered sequence of the x, y and z coordinates of every point necessary to create the

geometry. Through this array a number of strips of polygons are created which are placed in the correct part of the sub-graph of the object (as shown in Figure 7.13).

7.5.2 The Scaling Mechanism of the Wall object

Besides the different processes used to create its geometry, the Wall class differs from the other standard objects in the way the geometry is manipulated. While traditional objects are scaled or rotated using the two TransformGroups (TG) (shown in Figure 7.13), the Wall class manipulates the geometry directly.

Translations and rotation are still achieved by operating on the *moveAndRotate* TG above the geometry (shown in Figure 7.13). The scaling effect however is achieved by manipulating the geometry directly. This is done by calculating the new coordinates of the points representing the wall and by upgrading the values of the two lists. The newly calculated geometry then replaces the old one.

Further (as shown in Figure 7.13) the sub-objects are placed under a BranchGroup (BG) placed below the *moveAndRotate* TG. This means that every time the user moves the wall, the windows and doors are moved accordingly. However (as shown in Figure 7.13) the sub-object BG is not placed under the *scale* TransformGroup. This prevents the fixtures' size from being changed when the wall is scaled as would happen if the sub-object BG had been placed under the *scale* TG as shown in Figure 7.20.

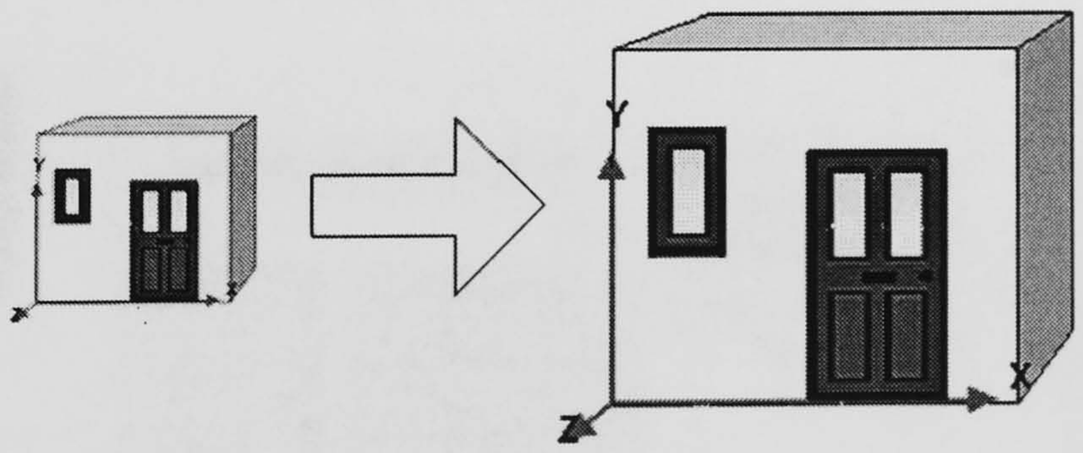


Figure 7.20: The result of the sub-objects being scaled with the wall

Since the geometries of the wall and fixtures are placed under separate nodes they can be scaled independently. However, if the scaling process of the geometry of the wall was done by using the *scale* TG (See Figure 7.13), every vertex of its geometry would be affected by the transformation. As noted, since the sub-objects

are not placed under the *scale* TransformGroup they are not changed in size. As a result, the wall together with the opening would be scaled, while the fixture would remain of the same size. This (as shown in Figure 7.21) would eventually result in a mismatch between the size of the opening and the fixture.

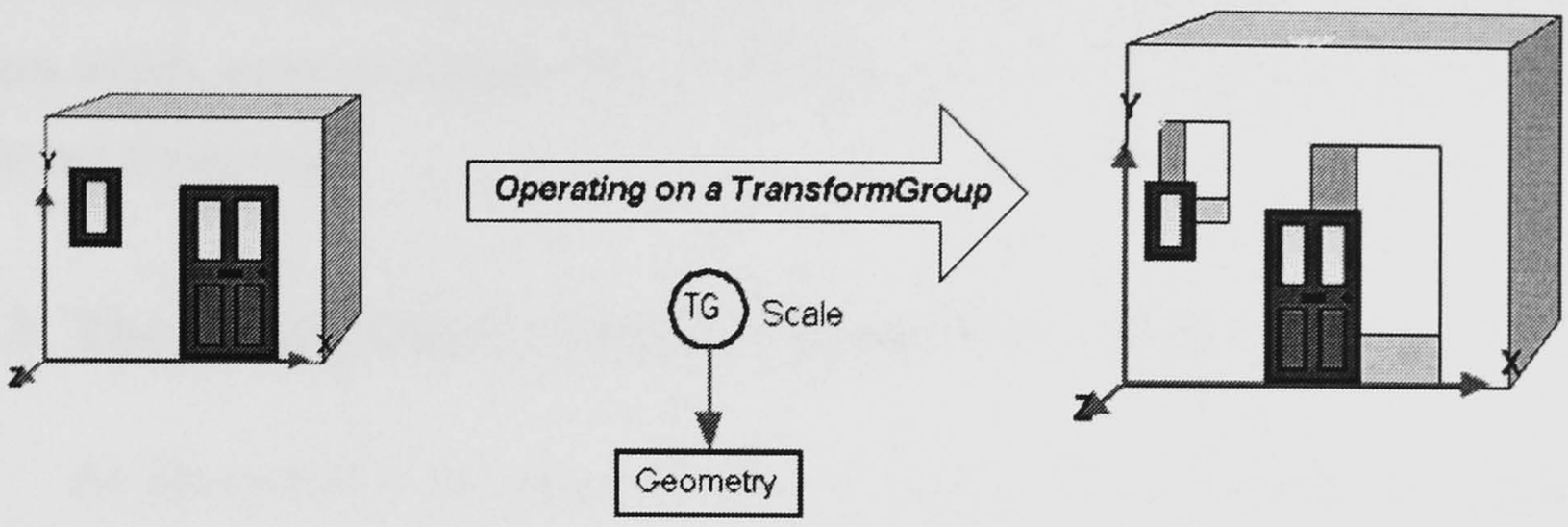


Figure 7.21: The scaling process through manipulation of the *scale* TransformGroup

The list-based mechanism developed in JCAD-VR allows instead the independent manipulation of each vertex of the geometry of the wall, including the vertexes used to define the openings. As a result, whenever the user changes the scale of the wall, the system changes only the coordinates of the vertexes used for the edge of the wall. In this way the openings size and position are not changed and as a result the user can scale the wall without affecting the size of the fixtures (as shown in Figure 7.22).

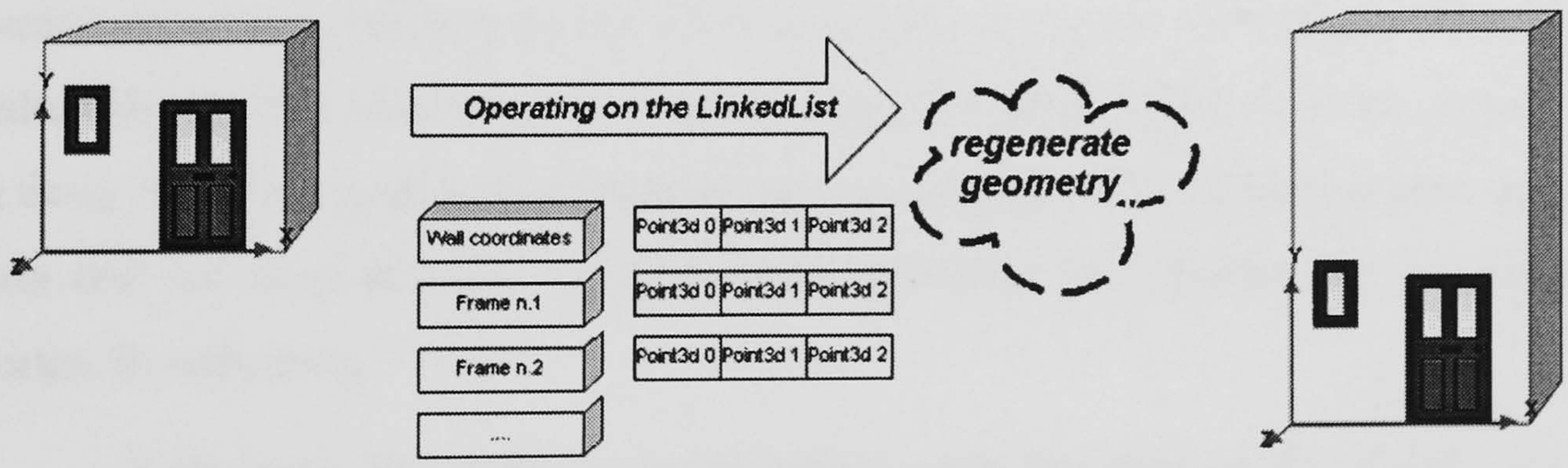


Figure 7.22: The scaling process through direct manipulation of the geometry

Likewise, every time the user modifies the dimension of a door or window the wall upgrades its geometry, changing the size of its opening to remain consistent with the new size of the fixture.

The drawback of this method is that every time the user changes the size of the wall or one of its sub-objects, the system has to calculate a new geometry. Further, in order for the action to be interactive the calculation has to be repeated for each frame and this creates an increased amount of information to be processed. However it must be noted that this does not adversely affect the performance of the system which, even on standard PCs, never demonstrates any visible reduction in the rendering frame rate.

7.5.3 The Constraint Handling System of the Wall

As illustrated in the previous section when a door or window is added to a wall its data is inserted in a list which is used as an internal database. Every time the geometry of a window is changed, this database is upgraded and then accessed by a routine that generates the new geometry according to its new values.

This approach allowed the development of a constraint check algorithm which ensures that every time a new sub-object is inserted or an existing one is modified, this does not conflict with the existing geometry. This is done through a two cycle algorithm that first checks if the proposed value is acceptable and then if compatible, upgrades the geometry.

Every time the user wants to change the dimension of a wall for instance, he/she selects it with a click on the right hand button of the mouse (as shown in the previous sections). The user chooses the "scale" command from the icon floating beside the object and then he/she starts dragging one of the 3D-widgets. The system calculates the new dimension and checks it against the database for every frame. This is done through a routine that extracts the coordinates of the existing vertex from the lists that are used to store the data of the geometry and checks that the proposed vertex fit with them.

If the new dimension is incompatible with the state of the object, e.g. the wall's proposed edge collides with an existing door or window then the choice is rejected. If the system finds instead that the proposed size is compatible with the state of the object then it inserts the new values in the database and upgrades the geometry.

The same process takes place when a new door or window is added to the wall. The system first checks if the dimensions proposed are compatible with the wall and with the existing openings. If the system finds that there is enough space for the fixture to be inserted then it upgrades the database and refreshes the geometry.

If instead the proposed fixture collides with the existing geometry its size is automatically reduced in order to fit the space available with an additional 30 cm around the opening. This process is illustrated in Figure 7.23 where the user creates a wall (Figure 7.23-a) and then he/she selects with a click the positions of a door and a window (marked with red circles in the figure). Each time the algorithm accepts the default dimensions (Figure 7.23-b) and places the fixtures in the wall after accessing the database.

Through another click, the user might try to create a second window between the first one and the door. The system finds the standard dimensions are incompatible with the state of the object and automatically adjusts the width of the window. This is done in order to leave a minimum distance of 30 cm between the window and the other two fixtures. The data is added to the database and the geometry is upgraded (Figure 7.23-c). The user then wishes to create a third window. The system finds that both the default height and width are not compatible and so it changes them in order to leave the 30 cm gap between the openings. The database is upgraded and the new geometry shown (Figure 7.23-d).

Likewise, if the user wants to move an existing fixture its new position will be checked first against the database. If the position is compatible with the state of the object it is accepted and the geometry is refreshed otherwise it is rejected and the old value is left. The advantage of this algorithm is that the user has immediate feedback on the action proposed. Therefore if the user starts moving a door and this gets too close to a window then the user visually perceives the door "colliding" into the existing fixture.

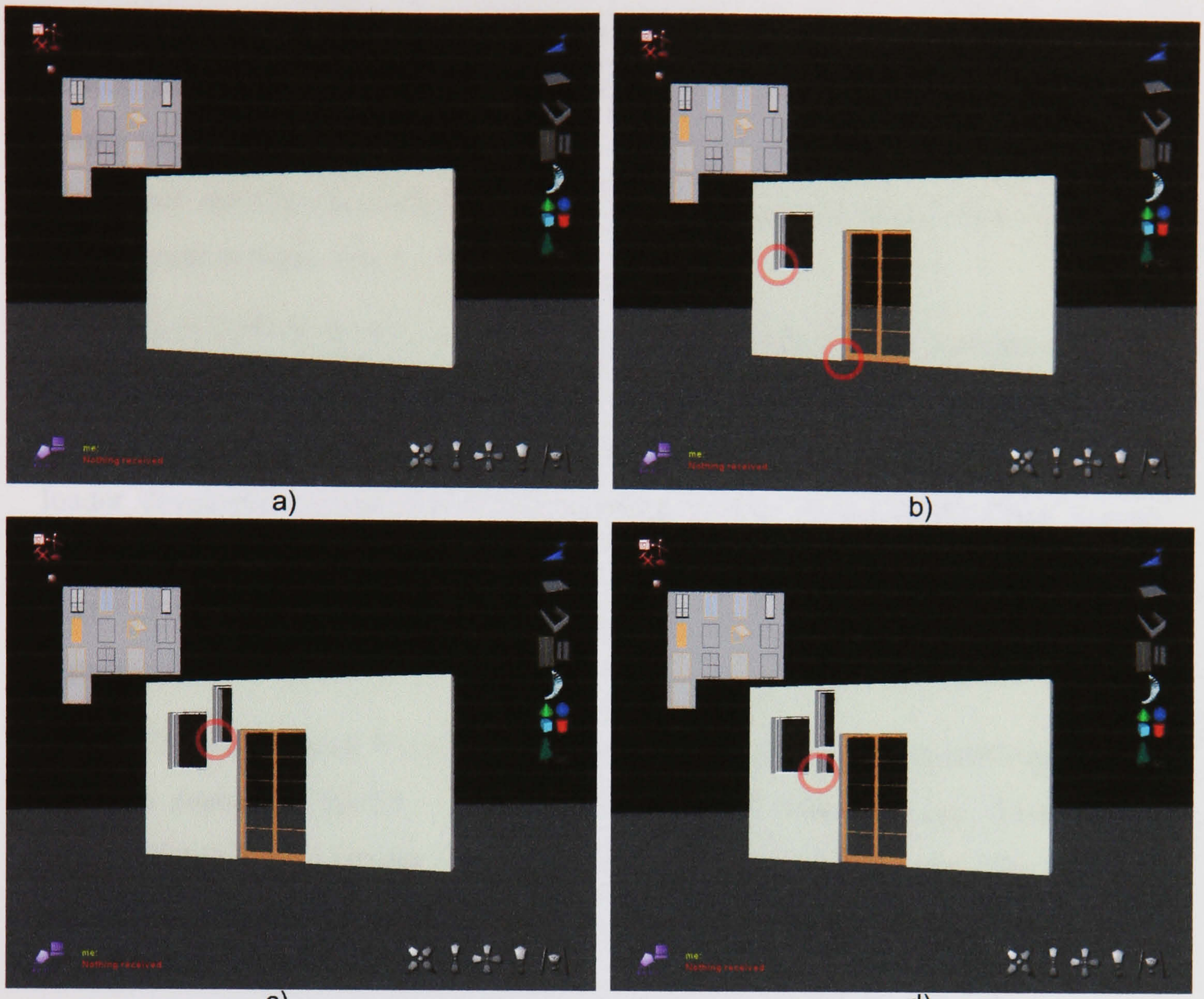


Figure 7.23: Constraint handling the creation of windows and doors

7.6 The Context Area of the Design

As described in the previous sections JCAD-VR can create a number of shapes. The user accesses the 3D-menu, selects the desired shape and then creates it by a sequence of click-and-drag actions with the mouse.

JCAD-VR also provides the means to retrieve information from an external file. This could define the context area which can be used as the base on top of which the geometries are created.

Therefore JCAD-VR allows the user to conveniently and rapidly provide a context area for their designs by importing 3D-models previously built through standard CAD packages.

When JCAD-VR is started, the user is asked to select a file to be used as the context of the virtual environment, if they do not then the system will start with an empty world. At present, the system can read VRML97 as well as "*.jcad" files, a file format specifically written to support JCAD-VR worlds (See Section 6.2.5.3 of the companion thesis Ucelli, 2002).

In JCAD-VR the GeometrWorld class provides the loading mechanism that is responsible for the creation and the handling of the objects imported from external sources. The GeometrWorld class, in turn, makes use of the VRML97Loader class, a loader developed by the Web3D Consortium as part of the Xj3D Open Source VRML/X3D Toolkit (Web3D Consortium, 2002). This is a library licensed under the GNU LGPL v2.1 (Free Software Foundation, 1999) and therefore it is freely distributable.

The loader reads VRML97 files and returns a Java 3D™ SceneGraph object and also converts VRML97 sensors into Java 3D™ Behavior class. This feature allows JCAD-VR to import environments that include animations, (for instance moving car or walking people) thus providing a more dynamic and realistic experience.

Figure 7.1 illustrates how the 3D-models necessary to render the context area are placed under the Environment BranchGroup, which is directly attached to the Locale node. This justifies the main difference between the nodes responsible for the geometry of the environment and the other objects created with JCAD-VR. The former are placed in a part of the DAG that does not belong to the *pickableBG* BranchGroup and therefore they are not included in the picking process (See Section 6.5.4.2).

As a consequence the user is not able to select the environment or to modify it. In this way the environment is clearly separate from the object being designed and this allows a simpler interaction with the system. In fact if the user were able to select the environment the overall interaction would become too complex since most of the picking action would return the object responsible for the environment. The user would consequently be forced to frequently cancel the environment accidentally

selected. Moreover the picking process would also process the geometry of the environment and this would ultimately yield poorer performances.

7.7 AvatarsBehavior

As visible in Figure 7.1 the layout of the DAG of JCAD-VR's Geometry Core includes the *avatars* BranchGroup. This is the node that contains the geometries used for the avatars. These are traditionally used in network based applications to represent the user's presence within the virtual world. In JCAD-VR avatars are rendered as human like figures made distinguishable from each other by having a 3D text of the users' login name beside them (as shown in Figure 7.24).

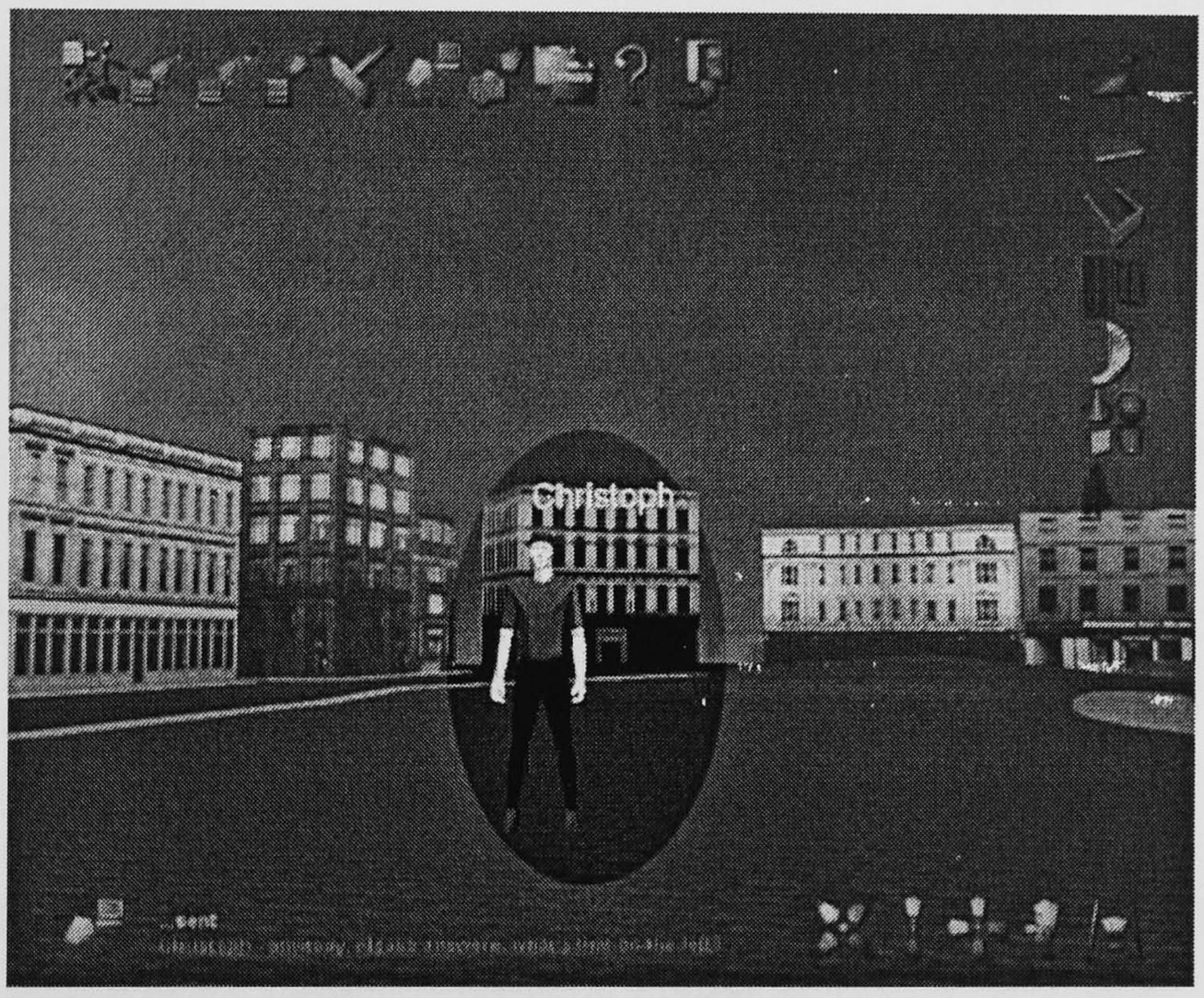


Figure 7.24: An avatar in JCAD-VR

The management of avatars is a process that greatly depends on efficient communication between clients and server. Every time a new user registers or leaves the system, or simply moves around the virtual world, a message containing the relevant information has to be broadcast to all the other users so that their client applications can upgrade the state of the world. At the same time the system has to retrieve information on the user's position and send it through the network to the server that eventually broadcasts it to every user.

In JCAD-VR the class responsible for this mechanism is called AvatarsBehavior and its internal functioning is illustrated in Figure 7.25.

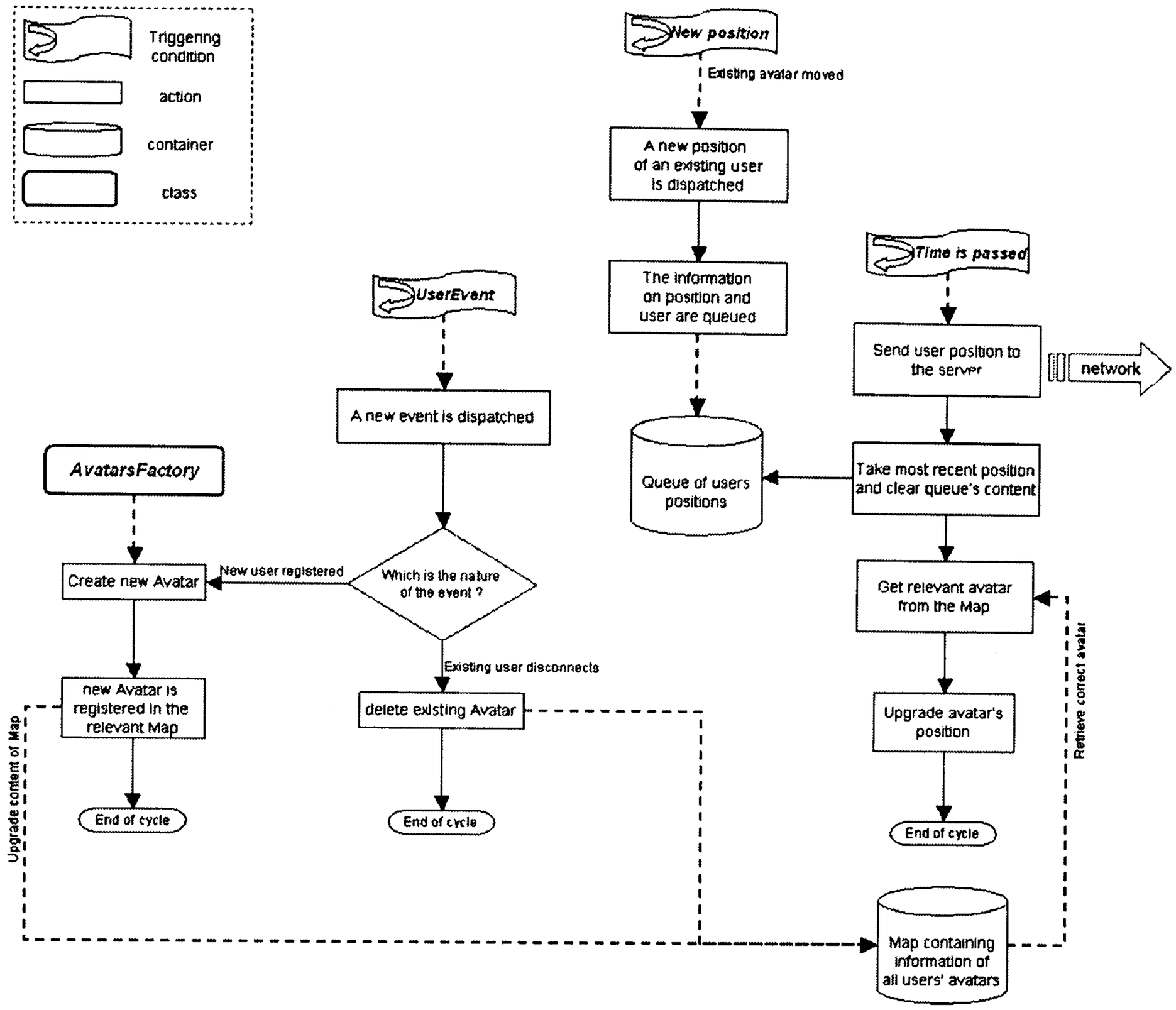


Figure 7.25: The AvatarsBehavior class

Every time a new user registers with the server or disconnects from it, a message is dispatched to every other user present in the virtual world. When each user's system receives the message it activates the AvatarsBehavior class.

Precisely, every time a new user registers to the system a new UserEvent is dispatched by the Network Core (See Section 6.2.1 of the companion thesis Ucelli, 2002). The *event/listener* mechanism (See Section 7.3.3) automatically invokes a routine that starts the creation or the deletion of an avatar, depending on the content of the UserEvent. Due to the *event/listener* pattern, at the same time in a different section of JCAD-VR, the same event triggers another class that starts the streams

necessary to upgrade the content of a new videoconference panel (See Section 6.5.3.3).

As shown in Figure 7.25, every time a UserEvent is fired the AvatarsBehavior creates a new Avatar object if the information contained in it indicates that a new user has been registered. The Avatar class contains the elements necessary to move the avatar within the environment as well as the geometry. However, the geometry of the avatar, similarly to the analogous process described in Section 7.4.3.1, is created through another class called AvatarsFactory.

During the start-up of JCAD-VR this loads the geometry from a 3D-model through the VRML97 importer (See Section 7.6) and stores it in the memory of the system. Every time a new Avatar object is created this requests a copy of the geometry from the AvatarsFactory, which returns a clone of the portion of the DAG containing the relevant nodes, including the geometry.

Once the geometry is retrieved the Avatar class uses the information in the UserEvent to create a 3D-text with the new user's login name, and this is placed above the avatar's head. Finally AvatarsFactory places a reference to the Avatar object into a Map. This is a logical container that is used to retrieve the reference to a specific avatar at a later time when for instance its position has to be moved.

As shown in Figure 7.25, if the UserEvent instead notifies that an existing user has disconnected from the server, thanks to the information contained in the event, the AvatarsBehavior class is able to retrieve from the Map the reference to the relevant avatar, which can be finally removed from the virtual world.

Figure 7.25 also shows that every time the class is notified of the new position of an existing avatar, the AvatarsBehavior class places the information in a queue. This technique avoids the overloading of the system since it allows the upgrade of the avatars' positions at a fixed frequency rather than every time a new position is received. Both the mechanisms dealing with creation/removal of avatars and the one managing the positions of the existing ones, access the Map using two independent and virtually parallel processes, or *threads*. For this reason the mechanism that controls the access to the Map is made *thread-safe* through

synchronization. This ensures that only one process at a time can access the information contained in the map and avoids potential conflicts.

Every 30 milliseconds the system reads the last set of information received from the queue, deletes the content of the queue, and uses the data to retrieve the relevant avatar from the Map and upgrade its position. At the same time AvatarsBehavior reads the position of the point of view of the user by accessing the *ControlView* TransformGroup (described in Section 6.4.1) and sends it to the network module for broadcasting. In this way every user is notified about the new position of the user and can upgrade the location of the relevant avatar.

7.8 Conclusions

This chapter completed the description of the 3D-Unit by illustrating the Geometry Core. It has shown the articulated process that leads to the creation of objects in JCAD-VR. The user creates shapes through mouse commands that are interpreted in a four stage mechanism whose details were provided throughout the chapter.

It has been illustrated how only the first stage in the process of recognition of mouse commands is directly connected to the pointing device from which it receives the events. The second and third stage process the information is used to change the state of the system. Finally the data is passed to the last stage where the action is fully decoded at the object level.

The result of this multi-level architecture is that other pointing devices can be easily supported through re-coding the first three levels. The last level is in fact independent from the interaction device used and it is used to directly manipulate the object. More pointing devices could be supported through the creation of a mechanism which could switch between different sections of code according to the device used in a similar to the visualisation sub-system described in Section 6.4.3.

The chapter also introduced the development of a fully parametric object, the wall, and illustrated how a JCAD-VR structure can be extended to include it

seamlessly. This proved that although JCAD-VR is a prototype it could be expanded and become a fully parametric collaborative VRAD system.

The chapter also illustrated how the system is able to import geometries from external files to be used as the context area of the design. The last section showed how the system handles the geometries of the avatars which are used to represent all the other users participating in a collaborative session.

The following chapter will introduce the experiment performed at the University of Strathclyde to test the capabilities and usability of JCAD-VR, and it will highlight a number of issues emerging from it.

8 Testing JCAD-VR: a Collaborative Session

8.1 Introduction

The three previous chapters highlighted the most important aspects of the JCAD-VR framework and described its internal architecture in detail. This chapter will report on the testing of the functionality of the software. An actual collaborative scenario was established to evaluate the effectiveness and ease of use of JCAD-VR in practise and the following sections will describe the experiment in detail.

The following section will also outline the *JCAD-VR TU/e Edition* of the software. This is the version of the prototype specifically customised for use in this experiment. This version features a number of mechanisms for monitoring and recording the progress of the design and communication processes.

8.2 The JCAD-VR TU/e Version

The *JCAD-VR TU/e Version* of the software was originally designed for a collaborative experiment between the Technische Universiteit Eindhoven (TU/e) in the Netherlands and the University of Strathclyde in Glasgow (See Chapter 9).

This version of the client application is enhanced with several features that help to monitor both the progress of the design and the information flow among the participants. In addition to the tools available in the standard JCAD-VR system, which have been outlined in the previous chapters, this version also features a number of routines specifically developed for the testing of the software, providing in particular:

- Automatic loading of a 3D-model of the context area chosen for the design task
- Automatic saving of the content of the VE
- Automatic capturing of screenshots during the session
- Automatic recording of the content of the communication by chat

The first feature makes the setting up of the environment for the collaborative session easier. As soon as the system is started a routine automatically loads the VRML97 file that was specifically chosen as the context for the design task rather than the user being asked to choose a file. This routine performs the loading procedure (described in Section 7.6) automatically. The participants are however free to load further 3D-models of virtual contexts by clicking the "load" 3D-icon and selecting the preferred VRML97 files.

The remaining three routines were developed to track the evolution of the design process throughout the experiment. The second routine specifically performs an automatic saving of the content of the VE in *jcad* files (See Section 6.2.5.3 of the companion thesis Ucelli, 2002) every 10 minutes. The files are then automatically stored in the "JCAD-VR\automatic_save\" folder with sequential numbers assigned (e.g. *file 1, file 2, file n*). In case of application failures this helps to prevent the loss of data and gives the ability to keep track of the progress of the design.

The third routine provides the means to progressively visualise the stages of the VE over certain intervals of time. Every 10 minutes this routine saves a screenshot of the scene in a *jpeg* file that is stored in the "JCAD-VR\automatic_save\" folder with a sequential number (e.g. *screenshot 1, screenshot 2, screenshot n*).

The fourth routine records the content of the communication through the chat and saves it automatically in a text file placed in the same directory as before. This allows the recording of the flow of information among the participants and it is a useful tool for checking the dynamics of the communication.

8.3 The In-House Collaborative Session

The *JCAD-VR TU/e Edition* system was tested in an experiment located at the faculty of Architecture of the University of Strathclyde in Glasgow. Three fourth year architecture students were selected for the experiment. They were all familiar and skilled in the use of CAAD packages but none of them had previous experience of Virtual Reality Aided Design (VRAD) systems. The students were first introduced

to the functions of JCAD-VR and then they were asked to participate in a collaborative session.

8.3.1 The Choice of the Configuration

It was decided to give the experiment a set of very precise and constrained conditions for a sample collaborative session. This required:

- The use of several different operative systems to test the multi-platform nature of JCAD-VR
- The simulation of a limited bandwidth
- The use of standard computer monitors as visualisation devices

Consequently two standard PCs, two Sgi computers, one Onyx2 and one O2 were used for the experiment. Their detailed hardware and software specifications are included in Appendix D. Specifically the two PCs and the Sgi Onyx2 were used for the client applications and the Sgi O2 was used to run the server. To ensure the network consistency in the unlikely event of all the three clients accidentally quitting at the same time (See Section 6.2.4 of the companion thesis Ucelli, 2002) the Sgi O2 was used to run a further client application remotely on the Onyx2.

In addition, in order to simulate having only a limited bandwidth available, the communication was restricted to the use of the chat and whiteboard tools. Furthermore, as illustrated in Section 6.5.3.3, due to the limited functionality of the Java Virtual Machine which was available for the Irix OS, JCAD-VR's videoconferencing module was not available on the Sgi computers. Therefore the decision not to use the videoconferencing module made it possible to use Sgi machines and at the same time provide all the students with the same tools. However, the video conferencing module had been informally tested during the introductory session, as shown in the video provided with this thesis (See Appendix C).

Finally it was decided that none of the students should use the Reality Center™ at the ABACUS Unit. As explained this was justified, by the purpose of the test which aimed to reproduce a realistic daily working scenario where architects

would be interacting from several different locations by using standard monitors as visualisation devices.

It is predictable however that the efficacy of JCAD-VR would improve if either the videoconferencing module or the Reality Center™ were used, although this has not yet been rigorously proven by further formal experiments.

8.3.2 The Training Session

An introductory session of only twenty minutes duration was held to show the students the features of JCAD-VR and to illustrate the tools accessible to them from its 3D-GUI.

The students were already aware of the basic concepts of VR and had some experience with Desktop-VR. As mentioned earlier all of them were skillful in 3D modelling. However, none had previous experience in using either VRAD systems or collaborative applications and none had used the JCAD-VR software before.

During the introductory session students showed immediate interest and quickly demonstrated that they comprehended the simple logic behind the 3D-GUI. After the short introduction they all proved to be quite confident with the system.

8.3.3 The Design Task and the Outcome of the Experiment

The three students were located with their own workstation (See Appendix C) in three different rooms within the Department of Architecture. Fixed and hand-held video cameras constantly filmed their behaviour (See Figure 8.1).

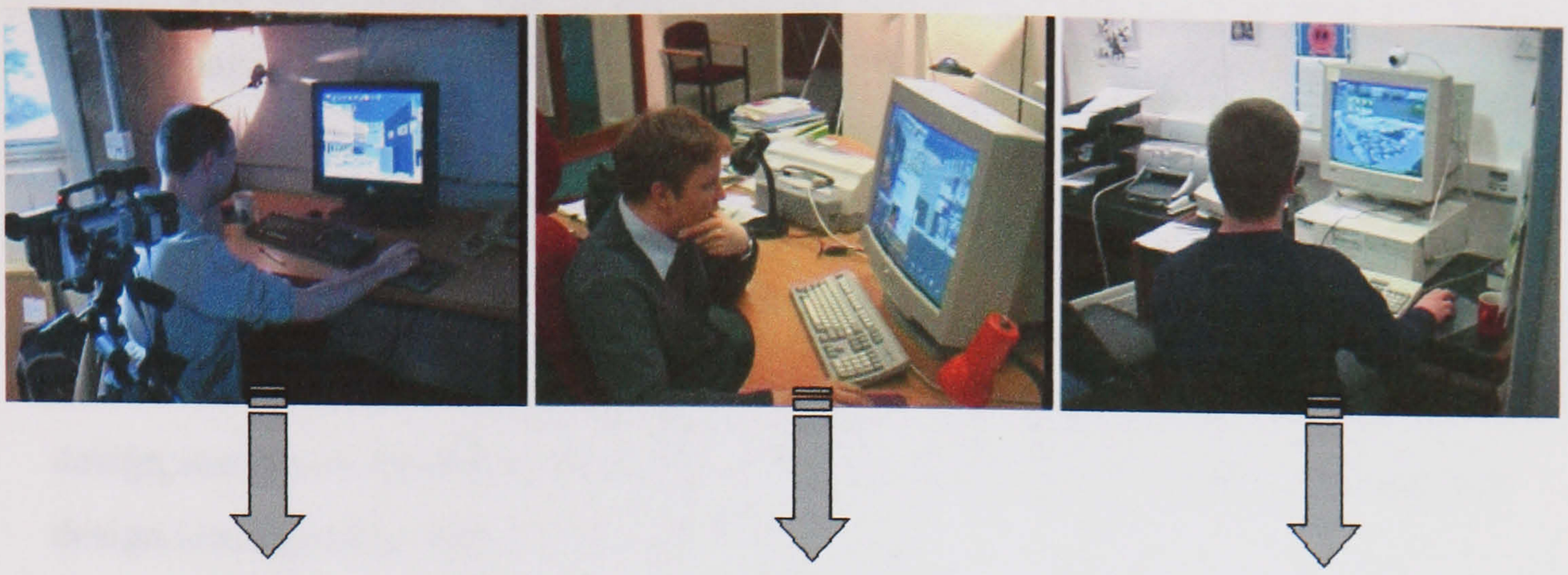


Figure 8.1: Images of the experiment with the students working on PCs and an Sgi Onyx2

None of the students were previously aware of the design task being set before they were informed at the beginning of the experiment. In this way the students did not have the chance to previously agree on design choices. Instead they were forced to only use the communication tools available within JCAD-VR to discuss their possible design solutions.

The three students were invited to design an information centre over the two-hour design session. The context area was already provided and it was loaded automatically by the system when it started. It consisted of a public square adjacent to a park.

Observing the video made during the session (See Appendix C) and studying the content of their communication through chat (See Appendix A) revealed several noteworthy aspects of the experiment.

First of all the system proved to be quite easy to use. Students did not have any particular difficulty in working with JCAD-VR and were able to use the system immediately after the 20-minute long training session. As mentioned in Section 7.4 due to a fault in the Java Virtual Machine for Windows OS, navigation with the PCs resulted in a slightly more complicated arrangement than originally planned. This resulted in the two students who were using PCs having to click on the relevant icon to switch to navigation mode rather than pressing the central button of the mouse. However, both students got accustomed to this after a few minutes.

The experiment was characterised by intense interaction amongst the three design participants. The students could communicate efficiently through the chat (See Appendix A) although they all stressed the potentially positive effects of having a more direct means of communication, such as through voice or by a video conferencing tool.

The students also felt it was important to be able to draw sketches during the design session on the digital whiteboard provided. This helped them to express their design ideas quickly, clearly and efficiently.

The students also remarked on how the collaborative approach led to a real sense of teamworking. Several design scenarios were in fact discussed, modified and agreed on within the common design environment. Interestingly the shared design system was a cause of satisfaction as well as frustration among the participants. It was noted that students frequently changed or modified their own design solutions as well as everyone else's. This raised an interesting issue: the students suggested the development of some form of priority routine through which each user could decide whether the object being created could or could not be modified by other users.

Finally from the technical point of view the experiment highlighted the lower stability of the system on the Sgi platform if compared to PCs. The Sgi version suddenly stopped running three times while the PC versions were stable for the entire duration of the experiment. This was the result of the lower efficiency of the Java Virtual Machine for Sgi and the higher level of optimisation reached on the PC platform. As already mentioned (in Section 5.6.1) a higher priority was given to achieving efficiency of the system for the PC platform. This choice was justified by the more widespread use of PCs compared to Sgi set-ups.

In addition to the sequence of screenshots taken during the experiment (See Figure 8.2 and Appendix B), a video camera was running in each of the rooms filming the students. The collaborative session was then fully documented by a video that was later edited in order to show the most significant moments of the experiment.

Figure 8.2: Screenshots of three different stages of the design during the experiment

The video is included in the thesis in Appendix C. It shows firstly the introductory session when the authors introduced JCAD-VR to the students and then the most relevant moments during the collaborative part of the experiment. The video also illustrates the final outcome of the design task (See Figure 8.3) and shows the students comments on both the JCAD-VR software and the collaborative session that were recorded at the end of the experiment.

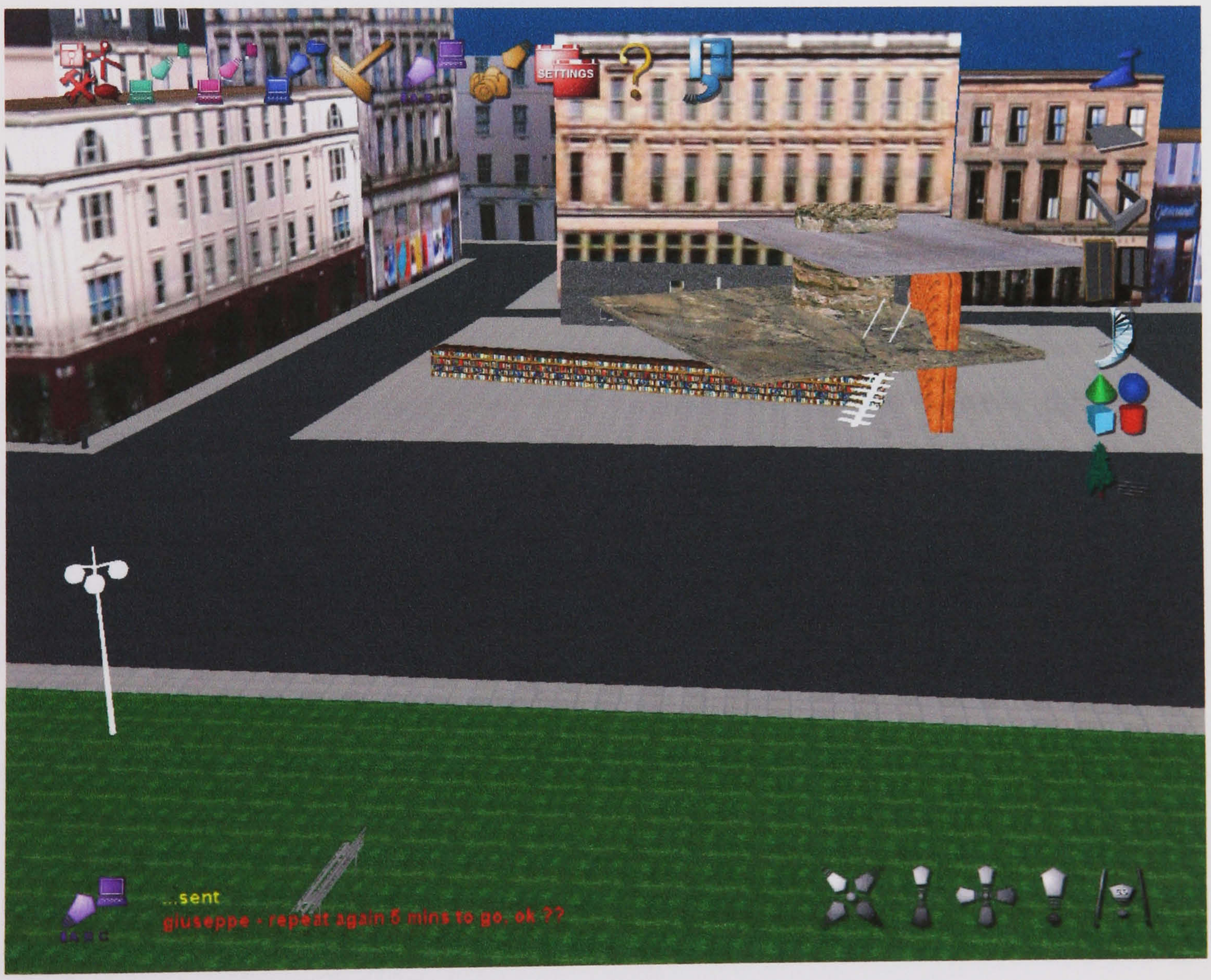


Figure 8.3: The final design of the collaborative session

From the students' interviews at the end of the experiment some suggestions emerged for further enhancements of the prototype. In particular they recommended the implementation of two other tools that would ease the dynamic of the collaboration: in their view a tool to help with the orientation in the VE, and a 2D map that would show from above the positions of all the participants in real time.

8.4 Conclusions

This chapter outlined the most important elements of the practical testing of the JCAD-VR system. The outcome of this limited experiment entirely proved that JCAD-VR, even at this prototype stage, is a fairly stable application that efficiently allows multiple participants to take active parts in a collaborative design session while communicating across the network.

Thanks to the specific features, which were implemented in the JCAD-VR TU/e Edition, the experiment could be recorded in a number of ways:

- The sequence of screenshots (See Appendix B)
- The sequence of jcad files
- The text of the chat (See Appendix A)
- The video of the experiment (See Appendix C)

The positive outcome of this experiment will certainly stimulate further trials and tests, such as the setting up of a collaborative session between different institutions possibly between different countries, building on the in-house experience reported.

Chapter 9 will outline the contributions that this thesis, together with its companion thesis, has brought to the field of research into design in VR and Collaborative Virtual Environments. It will also provide the conclusions and suggest the future for the JCAD-VR framework.

9 Summary, Contributions and Further Work

9.1 Introduction

The previous chapter highlighted the methodology followed to test the JCAD-VR system, which provided useful feedback on the application.

This last chapter will provide a summary of the JCAD-VR project and will also outline the positive contributions that the development of the software prototype brought to the field of architectural design. This was achieved due to the development of the features typical of Virtual Reality Aided Design (VRAD) and Collaborative Virtual Environments (CVE) systems.

Conclusions to this work will be proposed along with plans for further work and developments to improve the system.

The section describing further works will in addition describe the methodology and preparation of an experiment at a much larger scale which will involve students from the University of Strathclyde in Glasgow, and the Technische Universiteit Eindhoven in the Netherlands.

9.2 Summary

This research project fulfilled four main tasks (See Figure 1.2). The first task was the research study that allowed the identification of the users requirements and also provided an overview of the basic characteristics of VRAD (See Chapter 3) and CVE (See Chapter 3 of the companion thesis Ucelli, 2002) systems, along with an attempt to categorise them into taxonomies.

The second task was the development of the JCAD-VR framework. This phase contained the following four sub-tasks:

- **Object Oriented Approach**. The choice of the OO approach is explained with separated modules to address different functions of the system and to allow

concurrent development of the system prototype. This led to the choice of Java™ as the programming language for the project.

- **Systems specifications and requirements**. The definition of the characteristics necessary for a collaborative VR system to allow synchronous collaboration and communication while designing in the VE.

- **Identification of the systems components**. The definition of the most important functions of the system and of its two main components, the VR design environment and the collaborative platform.

- **Separation of the competency in the two main themes**. The definition of the sub modules belonging to the VE and to the collaborative platform and the consequent separation of the competency. During this phase the final framework for the JCAD-VR prototype was defined.

The accomplishment of the third task was the implementation of the JCAD-VR system, and this was achieved through the combined efforts of the two authors. This was first done by choosing software packages and libraries which were compatible and by completing their integration into one application. Secondly, by the concurrent development by each of the authors of relevant modules of the framework and by checking their successful integration at the most crucial stages of the development.

The fourth task was the testing of the final prototype. An actual collaborative design session was conducted and documented using several different media. The results of the test were collected and reported in Chapter 8.

9.3 Contributions

JCAD-VR is an original application specifically designed to improve the architectural design process. The project embraces different aspects of the development of VR systems for collaborative design environments tailored for the conceptual design.

JCAD-VR brings several original contributions to the design process by allowing the earlier use of VR technology and by providing practitioners with an

effective and easy to use collaborative tool. Section 9.3.1 lists the advances in design methodology which were achieved by the implementation of JCAD-VR and its Human-Computer Interface. Section 9.3.2 illustrates the contributions offered by the development of both the system's collaborative platform and communication tools. Finally Section 9.3.3 outlines the original contribution that the implementation of JCAD-VR brought to the development of the first example of a new generation of Computer Supported Cooperative Work (CSCW) system (See Section 2.4 of the companion thesis Ucelli, 2002) called the Collaborative Virtual Design Environment (CVDE) (See Section 3.4.5 of the companion thesis Ucelli, 2002).

9.3.1 Effective Design in VR

As mentioned in Section 3.2 traditional CAD systems are not suitable for using in the early stages of design. They are specifically made for the engineering stage and therefore do not encourage experimentation. For this reason architects usually tend to use traditional media and make the shift to digital tools at a later stage. VR is traditionally used as a visualisation tool rather than a design instrument.

JCAD-VR promotes instead, the earlier use of VR thus exploiting its superior visualisation capabilities to convey a new design experience. Through JCAD-VR the user is encouraged to experiment with several design solutions done quickly. This system makes VR available to architects and at the same time it promotes their creativity.

9.3.1.1 VR-Based Conceptual Design for Architecture

JCAD-VR, albeit at a prototype level, is a tool specifically designed to support conceptual design. The interactivity of the system helps the user experiment with the design. Specifically the system has been tailored to the needs of an architect. The architect can create a number of simple shapes, such as geometric primitives, as well as architecture-related objects. In addition a library provides a number of ready-to-use 3D-objects.

Objects are created in a very simple way. The user does not need to type in values but they use mouse commands to create and change geometries.

The user can conveniently import a 3D-model of the context area and can then start designing on top of it. In this way, the user can rapidly create several design prototypes and evaluate their qualities within the real context of the design.

JCAD-VR can be considered a true VRAD system for architecture since:

- It provides real time interaction
- It hides the mathematical description of the geometries
- It encourages experimentation and creativity through a responsive visual feedback
- It helps the architect to adopt digital tools early in the design process
- It helps architects by using new technologies that can enhance the efficiency of the design process
- It promotes a user-friendly environment that does not require special expertise

9.3.1.2 User-friendly Human-Computer Interface

As shown in Chapter 6 JCAD-VR features a user-friendly Human-Computer Interface (HCI). As shown in Chapter 8 an experiment proved that typical users can become familiar with the software within a few minutes.

Users can navigate easily within the virtual world in a way that resembles a real-life experience. As shown in Section 6.5.2.1 navigation has been constrained to avoid the confusion of most VR systems. This has been done by emphasising the separation of horizontal and vertical translations and limiting rotations to those around the vertical axis.

However, interaction with the system is not only limited to navigation. As shown in Chapter 7 the user can create a number of different types of shapes through a 3D-GUI.

The adoption of the 3D-GUI brings the intuitiveness of traditional 2D-GUIs to JCAD-VR's HCI. In this way the user can control the system through an HCI that is part of the virtual world. This is done through a number of 3D-menus and a set of

tools used to communicate with other users: two 3D-panels which show images from the videoconference and the text of the chat.

For the convenience of the user, any element of the 3D-GUI can be moved, scaled or rotated just like any other object in the virtual world.

JCAD-VR proved to be easy to use even with a conventional 2D mouse. The entire system has been engineered in order to be intuitive yet effective. The user can move, rotate or scale objects in a very simple way. The user selects the object, without need for typing in values and manipulates it using a choice of 3D-widgets. This makes the interaction less error-prone, user-friendlier and very visual.

9.3.2 The Flow of Information between the Participants

As mentioned in Chapter 2 of the companion thesis (Ucelli, 2002) the conventional design process often limits the collaboration between groups of practitioners to an inefficient and fragmented exchange of information.

JCAD-VR instead allows a simple and reliable flow of information between the participants in the design process and provides the means for fast communication of design ideas through a number of different communication methods. In traditional offices the collaboration among the practitioners is very often still accomplished orally and through the exchange of paper based documents such as sketches and plans. In contrast JCAD-VR offers an integrated, synchronous and fully digital means for collaboration and exchange of design choices.

9.3.2.1 Design Information Exchange

JCAD-VR provides the means for establishing an effective design information exchange among remote participants and offers a major contribution towards the creation of a distributed virtual architectural office. It supports synchronous collaboration and communication of design ideas by allowing several participants to interact with virtual objects. It provides the users with a virtual design arena that encourages the collaboration of remote design teams through the possibility of direct interaction with all the virtual elements present in the shared VE.

It also enhances the communication by offering a choice of several different tools that enable the flow of design ideas among the participants.

9.3.2.2 Communication Methods

JCAD-VR provides the user with various communication methods. Its modular and flexible architecture allows the implementation of a number of communication tools that can be easily combined or used separately according to either the network bandwidth, the power of the computers or the users needs. The users can choose between the three following configurations at their convenience:

- Video, audio, chat and whiteboard
- Audio, chat and whiteboard
- Chat and whiteboard

JCAD-VR also provides the means for fully collaborative experience by the use of avatars as virtual embodiments of the participants. These communication features, along with support for the synchronous exchange of design information through a shared VE, are the most significant contributions brought by the JCAD-VR prototype to the field of architectural design.

9.3.3 Integration into a Collaborative Virtual Design Environment (CVDE) System

JCAD-VR combines features typical of both CSCW systems and VR environments. From this point of view it is an original software prototype that can be considered a successful attempt towards the integration of the VR application and the CSCW means of communication, into a comprehensive collaborative tool. In addition JCAD-VR owns all the characteristics of the new CVDEs (See Section 3.4.5 of the companion thesis Ucelli, 2002) and it can be considered one of the first examples of this category of system. It supports:

- The establishment of a distributed virtual office for remote participants
- Synchronous communication and interaction

- Three-dimensional imaging and simulation
- Interactive collaboration based on virtual presence
- Ease of creative conceptualisation
- Flow of information and the sharing of knowledge
- The sharing of knowledge and expertise for virtual enterprises.

Lastly, as mentioned in Section 4.6 of the companion thesis (Ucelli, 2002), JCAD-VR is not a complete and fully optimised software package rather it has to be currently considered as the unpretentious proof of a concept. Its contribution then is in offering a valuable starting point towards the development of a finished commercial package.

9.3.4 Extendable Architecture and Portability

As mentioned in Section 4.4.3 of the companion thesis (Ucelli, 2002) JCAD-VR was developed in a modular fashion enabling the scheduling of the implementation of independent self-functioning modules and allowing concurrent software development. This open architecture gives developers the freedom to improve the system by implementing further tools or routines at any time according their needs. From this point of view JCAD-VR inherits the qualities of the object-oriented approach that Java™ offered by providing easy extendibility, maintainability and portability to several different platforms.

9.4 Further Work

As mentioned earlier in this chapter, JCAD-VR is based on a flexible and expandable architecture that can be easily improved with newly developed tools and routines. During its development a priority list was established to set the order for the implementation of the modules of the framework.

This procedure led inevitably to the selection and implementation of some modules while others with less priority were left for further development. These lower priority modules, (marked in black in Figure 9.1), will be given in this section

as examples for further development of the JCAD-VR system. Valuable suggestions for the implementation of new tools were provided by the previous experiment described in Chapter 8. This provided a valuable occasion to collect significant feedback about the software prototype.

The following sections will outline further development of the system and it will finally describe the methodology and preparation of a cross university test that would require the involvement of a number of students from two schools of Architecture.

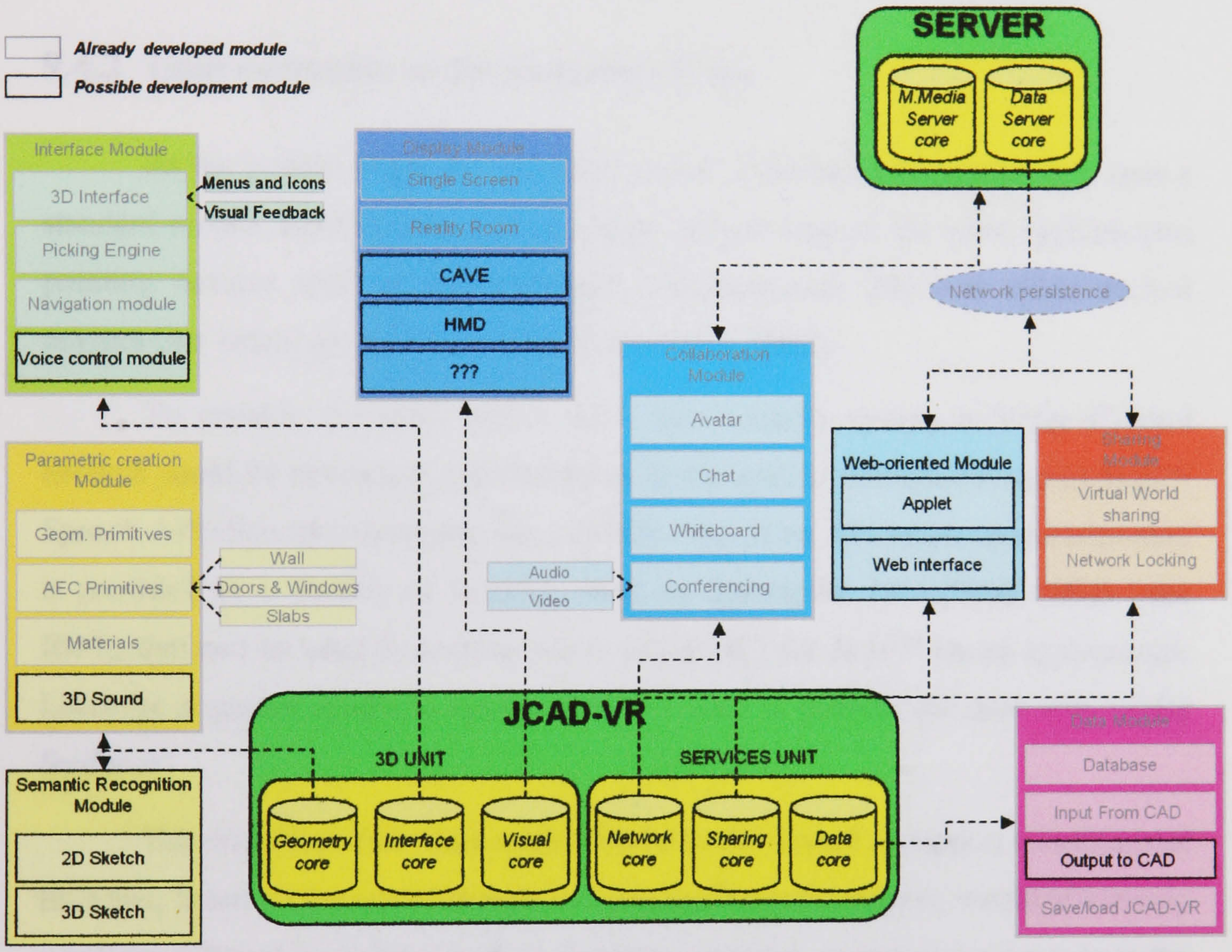


Figure 9.1: The JCAD-VR framework with the unimplemented modules marked in black

9.4.1 Enhancements to the Visual Core

Currently JCAD-VR supports only monoscopic visualisation using traditional monitors or a Reality Center™.

However, (as demonstrated in Section 6.4.3.3) the display core was developed in order to be easily expandable. The successful development of the code necessary to support stereoscopic views and tracking devices would lead to the support of more visualisation devices including CAVEs, Virtual Tables or HMD.

Additionally, support for multiple screen desktop based configurations could also be supported. In this case, the development of a "wizard" that would allow users to configure the system at start-up, could also be included. Users could configure a simple semi-immersive environment quickly, according to the number and position of screens available.

9.4.2 Improvements in the Interface Core

At the present stage the interface module (illustrated in Section 6.5) uses a standard mouse. Further development would include support for more sophisticated pointing devices such as SpaceMouse® (3DConnexion, 2002) or other tracked devices like virtual gloves (Immersion Corporation, 2002).

To provide the user with a more user-friendly system a Voice Control Module could be developed and interfaced to the system through the use of Java™ Speech API (Sun Microsystems, Inc., 2002e). This is an API whose implementation is provided by a number of vendors, (such as Speech for Java (IBM, 2002) from IBM), that can be used to develop voice driven HCI for Java™ based applications. Likewise a speech synthesis system could be used to provide the user with useful feedback.

The entire Interface Core could also be re-developed to support a multimodal HCI that, based on the use of immersive visualisation technologies, would effectively combine different modalities such as sketches, gestures, speech and gaze to help the designer accomplish tasks in a natural user-centred way. In fact, as cognitive scientists have proved, the design experience strongly benefits from the support of multi-sensorial, or *multimodal*, interactions.

Different *modalities* can be considered as complementary conceptual channels that can transmit information not easily acquired spatially. One of the main advantages of the integration of different modalities lies in the widened perceptual

and conceptual bandwidth available to the user to convey information regarding the object he is creating. Such integrated approach is founded upon the effective support of human communication patterns that can provide, if combined, spatial description and mutual interrelation hardly achievable through other means.

This way JCAD-VR could be thought of as a design tool that would allow the designer to freely sketch and manipulate 3D-shapes in a transparent fashion within an immersive Virtual Environment (VE) augmented through natural interactions such as speech, gesture or gaze interpretation.

Students involved in the experiment (described in Chapter 8), proposed further improvements by the development of a panel showing a plan view of the virtual world. This would be used to provide an overview of the environment as well as to indicate to other users where a specific object is.

9.4.3 Further Developments of the Geometry Core

As illustrated in Chapter 7 the JCAD-VR Geometry Core provides the means for the creation of a number of shapes. It has been also proven how the development of fully parametric shapes can be integrated into the existing framework. Therefore, further improvements should include the development of more parametric objects. A number of existing objects, such as stairs or furniture, could also be made fully parametric.

The library of objects could be extended to include more objects and more materials. This could also include lights and pre-recorded sounds such as people chatting and street noise etc. In this way users could quickly create specific environmental conditions in the VE.

The system could also be extended by implementing Boolean functions. This could be done in Java3D™ or simply interfacing the Geometry Core with an external modelling kernel.

Another module could also provide support for semantic recognition of sketches. Similar to the Space Pen system (Jung et al., 2002, described in Section 3.4.2.5) users could sketch shapes directly into the virtual environment. However in

JCAD-VR, the shape once recognised could be replaced by the relevant 3D-model, which could be sent to the other users through the network.

9.4.4 Enhancements to the Data Core

As described in Section 7.6 currently JCAD-VR loads files created with external CAD packages through a VRML97 loader (See Appendix D.5). In addition the system can also save and retrieve the content of the virtual world in jcad format. Further development would include support for more file formats. This would facilitate the integration of JCAD-VR with the existing workflow.

9.4.5 Enhancements to the Sharing Core

Enhancement to the Sharing Core would include the development of an Applet version of JCAD-VR. This would allow the system to be run inside an Internet browser. This would facilitate the use of the system since it would not require the user to install the system. This would be automatically launched once the relevant web page was loaded.

However, due to the actual size of the software (approx 50 Mb) further optimisations and a modular download system should be developed. The applet would start the download of a central core and then access the relevant sections as required. To partially solve this problem the jar archives could be cached inside the user's hard drive the first time the system is loaded.

9.4.6 Enhancements to the Collaboration Tools and Network Architecture

The network architecture of JCAD-VR could be improved by adding new collaborative tools and by optimising its transmission routines. The video conferencing system is not optimised and can result in being too demanding in terms of bandwidth usage when more than two clients are connected to the system. Under these circumstances communication could be allowed by the transmission of only the audio channel and by replacing the video stream with a 3D representation of the

participants simulating the oral communication through the movement of the mouth and facial expressions.

In the standard version, the JCAD-VR Server offers support to only one VE at a time. However, it could be modified to handle multiple worlds concurrently. At the moment this can only be achieved by launching several JCAD-VR Server applications on different computers.

The information flow about the virtual objects would be improved by optimising some of the routines for the transmission of data. For instance, at present every time the user deletes an object in the VE, the network again transmits all its parameters to the remote participants. Instead, the same information (the object being deleted) could be broadcast transmitting only the Deleted Status and the object's ID number. The same simplification could be applied in the case of the user wishing to modify of only a few of the objects parameters. This would partially reduce the amount of information that has to be transmitted across the network.

Section 6.2.2.1 of the companion thesis (Ucelli, 2002) described in detail the array used to transmit the parameters of virtual objects. This array contains some empty positions that could be filled with extra information about the existing objects or with parameters of newly implemented objects. The same array could be elongated to accommodate other relevant information if required.

In JCAD-VR the embodiment of a user is allowed through only one 3D model resembling a human. This is used together with 3D text showing the login names, to differentiate the participants of the session. The virtual scene could be enriched by providing the possibility of using several different 3D-models for the avatars. This would allow the possibility of choosing among embodiments resembling women or men, children, wheelchair users or simply using abstract models. For this purpose the ClientInFrame panel (described in Section 6.2.1.6 of the companion thesis Ucelli, 2002) contains an empty area for the avatar tool, which could be filled with a list of models from which users could select their preferred virtual appearance.

During the test (described in Chapter 8) students suggested the implementation of a tool that would allow users to point at directions and areas in the

VE while discussing design solutions. According to the students this would provide a clearer reference for the discussion. In this way users would point at an object, stating for instance that it was "North" from the square, or "East" from the red house. This issue would be solved by simply including in the virtual scene an object acting as an absolute reference for directions, e.g. a virtual compass, indicating North, South, East and West. As an alternative the system could include a routine that would allow users to point at virtual objects visually and use appropriate visual marks to annotate the object in the VE.

Students also suggested the implementation of a more sophisticated Locking Mechanism than the one included in JCAD-VR, which would still allow the possibility of choosing whether an object can be edited or not by other participants. This routine should extend the one already implemented by adding the possibility of temporarily locking objects which should not be modified by other users.

As mentioned in the introduction the methodology to run a cross-university test was developed, and therefore this is included as part of further work proposals. The following section will describe in detail the guidelines for this future experiment.

9.4.7 Preparation of the Collaborative Session between the two Universities

This section illustrates the general methodology that was developed to set up a more ambitious experiment than the in-house test described in Chapter 8. This test will involve architecture students from the University of Strathclyde in Glasgow, and from the Technische Universiteit Eindhoven (TU/e) in the Netherlands. This collaborative session has not yet been carried out due to the complexity of matching timetables and classes between the two institutions. However, the preparation work described in the following sections was included in this thesis because it offers a structured guideline for further experiments in the future.

9.4.7.1 The Methodology

The experiment would be divided into two sessions of 2 hours each, and would involve a minimum of 6 students in Glasgow and 6 students in Eindhoven (3 students at each session in both institutions).

The two sessions would be programmed as follows:

- 15 minutes to explain the exercise to the students and to distribute the material
- 1 hour and 45 minutes to fulfil the design task.

During each session 3 students in Glasgow would be working with 3 students in Eindhoven. One student at the University of Strathclyde and one at the TU/e would be matched and treated as a group. For each session there would therefore be 3 groups of 2 people.

Each group would work on the assigned task, to design an information kiosk by collaborating over the net. Different groups would use either *JCAD-VR TU/e Version* packages or emails with files as attachments. Specifically, as illustrated in Figure 9.2:

- Group 1 would be asked to accomplish the design task during a collaborative session using JCAD-VR TU/e Version.
- Group 2 would be using JCAD-VR TU/e Version which was not connected to the network plus emails, as their means of collaboration and for sending and receiving files with the upgraded version of their design.
- Group 3 would be asked to use traditional sketches and emails to allow the flow of information.

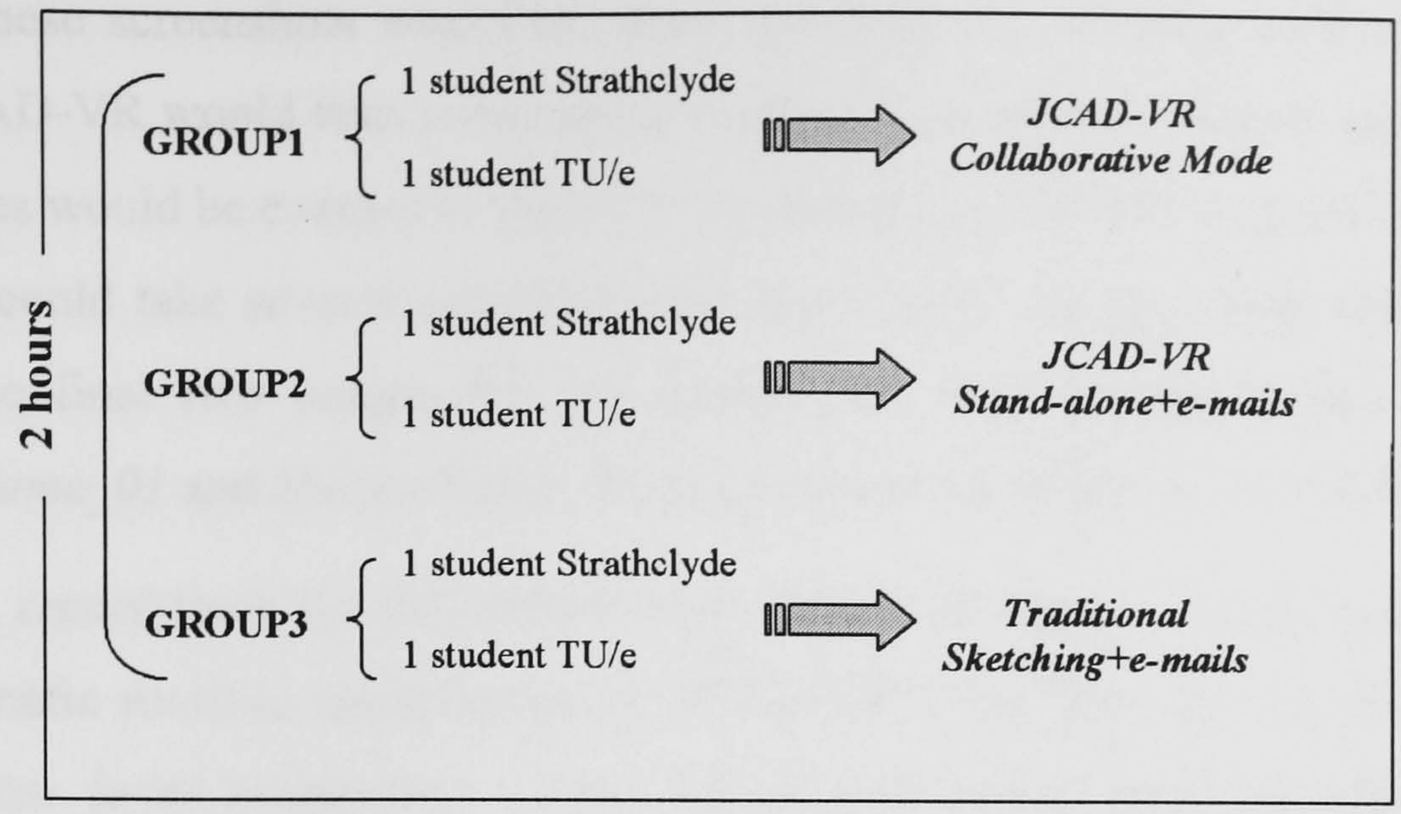


Figure 9.2: Schema for the cross-university experiment

Activity for Group 1

The couple would be asked to fulfil the design task using JCAD-VR TU/e Version running in collaborative mode. They could use JCAD-VR's tools to participate and thus they would test the capabilities of the software to let users collaborate in a synchronous manner. Both students would work concurrently on the design task as described in Appendix F.

The material to be produced at the end of the 2 hours would include 2 screenshots per student. The points of view chosen have to clearly visualize the final stage of their design. Possibly one of the two views (illustrated in Figure 9.3) should be chosen:

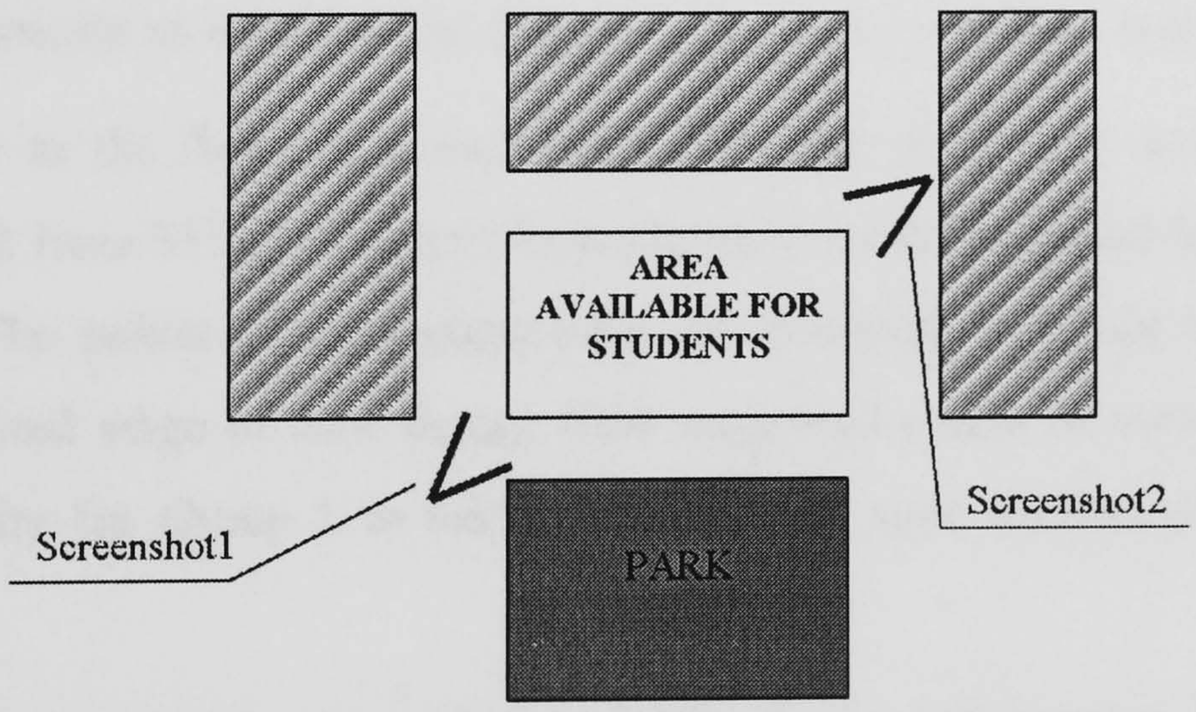


Figure 9.3: Two screenshots per student will show the final design achieved through the JCAD-VR collaborative session

These screenshots would be saved manually by clicking on the screenshot icon. JCAD-VR would then automatically assign a sequential name to each of them. These files would be eventually stored in the directory *JCAD-VR\capture*. Potentially students could take several snapshots and then choose the two most representative ones. The final two images (in jpg format) for each student would be named *StudentName_01* and *StudentName_02* and saved in the directory *JCAD-VR\capture*.

A record track for the collaborative session of Group 1 would be kept using the automatic routines implemented in JCAD-VR TU/e Version (See Section 8.2). The system saves screenshots of the VE in jcad format automatically every 10 minutes. The entire environment would be stored with a sequential number (i.e. *screenshot 1, screenshot 2, screenshot n... File 1, File 2, File n*) in the *JCAD-VR\automatic_save* folder. In addition the content of the chat board would be recorded. These routines would help keep a sequential track of the most relevant design achievements during the experiment and would highlight any problem during the collaborative session.

Activity for Group 2

The pair would be asked to fulfil the design task described, by using JCAD-VR TU/e Version as a stand-alone application. As a consequence of not being connected to the server, the system would not activate any collaborative tools.

Students would communicate through emails and exchange files saved in JCAD-VR format (jcad) as email attachments. Students of both universities would be free to communicate as much and as frequently as they wish (See Appendix F).

Similar to the first group two screenshots per student (2 screenshots from Strathclyde + 2 from TU/e = 4 screenshots per pair) would be asked for at the end of the 2 hours. The points of view chosen for the screenshots would have to clearly visualize the final stage of their design (See suggested points of view in Figure 9.3 and see Activity for Group 1 to learn about how to save screenshots manually in JCAD-VR).

Students would be also asked to keep all the information, which flowed within their group, by keeping all the emails received and the jcad files exchanged. This would keep track of the frequency of the exchange of information and would

record the evolution of the design. The emails received during the experiment would have to be saved in a separated folder whose path in the hard-drive would have to be agreed with a tutor before the beginning of the session. In addition, JCAD-VR TU/e Version (See Section 8.2) would automatically save screenshots in its internal folder (*JCAD-VR\automatic_save*) every 10 minutes, and every 5 minutes it would perform a back-up saving of jcad files in *JCAD-VR\automatic_save*.

Activity for Group 3

The pair would be asked to fulfil the design task using traditional free hand sketching. Students would exchange scanned files of sketches etc. as email attachments and use emails to communicate. Students from both universities would be free to exchange information and files over the net as frequently as they wish. They would also require a scanner and printer in order to produce the final design. The paper format for the experiment would be A4 size.

The material to be produced at the end of the 2 hours would be, 1 perspective drawing per student (1 from Strathclyde + 1 from TU/e = 2 perspectives per group) where the point of view chosen would have to clearly visualize the final stage of their design (Figure 9.3 shows the suggested points of view).

Students would be asked to keep all the evidence of the flow of information within the group by keeping all the received emails and the exchanged files. In this way it would be possible to keep track of the frequency of the exchange of information and to record the evolution of the design. Similar to Group 2, the emails received would have to be saved in a local folder. The location of this folder in the hard-drive would have to be agreed with the tutor before the beginning of the session.

Material to be Provided to the Students to fulfill the Design Task

Once launched, the system would be ready for the experiment and supplied with the appropriate background for Group 1 and Group 2 (See Section 8.2).

For Group 3, a tutor would provide the students with several plans of the site including:

- A general plan

- An elevation of the site, showing where to locate the kiosk
- 2 sections
- 2 perspectives.

Additional Introductory Session for those Students using Software

An introductory session of one hour would be necessary to introduce the students to the software package and to allow some practice before the experiment.

This session could be arranged separately by the students in each university.

Evaluation

This experiment would help in evaluating the collaborative capabilities that JCAD-VR offers to users and it would make a comparison of it possible, against other collaborative methods. This test would allow the monitoring of the information exchange, the working conditions and the design solutions through the sequences of snapshots (for those students working with JCAD-VR TU/e Version) and the exchange of files. The consistency of all the students' final products would give the opportunity to qualitatively evaluate the efficiency of JCAD-VR against the more traditional methods. In this experiment the snapshots recorded at a fixed time interval, and the data, such as the time recorded on the emails, would provide the means for reconstructing the students' design processes. Moreover, through each group's final perspectives it would be possible to evaluate if the communication between the two members of each group was precise enough to allow them accomplish a common design task.

In addition, the tutors would be provided with a questionnaire for each of the three groups involved in the experiment (See Appendix E). Through the students answers to these questionnaires it would be possible to obtain further feedback on the effectiveness of each of the collaborative methods and about the JCAD-VR system.

9.4.7.2 The Informal Testing

The cross-university test has not taken place. However a short and informal session to check for problems in the connection between the two institutions was carried out. The main concern was on the delay in the flow of information due to the

distance between the universities and on potential network problems arising from the presence of firewalls. However, this informal session was successful and proved the feasibility of a long distance experiment.

9.5 Conclusions

The JCAD-VR prototype promotes the change of usage of VR from being a mere presentation medium to being a more powerful and effective design tool. It proved the feasibility of using VR systems as the future interface for the next generation of computer aided design applications for architecture and it provided the user with multimedia tools and a network platform for enhancing communication and allowing collaboration among remote participants.

JCAD-VR is an original software package specifically developed for architects and even if not yet complete or optimised as a commercial application, it has allowed collaborative sessions and the effective exchange of design information.

This chapter summarised the contribution towards enhancement of the architectural design process brought by the JCAD-VR project, and it illustrated further developments and improvements for the prototype.

The development of JCAD-VR gave the authors the possibility to investigate and solve many theoretical and technical issues. In several occasions the authors experienced both excitement and frustration caused by the nature of the research and the fast progress of the technology involved. In particular the implementation of the Visual Core of the system took months of intense work in order to allow the use of the Virtual Environment Laboratory (VEL) (University of Strathclyde, 2002), and to achieve flexibility in the use of visual devices for the framework. Today the same issue would be solved in less time and more efficiently thanks to the newly released Java3D™ API 1.3, which supports multiple screen configurations by default through a much simpler mechanism.

This highlights the issue of working with cutting edge technology where advances in hardware and software can in some cases vanish efforts or make research results soon obsolete. However, to conclude with a personal comment the most

exciting moment of the entire project was the testing of the software. During the experiment, in fact, people other than the authors were involved in using JCAD-VR and their quick understanding of the system along with their enthusiasm has been a source of great excitement and encouragement for the authors.

References

3DConnexion (2002). Product Overview - SpaceMouse® Classic. Retrieved 13th October 2002 from the World Wide Web: http://www.logicad3d.com/products/Classic.htm

Achten, H., De Vries, B. and Jessurun, J. (2000). DDDoolz. A Virtual Reality Sketch Tool for Early Design. In Tan, B. Tan, M., Wong, Y. (Eds.), *CAADRIA 2000, Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia, Singapore, 18-19 May 2000* (pp. 451-460).

Alvarado, R. G., Marquez, J. C. P. and Vildosola G. V. (2001). Qualitative Contribution of a VR-System to Architectural Design: why we Failed?. In Gero, J. S., Chase, S. and Rosenman, M. (Eds.), *CAADRIA 2001, Proceedings of the Sixth Conference on Computer Aided Architectural Design Research in Asia, Sydney, Australia* (pp. 423-427).

Anderson, D., Frankel, J. L., Marks, J., Agarwala, A., Beardsley, P., Hodgins, J., Leigh, D., Ryall, K., Sullivan, E. and Yedidia, J. S. (2000). Tangible Interaction + Graphical Interpretation: A New Approach to 3D Modeling. In *Proceedings of the 27th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques, New Orleans, USA* (pp. 393-402). New York: ACM Press/Addison-Wesley Publishing Co. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/344779.344960

Arthur, K. W., Booth, K. S. and Ware, C. (1993). Evaluating 3D Task Performance for Fish Tank Virtual Worlds. In *ACM Transactions on Information Systems (TOIS)*, Volume 11, Issue 3 (July 1993), pp. 239-265. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/159161.155359

Arvo, J. and Novins, K. (2000). Fluid Sketches: Continuous Recognition and Morphing of Simple Hand-Drawn Shapes. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology, San Diego,*

2000, USA (pp. 73-80). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/354401.354413

Ataman, O. (2000). Measuring the Impact of Media on Architectural Design. In Ripper, J. Pessoa, A. and Rodriguez, D. (Eds.), *SIGraDi 2000 - Costruindo o Espaco Digital* (Constructing the Digital Space), *Anais do IV Congresso Iberoamericano de Gràfica Digital, Rio de Janeiro, Brazil* (pp. 93-97).

Aukstakalnis, S. and Blatner, D. (1992). *Silicon Mirage: The Art and Science of Virtual Reality.* Berkeley, CA: Peach Pit Press.

Balakrishnan, G. and Kurtenbach, G. (1999b). Exploring Bimanual Camera Control and Object Manipulation in 3D Graphics Interfaces. In *Proceedings of the CHI 99 Conference on Human Factors In Computing Systems: the CHI is the Limit, Pittsburgh, Pennsylvania, USA* (pp. 56-62). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/302979.302991

Balakrishnan, R., Fitzmaurice, G., Kurtenbach, G. and Buxton, W. (1999a). Digital Tape Drawing. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology, Minneapolis, Minnesota, USA* (pp. 161-169). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/320719.322598

Barco (2002). Baron. Retrieved 02nd October 2002 from the World Wide Web: http://www.barco.com/projection_systems/virtual_and_augmented_reality/content/products/product.asp?Element=312

Barrilleaux, J. (2001). 3D User Interfaces with Java 3D. A Guide to Computer-Human Interaction in Three Dimension. Greenwich: Manning.

Bolas, M. (1994). Designing Virtual Environments. In Loeffler, C. E. and Anderson, T. (Eds.), *The Virtual Reality Casebook* (pp. 49-55). New York: Van Nostrand Reinhold.

Bootstrap Alliance (2002). Biography of Douglas Carl Engelbart. Retrieved 29th August 2002 from the World Wide Web: http://www.bootstrap.org/engelbart/

Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I. (2001). An Introduction to 3-D User Interface Design. In *PRESENCE: Teleoperators and Virtual Environments*, 10(1), February 2001, pp. 96-108. Cambridge (USA): The MIT Press.

Burdea, G. and Coiffet, P. (1994). *Virtual Reality Technology*. New York: John Wiley & Sons, Inc.

Butterworth, J., Davidson, A., Hench, S. and Olano. T. M. (1992). 3DM: A Three Dimensional Modeler Using a Head-Mounted Display. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics, Cambridge, Massachusetts, USA* (pp. 135-138). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/147156.147182

Camarata, K., Do, E. Y., Johnson, B. R. and Gross, M. D. (2002). Navigational Blocks – Navigating Information Space with Tangible Media. In *Proceedings of the 7th International Conference on Intelligent User Interfaces, San Francisco, California, USA* (pp. 31-38). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/502716.502725

Chan, C., Hill, L. and Cruz-Neira, C. (1999a). It is Possible to Design in Full Scale? A CAD Tool in a Synthetic Environment. In Jingwen, G. and Zhaoji, W. (Eds.), *CAADRIA '99, Proceedings of the Fourth Conference on Computer Aided Architectural Design Research in Asia* Shanghai, China (pp. 43-52).

Chan, C., Maves, J. and Cruz-Neira, C. (1999b). An Electronic Library for Teaching Architectural History. In Jingwen, G. and Zhaoji, W. (Eds.), *CAADRIA '99, Proceedings of the Fourth Conference on Computer Aided Architectural Design Research in Asia,* Shanghai, China (pp. 335-344).

Chen, M., Mountford, S. J. and Sellen, A. (1988). A Study in Interactive 3-D Rotation using 2-D Control Devices. In *Proceedings of the 15th SIGGRAPH annual International Conference on Computer Graphics and Interactive Techniques, 1988* (pp. 121-129). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/54852.378497

Clark, J. H. (1976). Designing Surfaces in 3-D. In *Communications of the ACM*, Volume 19, Issue 8 (August 1976), pp. 454-460. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/360303.360329

Cockburn, A. and McKenzie, B. (2002). Evaluating the Effectiveness of Spatial Memory in 2D and 3D Physical and Virtual Environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing our World, Changing Ourselves, 2002, Minneapolis, Minnesota, USA* (pp. 203-210). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/503376.503413

Cohen, J. M., Huges, J. F. and Zeleznik, R. C. (2000). Harold: a World Made of Drawings. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering, Annecy, France* (pp. 83-90). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/340916.340927

Cohen, J. M., Markosian, L., Zeleznik, R. C., Hughes, J., F. and Barzel, R. (1999). An Interface for Sketching 3D Curves. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics, Atlanta, Georgia, USA* (pp. 17-21). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/300523.300655

Cohen, P. R., Johnston, M., McGee, D., Oviatt, S. L., Clow, J. and Smith, I (1998). The Efficiency of Multimodal Interaction: a Case Study. In Mannel, R. H. and Robert-Ribes, J. (Eds.), *Proceedings of the International Conference on Spoken Language (ICSLP'98),Sydney, Australia* (pp. 249-252). Sydney: Australian Speech Science and Technology Association, Incorporated (ASSTA).

Conner, D. B., Snibbe, S. S., Herndon, K. P., Robbins, D. C., Zeleznik, R. C. and van Dam, A. (1992). Three-dimensional widgets. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics, 1992, Cambridge, Massachusetts, USA* (pp. 183-188). New York ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/147156.147199

Cruz-Neira, C., Sandin, D. J. and DeFanti, T. (1993). Surround-Screen Projection-Based Virtual Reality: the Design and Implementation of the CAVE. In *Proceedings of the 20th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques, Anaheim, CA, USA* (pp. 135-142). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/166117.166134

Dagit, C. E. (1993). Establishing Virtual Design Environments in Architecture Practice. In U. Flemming and S. Van Wyk (Eds.), *CAAD Futures '93 Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures* 1993 (pp.513-521). North-Holland: Elsevier Science Publishers B.V.

Daru, R. (1991). Sketch as Sketch Can – Design sketching with imperfect aids and sketchpads of the future. In Pittioni, G. (Ed.), *Experiences with CAAD Education and Practice, eCAADe Conference Proceedings,* Munich, Germany (pp. 162-172).

De Amicis, R., Bruno, F., Stork, A. and Luchi, M. L. (2002). The Eraser Pen: a New Interaction Paradigm for Curve Sketching in 3D. In Marjanovic (Ed.), *Proceedings of the Design 2002 7th International Design Conference, Dubrovnik, Croatia* (Volume 1, pp. 465-470). Glasgow: The Design Society.

De Vries, B. (2000). Sketching in 3D. In Donath, D. (Ed.), *Promise and Reality: State of the Art versus State of Practice in Computing for the Design and Planning Process, 18th eCAADe Conference Proceedings, Weimar, Germany* (pp. 277-280). Weimar: VDG Verlag.

Deering, M. (1992). High Resolution Virtual Reality. In *Proceedings of the 19th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques* (pp. 195-202). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/133994.134039

Deering, M. (1995). HoloSketch: A Virtual Reality Sketching / Animation Tool. In *ACM Transactions on Computer-Human Interaction (TOCHI)*, Volume 2, Issue 3 (September 1995), pp. 220-238. New York: ACM Press. Retrieved 14th

October 2002 from the World Wide Web: http://doi.acm.org/10.1145/210079.210087

Deering, M. (1996). The HoloSketch VR Sketching System. In *Communications of the ACM*, Volume 39, Issue 5 (May 1996), pp. 54-61. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/229459.229466

Deisinger, J, Blach, R., Wesche, G., Breining, R. and Simon, A. (2000). Towards immersive modeling – challenges and recommendations: A workshop analysis of the needs of designers. In Mulder, J. and van Liere, R. (Eds.), *Eurographics Workshop on Virtual Environments 2000, Amsterdam, The Netherlands* (pp. 145-156). Wien: Springer.

Do, E. Y. (2000). Sketch that Scene for Me: Creating Virtual Worlds by Freehand Drawing. In Donath, D. (Ed.), *Promise and Reality: State of the Art versus State of Practice in Computing for the Design and Planning Process, 18th eCAADe Conference Proceedings, Weimar, Germany* (pp. 265-268). Weimar: VDG Verlag.

Do, E. Y. (2001). Sketching Interfaces for Conceptual Design and Analysis in Architecture. In *CHI Workshop, Tools, Conceptual Frameworks and Empirical Studies for Early Stages of Design, April 2001, Seattle, USA*. Retrieved 7th June 2002 from the World Wide Web: http://depts.washington.edu/dmgwksp/PP/do.pdf

Donath D., Regenbrecht, H and Springer, J. (1998). VoxDesign User's Guide. Retrieved 1st July 2002 from the World Wide Web: http://www.uni-weimar.de/architektur/InfAR/forschung/voxDesign/

Dorta, T., LaLande, P. (1998). The Impact of Virtual Reality on the Design Process. In S. van Wyk and T. Seebohm (Eds.) Digital Design Studios: Do Computers Make a Difference in Design Studio? In *ACADIA Conference Proceedings 1998, Quebec City, Quebec, Canada, 22-25 October*, (pp. 138-160).

Durand, F. (2002). An Invitation to Discuss Computer Depiction. In *Proceedings of the Second International Symposium on Non-photorealistic Animation and*

Rendering, 2002, Annecy, France (pp. 111-124). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/508530.508550

EAEA - European Architectural Endoscopy Association (1995). Statute of the Association. Retrieved 5th June 2002 from the World Wide Web: http://info.tuwien.ac.at/eaea/e-statutes.html

Eckel, B. (2000). *Thinking in Java, 2nd Edition.* Retrieved 07th August 2002 from the World Wide Web: http://www.mindview.net/Books/TIJ/

Elmer-Dewitt, P. (1993). Cyberpunk. *Time Magazine*, February 8, pp. 58-65.

Fakespace Systems (2002). ImmersaDesk®R2. Retrieved 2nd October 2002 from the World Wide Web: http://www.fakespacesystems.com/workdesk.shtml

Fiorentino, M., De Amicis, R., Stork, A. and Monno, G. (2002). Surface Design in Virtual Reality as Industrial Application. In Marjanovic (Ed.), *Proceedings of the Design 2002 7th International Design Conference, Dubrovnik, Croatia* (Volume 1, pp. 477-482). Glasgow: The Design Society.

Fisher, S. S., McGreevy, M., Humphries J. and Robinett W. (1987). Virtual Environment Display System. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics, 1987, Chapel Hill, North Carolina, USA* (pp. 77-87). New York: ACM Press.

Fishkin, K. P., Gujar, A., Harrison, B. L., Moran, T. P. and Want, R. (2000). Embodied User Interfaces for Really Direct Manipulation. In *Communications of the ACM*, Volume 43, Issue 9 (September 2000), pp. 74-80. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/348941.348998

Forsberg, A., Herndon, K. and Zeleznik, R. (1996). Aperture Based Selection for Immersive Virtual Environments. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology, 1996, Seattle, Washington, USA* (pp. 95-96). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/237091.237105

Free Software Foundation (1999). The GNU Lesser General Public License. Retrieved 30th September 2002 from the Word Wide Web: http://www.gnu.org/copyleft/lesser.html

Fukuda, T., Nagahama, R. Oh, S., Kaga, A. and Sasada, T. (2000). Collaboration Support System for Nightscape Design Based on VR Technology. In Tan, B. Tan, M., Wong, Y. (Eds.), *CAADRIA 2000, Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia, Singapore, 18-19 May 2000* (pp. 501-510).

Gatermann, H. (2000). From Vrml to Augmented Reality via Panorama-Integration and EAI-Java. In Ripper, J. Pessoa, A. and Rodriguez, D. (Eds.), *SIGraDi 2000 - Costruindo o Espaco Digital* (Constructing the Digital Space), *Anais do IV Congresso Iberoamericano de Gràfica Digital, Rio de Janeiro, Brazil* (pp. 254-256).

Gatermann, H. (2001). First Step to Augmented Reality: Combining Vrml and Pano-Photos. In Gero, J. S., Chase, S. and Rosenman, M. (Eds.), *CAADRIA 2001, Proceedings of the Sixth Conference on Computer Aided Architectural Design Research in Asia, Sydney, Australia* (pp. 423-427).

Gavin, L. (1999). Architecture of the Virtual Place. In Brown, A., Knight, M. and Berridge, P. (Eds.), *Architectural Computing from Turing to 2000, 17th eCAADe Conference Proceedings, Liverpool, Great Britain* (pp. 418-423).

Gentner, D. R., Grudin, J. (1990). Why Good Engineers (Sometimes) Create Bad Interfaces. In *Conference Proceedings on Empowering People: Human Factors in Computing System: Special Issue of the SIGCHI Bulletin, 1990, Seattle, Washington, USA* (pp. 277-282). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/97243.97287

Gibson, J. J. (1950). Visual World. Boston: Houghton Mifflin Company.

Gibson, W. (1984). Neuromancer. New York: Ace Books.

Green, M. and Halliday, S. (1996). A Geometric Modeling and Animation System for Virtual Reality. In *Communications of the ACM*, Volume 39, Issue 5 (May

1996), pp. 46-53. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/229459.229465

Gröhn, M. G., Mantere, M., Savioja, L., Takala, T. (2001). 3D Visualization of Building Services in Virtual Environment. In Penttilä, H. (2001), *eCAADe 2001 Conference Proceedings, Helsinki University of Technology, Helsinki, Finland* (pp. 523-528). Otaniemi, Finland: Otamedia Oy.

Gross, M. D. and Do, E. Y (1996a). Ambiguous Intentions: a Paper-Like Interface for Creative Design. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology, Seattle, Washington, USA* (pp. 183-192). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/237091.237119

Gross, M. D. and Do, E. Y (1996b). Demonstrating the Electronic Cocktail Napkin: a Paper-Like Interface for Early Design. In *Proceedings of the CHI '96 Conference Companion on Human Factors in Computing Systems: Common Ground, Vancouver, British Columbia, Canada* (pp. 5-6). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/257089.257092

Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A. and Buxton, B. (2002). Creating Principal 3D Curves with Digital Tape Drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Changing our World, Changing Ourselves, Minneapolis, Minnesota, USA* (pp. 121-128). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/503376.503398

Grudin, J. (1990a). Interface. In *Proceedings of the Conference on Computer-Supported Cooperative Work, 1990, Los Angeles, California, USA* (pp. 269-278). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/99332.99360

Grudin, J. (1990b). The Computer Reaches Out: the Historical Continuity of Interface Design. In *Conference Proceedings on Empowering People: Human factors in computing system: special issue of the SIGCHI Bulletin, 1990, Seattle, Washington, USA* (pp. 261 –268). New York: ACM Press. Retrieved

14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/97243.97284

Haik, E., Barker, T., Sapsford, J. and Trainis, S. (2002). Investigation into Effective Navigation in Desktop Virtual Interfaces. In *Proceeding of the 7th International Conference on 3D Web Technology, 2002, Tempe, Arizona, USA* (pp. 59-66). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/504502.504513

Hamilton, J. (1992). Virtual Reality. How Technology can amplify the human mind. *Business Week*, October 5, pp. 96-105.

Hanson, A. J. and Wernert, E. A. (1997). Constrained 3D navigation with 2D controllers. In *Proceedings of 8th Visualization '97 Conference, 1997, Phoenix, USA* (pp. 175-182). USA: IEEE Computer Science Press.

Harrison, B. L., Fishkin, K. P., Gujar, A., Mochon, C. and Watt, R. (1998). Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces. In *Conference Proceedings on Human Factors in Computing Systems, 1998, Los Angeles, California, USA* (pp. 17-24). New York: ACM Press/Addison-Wesley Publishing Co. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/274644.274647

Hart, R. A. and Moore, G.T. (1973). The Development of Spatial Cognition: A Review. In Downs, R. M. and Stea, D. (Eds.) *Image and Environment: Cognitive Mapping and Spatial Behavior*. Chicago: Aldine Publishing Company.

Heilig, M. (1992). Enter the Experiential Revolution: a VR Pioneer Looks Back to the Future. In Jacobson, L. (Ed.). *Cyberarts, Exploring Art and Technology* (pp. 292-305). Backbeat Books.

Hinckley, K., Pausch, R. and Goble, C. G. (1994). A Survey of Design Issues in Spatial Input. In *Proceedings of the Annual ACM Symposium on User Interface Software and Technology, 1994, Marina del Rey, California, USA* (pp. 213-222). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/192426.192501

Hinckley, K., Pausch, R., Proffitt, D. and Kassell, N. (1998). Two-Handed Virtual Manipulation. In *ACM Transactions on Computer-Human Interaction, Vol. 5, No. 3, September 1998,* pp. 260-302. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/292834.292849

Hinckley, K., Tullio, J., Pausch, R. and Proffitt, D. and Kassell, N. (1997). Usability Analysis of 3D Rotation Techniques. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology, 1997, Banff, Alberta, Canada* (pp. 1-10). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/263407.263408

Hubona, G. S., Wheeler, P. N., Shirah, G. W. and Brandt, M. (1999). The Relative Contributions of Stereo, Lighting, and Background Scenes in Promoting 3D Depth Visualization. In *ACM Transactions on Computer-Human Interaction (TOCHI),* Volume 6, Issue 3 (September 1999), pp. 214-242. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/329693.329695

Hughes, M., Hughes, C., Shoffner, M. and Winslow, M. (1997). *Java Network Programming*. Greenwich, CT, USA: Manning Publications Co.

IBM (2002). Speech for Java. Retrieved 13th October 2002 from the Word Wide Web: http://www.alphaworks.ibm.com/tech/speech

Igarashi, T., Edwards, W. K., LaMarca, A. and Mynatt, E. D. (2000). An Architecture for Pen-Based Interaction on Electronic Whiteboards. In *Proceedings of the Working Conference on Advanced Visual Interfaces, Palermo, Italy* (pp. 68-75). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/345513.345256

Igarashi, T., Kadobayashi, R., Mase, R. and Tanaka, H. (1998). Path drawing for 3D walkthrough. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology, 1998, San Francisco, California, United States* (pp. 173-174). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/288392.288599

Igarashi, T., Matsuoka, S. and Tanaka, H. (1999). Teddy: A Sketching Interface for 3D Freeform Design. In *Proceedings of the 26th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques* (pp. 409-416). New York: ACM Press/Addison-Wesley Publishing Co. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/311535.311602

Immersion Corporation (2002). Hardware products. Retrieved 13th October 2002 from the World Wide Web: http://www.immersion.com/products/3d/interaction/hardware.shtml

Johnston, M., Cohen, P. R., McGee, D., Oviatt, S. L., Pittman, J. A. and Smith, I. (1997). Unification-Based Multimodal Integration. In *Proceedigns of the 35th Annual Meeting of the Association for Computational Linguistics ACL*, (pp. 281-288). New York.

Jozen, T., Wang, L. and Sasada, T. (1999). Sketch VRML – 3D Modeling of Conception. In Brown, A., Knight, M. and Berridge, P. (Eds.), *Architectural Computing from Turing to 2000, 17th eCAADe Conference Proceedings, Liverpool, Great Britain* (pp. 557-563).

Jung, T., Gross, M. D. and Do, E. Y. (2002). Annotating and Sketching on 3D Web Models. In *Proceedings of the 7th International Conference on Intelligent User Interfaces, San Francisco, California, USA* (pp. 95-102). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/502716.502733

Kameyama, K. (1997). Virtual Clay Modeling System. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 1997, Lausanne, Switzerland* (pp. 197-200). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/261135.261171

Kaplan, D. (1993). An Interactive Life. *Newsweek*, May 31, pp. 42-45.

Keefe, D. F., Feliz, D. A., Moscovich, T., Laidlaw, D. H. and LaViola, J. J. (2001). CavePainting: A Fully Immersive 3D Artistic Medium and Interactive Experience. In *Proceedings of the 2001 Symposium on Interactive 3D*

graphics, Research Triangle Park, NC, USA (pp. 85-93). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/370000.364370

Knight, M. W. and Brown, A. G. P. (1999). Working in Virtual Environments through appropriate Physical Interfaces. In Brown, A., Knight, M. and Berridge, P. (Eds.), *Architectural Computing from Turing to 2000, 17th eCAADe Conference Proceedings, Liverpool, Great Britain* (pp. 431-436).

Knight, M. W. and Brown, A. G. P. (2000). Interfaces for Virtual Environments; a Review of Recent Developments and Potential Ways Forward. In Donath, D. (Ed.), *Promise and Reality: State of the Art versus State of Practice in Computing for the Design and Planning Process, 18th eCAADe Conference proceedings, Weimar, Germany* (pp. 215-219). Weimar: VDG Verlag.

Koller, D. R., Mine, M. R. and Hudson, S. E. (1996). Head-Tracked Orbital Viewing: an Interaction Technique for Immersive Virtual Environments. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology, 1996, Seattle, Washington, USA* (pp. 81-82). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/237091.237103

Krueger, M. W. (1994). Foreword to the book. In Burdea, G. and Coiffet, P. *Virtual Reality Technology*. New York: John Wiley & Sons, Inc.

Kurmann, D. (1995). Sculptor - A Tool for Intuitive Architectural Design. In Tan, M. and Teh, R. (Eds.), *The Global Design Studio, Proceedings of the Sixth International Conference on Computer-Aided Architectural Design Futures*, Singapore (pp. 323-330).

Kurmann, D. (1998). Sculptor - How to Design Space? In Sasada, T., Yamaguchi, S., Morozumi, M., Kaga, A. and Homma, R. (Eds.), *CAADRIA '98 Proceedings of The Third Conference on Computer Aided Architectural Design Research in Asia, Osaka, Japan* (pp. 317-326).

Kurmann, D. (1999). Sculptor – a virtual design tool. Retrieved 7th June 2002 from the World Wide Web: http://caad.arch.ethz.ch/~kurmann/sculptor/

Kurmann, D., Elte, N. and Engeli, M. (1997). Real-time Modeling with Architectural Space. In Junge, R. (Ed.), *CAAD Futures 1997, Conference Proceedings, Dordrecht, The Netherlands* (pp. 809-819). Dordrecht: Kluwer Academic.

Kurtenbach, G., Fitzmaurice, G., Baudel, T. and Buxton, W. (1997). The Design of a GUI Paradigm based on Tablets, Two-Hands, and Transparency. In *Conference Proceedings on Human Factors in Computing Systems, Atlanta, Georgia, USA* (pp. 35-42). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/258549.258574

Lawson, B. (1990). *How Designers Think*. London, Boston: Butterworth Architecture.

Liu, S. and Huang, Z. (2000). Interactive 3D Modeling Using Only One Image. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 2000, Seoul, Korea* (pp. 49-54). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/502390.502400

Loeffler, C. E. and Anderson, T. (Eds.) (1994). Introduction to the book *The Virtual Reality Casebook* (pp. xiii-xxv). New York: Van Nostrand Reinhold.

Mackinlay, J. D., Card, S. K. and Robertson G. G. (1990). Rapid Controlled Movement Through a Virtual 3D Workspace. In *Proceedings of the 17th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques*, 1990, Dallas, USA (pp. 171-176). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/97879.97898

Mankoff, J., Hudson, S. E. and Abowd, G. D. (2000). Interaction Techniques for Ambiguity Resolution in Recognition-Based Interfaces. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology, 2000, San Diego, California, USA* (pp. 11-20). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/354401.354407

Markosian, L., Cohen, J. M., Crulli, T. and Hughes, J. (1999). Skin: A Constructive Approach to Modeling Free-form Shapes. In *Proceedings of the 26th SIGGRAPH annual International Conference on Computer Graphics and Interactive Techniques* (pp. 393-400). New York: ACM Press/Addison-Wesley Publishing Co. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/311535.311595

Markosian, L., Kowalski, M. A., Goldstein, D., Trychin, S. J., Hughes, J. F. and Bourdev, L. D. (1997). Real-Time Nonphotorealistic Rendering. In *Proceedings of the 24th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques* (pp. 415-420). New York: ACM Press/Addison-Wesley Publishing Co. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/258734.258894

Mase, J. (2000). Moderato: 3D Sketch CAD with Quick Positioned Working Plane and Texture Modelling. In Donath, D. (Ed.), *Promise and Reality: State of the Art versus State of Practice in Computing for the Design and Planning Process, 18th eCAADe Conference proceedings, Weimar, Germany* (pp. 269-272). Weimar: VDG Verlag.

Matsumiya, M., Takemura, H. and Yokoya, N. (2000). An Immersive Modeling System for 3D Free-Form Design using Implicit Surfaces. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 2000, Seoul, Korea* (pp. 67-74). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/502390.502404

Maybury, M. (1998). Intelligent User Interfaces: an Introduction. In *Proceedings of the 4th international conference on Intelligent user interfaces, Los Angeles, California, USA* (pp. 3-4). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/291080.291081

McDonnell, K. T., Qin, H. and Wlodarczyk, R. A. (2001). Virtual Clay: a Real-Time Sculpting System with Haptic Toolkits. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (pp. 179-190). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/370000.364395

Microsoft Corp. (2002). Microsoft DirectX®. Multimedia Technology for Windows-Based Gaming and Entertainment. Retrieved 25th September 2002 from the Word Wide Web: http://www.microsoft.com/windows/directx/

Moloney, J. (1999). Bike-R: Virtual Reality for the Financially Challenged. In Brown, A., Knight, M. and Berridge, P. (Eds.), *Architectural Computing from Turing to 2000, 17th eCAADe Conference Proceedings, Liverpool, Great Britain* (pp. 410-413).

Morgan, J. (1994). Real Artifice: Myron Krueger's Beautiful Interface – an interview to Myron Krueger. In Loeffler, C. E. and Anderson, T. (Eds.), *The Virtual Reality Casebook* (p. 171-177). New York: Van Nostrand Reinhold.

Mountford, S. J. (1990). Designers: meet your users. In Chew, J. C., Whiteside, J. (Eds.). *Conference proceedings on Empowering people: Human factors in computing system: special issue of the SIGCHI Bulletin, 1990, Seattle, Washington, USA* (pp. 439 –442). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/97243.97321

Myers, B., Ioannidis, Y., Hollan, J., Cruz I., Bryson, S., Butlerman, D., Catarci, T., Citrin, W., Glinert, E. and Grudin, J. (1996). Strategic Directions in Human-Computer Interaction. In *ACM Computing Surveys (CSUR), Special ACM 50th-Anniversary issue: Strategic Directions in Computing Research*, Volume 28, Issue 4 (December 1996), pp. 794-809. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/242223.246855

Mynatt, E. D., Igarashi, T., Edwards, W. K. and LaMarca, A. (1999). Flatland: new Dimensions in Office Whiteboards. In *Proceeding of the CHI 99 Conference on Human Factors in Computing Systems: the CHI is the Limit, Pittsburgh, Pennsylvania, USA* (pp. 346-353). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/302979.303108

Negroponte, N. (1993). Virtual Reality: Oxymoron or Pleonasm? *Wired, December (1) 6.* Retrieved 31st may 2002 from the World Wide Web: http://www.wired.com/wired/archive/1.06/negroponte.html?pg=1

Negroponte, N. (1996). Being digital. London: Coronet Books.

Northrup, J. D. and Markosian L., (2000). Artistic Silhouettes: A Hybrid Approach. In *Proceedings of the First International Symposium on Non-Photorealistic Animation and Rendering, Annecy, France* (pp. 31-37). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/340916.340920

Object Management Group (OMG®) (2002). UML™ Resource Page. Retrieved 25th September 2002 from the Word Wide Web: http://www.omg.org/uml/

OpenGL® (2002). The Industry Foundation for High Performance Graphics. Retrieved 21th September 2002 from the Word Wide Web: http://www.opengl.org/

Oviatt, S., Cohen, P. (2000). Multimodal Interfaces that Process what Comes Naturally. In *Communications of the ACM*, 43 (3), March 2000, pp. 45-54. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/330534.330538

ParallelGraphics (2002). Cortona VRML client. Retrieved 02nd October 2002 from the Word Wide Web: http://www.parallelgraphics.com/products/cortona/

Park, H. (1996). Digital and Manual Media in Design. In af Klercker, J., Ekholm, A. and Fridqvist, S. (1996) *Education for Practice, 14th eCAADe Conference Proceedings, Lund, Sweden* (pp. 325-334).

Patten, J. and Ishii, H. (2000). A comparison of spatial organization strategies in graphical and tangible user interfaces. In *Proceedings of DARE (Designing Augmented Reality Environments) 2000 on Designing augmented reality environments, 2000, Elsinore, Denmark* (pp. 41-50). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/354666.354671

Pausch, R., Proffitt, D. and Williams, G. (1997). Quantifying Immersion in Virtual Reality. In *Proceedings of the 24th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques* (pp. 13-18). New York: ACM Press/Addison-Wesley Publishing Co. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/258734.258744

Pimentel K. and Teixeira, K. (1995). Virtual Reality: Through the New Looking Glass, 2nd Edition. New York: Intel/McGraw-Hill.

Polhemus (2002). Fastrak. Retrieved 14th October 2002 from the Word Wide Web: http://www.polhemus.com/ftrakds.htm

Poupyrev, I., Billinghurst, M., Weghorst, S. and Ichikawa, T. (1996). The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology, 1996, Seattle, Washington, USA* (pp. 79-80). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/237091.237102

Pratini, E. (2001). New Approaches to 3D Gestural Modeling – the 3D SketchMaker Project. In Penttilä, H. (2000), *eCAADe 2001 conference proceedings, Helsinki University of Technology, Helsinki, Finland* (pp. 466-471). Otaniemi, Finland: Otamedia Oy.

Raskar, R. and Cohen, M. (1999). Image Precision Silhouette Edges. In *Proceedings of the 1999 Symposium on Interactive 3D graphics, Atlanta, Georgia, USA* (pp. 135-140). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/300523.300539

Reeves, B. and Nass, C. (2000). Perceptual User Interfaces: Perceptual Bandwidth. In *Communications of the ACM*, Volume 43, Issue 3 (March 2000), pp. 65-70. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/330534.330542

Regenbrecht, H., Kruijff, E., Seichter, H. and Beetz, J. (2000). VRAM a Virtual Reality Aided Modeller. In Donath, D. (Ed.), *Promise and Reality: State of the*

Art versus State of Practice in Computing for the Design and Planning Process, 18th eCAADe Conference proceedings, Weimar, Germany (pp. 235-237). Weimar: VDG Verlag.

Rheingold, H. (1994). Ethical Questions Posed by Virtual Reality Technology. In Loeffler, C. E. and Anderson, T. (Eds.), *The Virtual Reality Casebook* (p. 214-217). New York: Van Nostrand Reinhold.

Robbe-Reiter, S., Carbonell, N. and Dauchy, P. (2000). Expression Constraints in Multimodal Human-Computer Interaction. In *Proceedings of the 5th international conference on Intelligent user interfaces, 2000, New Orleans, Louisiana, USA* (pp. 225-228). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/325737.325852

Roberts, A. (1999). Virtual Site Planning. In Brown, A., Knight, M. and Berridge, P. (Eds.), *Architectural Computing from Turing to 2000, 17th eCAADe Conference Proceedings, Liverpool, Great Britain* (pp. 442-448).

Sachs, E., Roberts, A. and Stoops, D. (1991). 3-Draw: A Tool for Designing 3D Shapes. In *IEEE Computer Graphics and Applications*, November 1991, Vol. 11, N. 6, pp. 18-24.

Schmalstieg, D., Encarnação, L. M. and Szálavári, Z. (1999). Using Transparent Props For Interaction with The Virtual Table. In *Proceedings of the 1999 symposium on Interactive 3D graphics, 1999, Atlanta, Georgia, USA* (pp. 147-153). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/300523.300542

Sener, B., Vergeest, J. S. M. and Akar, E. (2002). New Generation Computer-Aided Design Tools: Two Related Research Projects Investigating the Future Expectations of Designers. In Marjanovic (Ed.), *Proceedings of the Design 2002 7th International Design Conference, Dubrovnik, Croatia* (Volume 1, pp. 539-544). Glasgow: The Design Society.

SensAble Technologies (2002). Products – Phantom. Retrieved 2nd October 2002 from the World Wide Web: http://www.sensable.com/haptics/products/phantom.html

Sherman B. and Judkins P. (1992). Glimpses of Heaven, Visions of Hell. Virtual Reality and its Implications. London: Hodder & Stoughton.

Shih, N. and Yang, C. (1999). A VR-Based Preference Study on the Selection of Egress Route in Evacuation. In Jingwen, G. and Zhaoji, W. (Eds.), *CAADRIA '99, Proceedings of the Fourth Conference on Computer Aided Architectural Design Research in Asia, Shanghai, China* (pp. 63-70).

Shoemake, K. (1992). ARCBALL: A User Interface for Specifying Three-Dimensional Orientation using a Mouse. In *Proceedings of Graphics Interface '92 Conference, Vancouver, Canada* (pp. 151-156). San Francisco: Morgan Kaufman Publishers.

Shukur, F. (2000). *Lost Information in Design*. Lund, Sweden: KF-Sigma.

Singh, G., Feiner, S. K. and Thalmann, D. (1996). Virtual Reality: Software and Technology. *Communications of the ACM*, Volume 39, Issue 5 (May 1996), pp. 35-36. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/229459.229463

Smardon, R. C., Palmer, J. F., Pelleman, J. P. (1986). *Foundations for Visual Project Analysis*. New York: John Wiley & Sons.

Smart Technologies, Inc. (2002). SMART Board interactive whiteboard. Retrieved 14th October 2002 from the Word Wide Web: http://www.smarttech.com/products/smartboard/index.asp

Song, C. G., Kwak, N. J. and Jeong, D. H. (2000). Developing an Efficient Technique of Selection and Manipulation in Immersive V.E. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 2000, Seoul, Korea* (pp. 142-146). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/502390.502417

Sowizral, H., Rushforth, and Deering M. (2000). The Java 3D™ API Specification, Second Edition. Boston: Addison Wesley.

Sowizral, H. and Deering M. (1999). The Java 3D API and Virtual Reality. In *IEEE Computer Graphics and Applications*, May/June 1999, Vol. 19, N. 3, pp. 12-15.

Stampe, D. and Roehl, B. (1994). The Rend386 Project: History and Current Status. In Loeffler, C. E. and Anderson, T. (Eds.), *The Virtual Reality Casebook* (pp. 56-59). New York: Van Nostrand Reinhold.

Stellingwerff, M. (1999). SketchBox. In Brown, A., Knight, M. and Berridge, P. (Eds.), *Architectural Computing from Turing to 2000, 17th eCAADe Conference Proceedings, Liverpool, Great Britain* (pp. 491-497).

Stellingwerff, M. and Breen, J. (1995). Applications of Optical and Digital Endoscopy. In Martens, B. (Ed.), *Proceedings of the 2nd European Architectural Endoscopy Association Conference in Wien*, Austria, 1995. ISIS Publications: Wien.

Stenger, N. (1994). Angels: a Virtual Movie. In Loeffler, C. E. and Anderson, T. (Eds.), *The Virtual Reality Casebook* (pp. 3-6). New York: Van Nostrand Reinhold.

Sukaviriya, P. and Foley, J. D. (1990). Coupling a UI Framework with Automatic Generation of Context-Sensitive Animated Help. In *Proceedings of The Third Annual ACM SIGGRAPH Symposium on User Interface Software and Technology, 1990, Snowbird, Utah, USA* (pp. 152-166). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/97924.97942

Sun Microsystems, Inc. (1998). The Java 3D™ API: for Developers and End-Users. Retrieved 06th September 2002 from the Word Wide Web: http://java.sun.com/products/java-media/3D/collateral/presentation/

Sun Microsystems, Inc. (1999). The Java 3D API Specification. Introduction to Java 3D. Retrieved 17th September 2002 from the Word Wide Web: http://java.sun.com/products/java-media/3D/forDevelopers/j3dguide/Intro.doc.html

Sun Microsystems, Inc. (2000). Java 3D™ API Datasheet. Retrieved 06th September 2002 from the Word Wide Web: http://java.sun.com/products/java-media/3D/collateral/java3d.pdf

Sun Microsystems, Inc. (2002a). What is Java™ Technology. Retrieved 08th September 2002 from the Word Wide Web: http://java.sun.com/java2/whatis.htm

Sun Microsystems, Inc. (2002b). Java 3D™ API. Retrieved 06th September 2002 from the Word Wide Web: http://java.sun.com/products/java-media/3D/

Sun Microsystems, Inc. (2002c). Release Notes Java™ 2 SDK, Standard Edition Version 1.3.1. Retrieved 06th September 2002 from the Word Wide Web: http://java.sun.com/j2se/1.3/relnotes.html

Sun Microsystems, Inc. (2002d). Java™ Media Framework API. Retrieved 06th September 2002 from the Word Wide Web: http://java.sun.com/products/java-media/jmf/index.html

Sun Microsystems, Inc. (2002e). Java™ Speech API. Retrieved 13th October 2002 from the Word Wide Web: http://java.sun.com/products/java-media/speech/

Superscape plc (2001). Swerve in Action. Retrieved 6th June 2002 from the World Wide Web: http://www.superscape.com/inaction/index.asp

Sutherland, I. E. (1963). *SKETCHPAD: A Man-Machine Graphical Communication System*. PhD Thesis, Massachusetts Institute of Technology.

Sutherland, I. E. (1965). The Ultimate Display. In Kalenich, W. A. (Ed.), *Proceedings of IFIP Congress 1965: Volume 2, New York, 1965* (pp. 506-508). New York, Washington D.C.: Spartan Books.

Szeliski, R. and Tonnesen, D. (1992). Surface Modeling with Oriented Particle System. In *Proceedings of the 19th SIGGRAPH Annual International Conference on Computer Graphics and Interactive Techniques,* (pp. 185-194). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/133994.134037

Takeuchi, A. (1994). Preface to the book. In Loeffler, C. E. and Anderson, T. (Eds.), *The Virtual Reality Casebook* (p. xi). New York: Van Nostrand Reinhold.

Tanriverdi, V. and Jacob, R. J. (2000). Interacting with Eye Movements in Virtual Environments. In *Proceedings of the CHI 2000 Conference on Human Factors*

in Computing Systems, 2000, The Hague, The Netherlands (pp. 265-272). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/332040.332443

Temple, S. (1994). Thought Made Visible – the Value of Sketching. In *Co-Design Journal*, 10.11.12(1994), pp. 16-25.

Terstriep, D. E. and Jeffery, A. (1993). *The NCSA (National Center for Supercomputing Applications) Data Transfer Mechanism Programming Manual*. Retrieved 20th March 2002 from the World Wide Web: http://archive.ncsa.uiuc.edu/SDG/Software/DTM/Docs/dtm.toc.html

The Lancet (1991). Being and Believing: Ethics of Virtual Reality (Editorial). *The Lancet,* August 3*;* pp. 283-284.

Tromp, J. and Snowdon, D. (1997). Virtual Body Language: Providing Appropriate User Interfaces in Collaborative Virtual Environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, 1997, Lausanne, Switzerland* (pp. 37-44). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/261135.261143

Turner, A., Chapman, D. and Penn, A. (1999). Sketching a Virtual Environment: Modeling Using Line Drawing Interpretation. In *Proceedings of the ACM symposium on Virtual reality software and technology, London, United Kingdom* (pp. 155-161). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/323663.323687

Ucelli G. (2002). Communication and Collaboration within a VR System for Architectural Design. Unpublished PhD thesis, University of Strathclyde.

Ucelli G., Conti G., Lindsay M., and Ryder G. (2000). From "Soft" to "Hard" Prototyping: A Unique Combination of VR (Virtual Reality) and RP (Rapid Prototyping) for Design. In *Proceedings of the seventh UK VR-SIG (UK Virtual Reality Special Interest Group) 2000 conference in Glasgow, United Kingdom* (pp. 1-9).

University of Strathclyde (2002). The Virtual Environmental Laboratory. Retrieved 22nd September 2002 from the Word Wide Web: http://www.strath.ac.uk/Departments/Architecture/VEL/index.html

Wang, C. (1999). Architectural Design Thinking in Virtual Reality. In Jingwen, G. and Zhaoji, W. (Eds.), *CAADRIA '99, Proceedings of the Fourth Conference on Computer Aided Architectural Design Research in Asia, Shanghai, China* (pp. 71-80).

Ware, C. and Franck, G. (1996). Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions. In *ACM Transactions on Graphics (TOG)*, Volume 15, Issue 2 (April 1996), pp. 121-140. New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/234972.234975

Web3D Consortium (1997). *The Virtual Reality Modeling Language International Standard ISO/IEC 14772-1:1997*. Retrieved 02nd October 2002 from the Word Wide Web: http://www.web3d.org/Specifications/VRML97/

Web3D Consortium (2002). Xj3D Open Source VRML/X3D Toolkit. Retrieved 30th September 2002 from the Word Wide Web: http://www.web3d.org/TaskGroups/source/xj3d.html

Weber, R. (1995). On the Aesthetics of Architecture: a Psychological Approach to the Structure and the Order of Perceived Architectural Space. Great Britain: Athenaeum Press.

Wesche, G. and Droske, M. (2000). Conceptual Free-Form Styling on the Responsive Workbench. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, Seoul, Korea* (pp. 83-91). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/502390.502406

Wesche, G. and Droske, M. (2001). FreeDrawer – A Free-Form Sketching System on the Responsive Workbench. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, Banff, Canada* (pp. 167-174). New York:

ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/505008.505041

Wu, L., Oviatt, S. and Cohen, P. (1999). Multimodal Integration: a Statistical View. In *IEEE Transactions on Multimedia*, 1, 4, pp. 334-342.

Yessios, C. I. (1987). The Computability of Void Architectural Modelling. In Kalay, Y. E. (Ed.), *Computability of Design – Principles of Computer-Aided Design* (pp. 141-172). New York: Wiley-Interscience.

Zampi, G. and Conway, L. M. (1995). *Virtual Architecture*. London: McGraw-Hill.

Zeleznik, R. and Forsberg, A. (1999). UniCam — 2D Gestural Camera Controls for 3D Environments. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics, 1999 , Atlanta, Georgia, USA* (pp. 169-173). New York: ACM Press. Retrieved 14th October 2002 from the World Wide Web: http://doi.acm.org/10.1145/300523.300546

Appendix A: The Content of the Chat during the Experiment

A.1 Overview of the Content of the Chat

The following pages report the text of the chat recorded during the experiment described in Chapter 8. The first part of the text was recorded during the introductory session (lines 1 - 20), the second during the experiment (lines 21 – 171).

Four users are registered in the session, three students named Ross, Eddie and Christoph, and one of the authors, Giuseppe, was included to supervise the outcome of the experiment. He intervenes at points to announce the beginning of the experiment (line 21), to communicate the crash of the application run on the Sgi machine (lines 37, 39, 154), to warn students of the imminent end of the experiment (lines 149, 151, 153 and 157, 158) and to announce the end of the test (lines 167, 168).

From the analysis of the text it is possible to appreciate some interesting features of the system, for example line 39 stresses the network persistence mechanism described in Section 6.2.4 of the companion thesis (Ucelli, 2002).

The text also contains suggestions for improvements highlighted by the students. Specifically need to use a virtual compass or a grid to help localise the object is reported at lines 72 and 83. The development of a mechanism to place objects perpendicularly is suggested at line 73. The further development of the navigation mode is suggested at line 83.

Finally the text shows the relative unreliability of JCAD-VR on the Sgi Irix platform. While the two PCs used in the experiment proved to be reliable throughout the Sgi Onyx2 crashed three times (lines 37, 150, 154).

A.2 The Text of the Chat

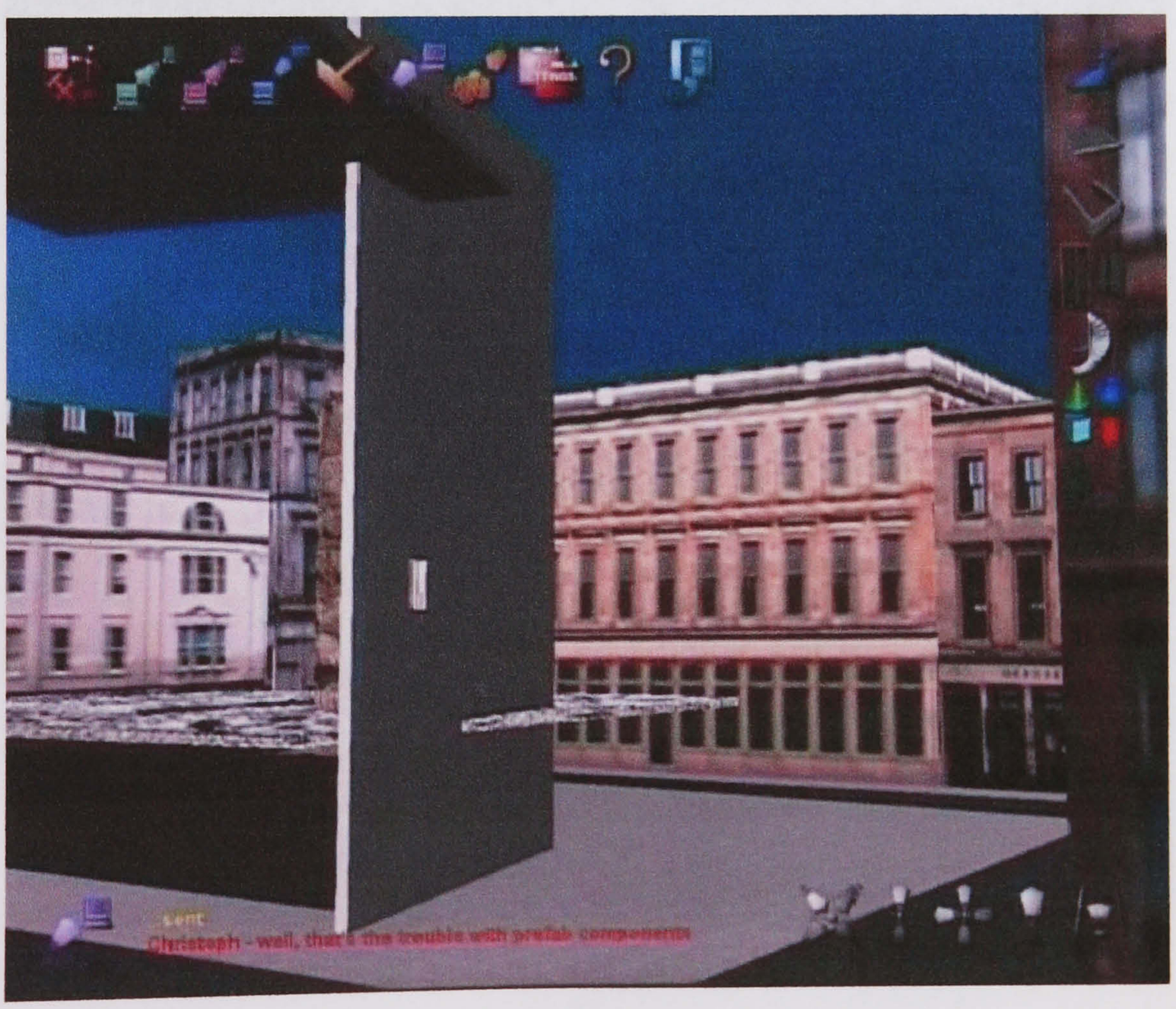
1. Nothing received
2. **Eddie** - hi we are here...giuseppe ????????????& ed...
3. **Ross** - hello
4. **Christoph** - open chat window for sketching, please
5. **Eddie** - are you asleep Christoph ?
6. **Christoph** - i am not sleeping, just busy

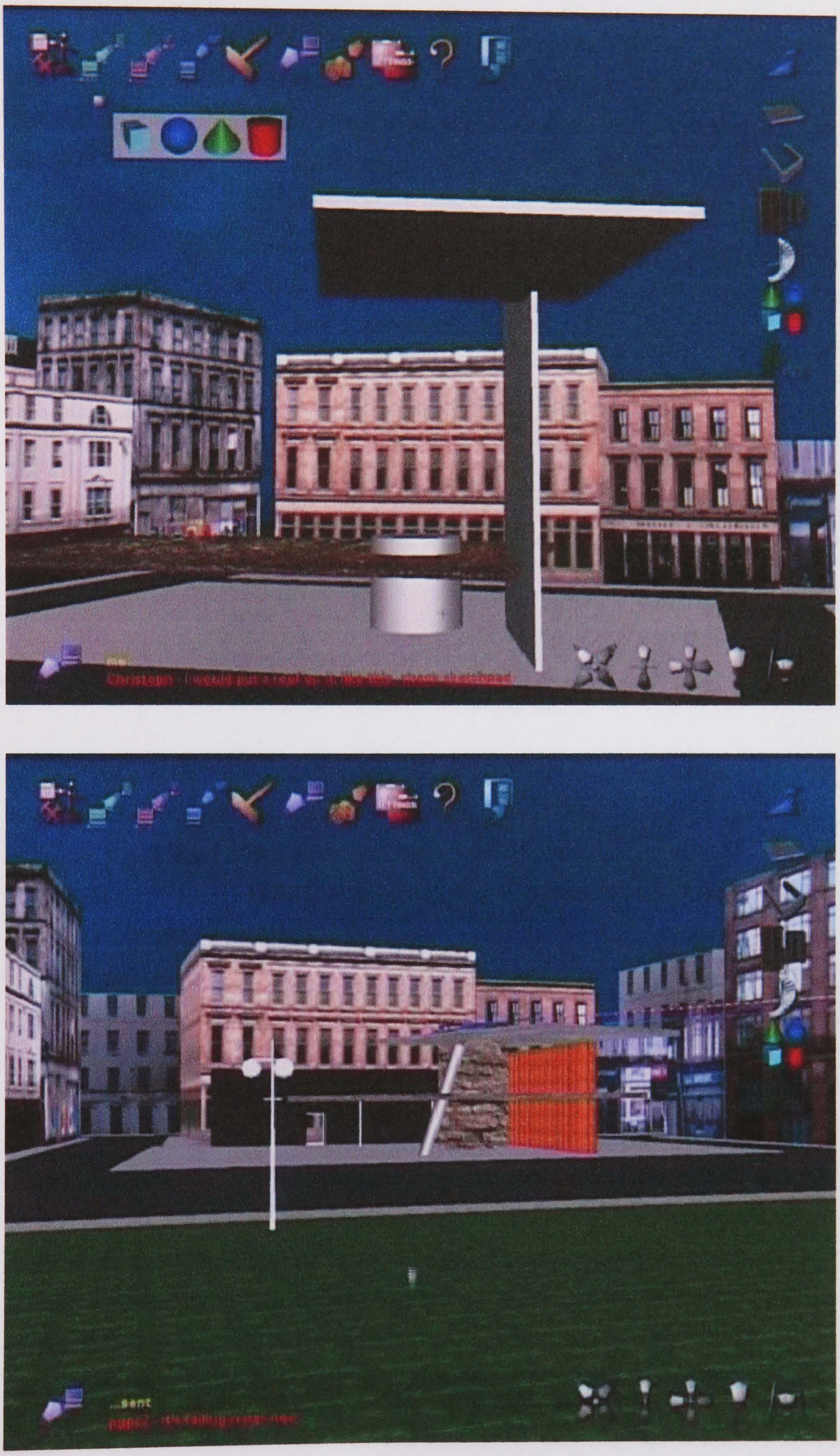
7. **Christoph** - who is sketching?\\
8. **Eddie** - not me
9. **Christoph** - was me, Christoph
10. **Christoph** - was this somebody's design?????
11. **Christoph** - eddie are you Edward ??
12. **Christoph** - eddie is that you?? sketching with red??
13. **Christoph** - i am white, Christoph
14. **Christoph** - who created stair??\anybody answere me\
15. **Ross** - it wasnt me
16. **Christoph** - cleaned up the mess ?:)
17. **Ross** - who is cleaning up the mess??
18. **Eddie** - you didnt like it??
19. **Eddie** - um probably
20. **Christoph** - i am talking about the geometry mistake??
21. **Giuseppe** - hi guys , shall we start?? go...
22. **Ross** - ?ok
23. **Edward** - ok
24. **Christoph** - ok
25. **Ross** - ok
26. **Eddie** - ?how about a big wall??
27. **Ross** - thats ok with me
28. **Ross** - ok
29. **Christoph** - hi, i',m there ?:)
30. **Ross** - i put a cylinder on the coner
31. **Eddie** - what is it made of??
32. **Eddie** - this is a kiosk..... how big??
33. **Eddie** - is it selling hot dogs??
34. **Ross** - □is that too big??????????????????
35. **Eddie** - anybody??
36. **Eddie** - probably
37. **Giuseppe** - guys Christoph crashed he's re-booting
38. **Ross** - ok so what will do do then??
39. **Giuseppe** - guys if someone crashes keep on working it's no problem. Once on-line again the system loads the content of the world automatically
40. **Christoph** - hi, guys, guess who's back??!
41. **Ross** - its looking very abstract
42. **Christoph** - i would put a roof on it, like this, check sketchpad
43. **Ross** - ??????????????????????where will the entrance be??
44. **Christoph** - na, the whole thing is the kiosk, like a pavillion
45. **Christoph** - have to check the roof it does look a little bit nasty
46. **Eddie** - what's going on??
47. **Ross** - ?are theyre any windows??
48. **Christoph** - well, that's the trouble with prefab components
49. **Christoph** - where's glas?
50. **Christoph** - ross, are you ross?\
51. **Ross** - yes i tink so
52. **Ross** - opps i del the wall
53. **Christoph** - sometimes is tricky, where are u?
54. **Eddie** - is this still a small kiosk??
55. **Ross** - not any more ?:?)
56. **Christoph** - na, not really, small elements are tricky, though
57. **Christoph** - can u copy elements?????
58. **Christoph** - barcelona pavillion i guess ;)
59. **Ross** - do we want steps up to the platform??????????????
60. **Eddie** - yes why not.. big ones
61. **Christoph** - why not, give it a shot

62. **Eddie** - steps??
63. **Ross** - ?stpes or st\irs??????????
64. **Eddie** - ones like this
65. **Christoph** - can we reduce the height of the cylinder, please
66. **Ross** - ok
67. **Eddie** - what should the stairs be made from??
68. **Christoph** - you can't change materials of symbols
69. **Eddie** - no
70. **Christoph** - na, can't apply material to stairs
71. **Eddie** - i think we need to do some site analysis
72. **Christoph** - where;s south, would really like a grid or something
73. **Christoph** - hard to get objects perpenticular
74. **Christoph** - what's that ?
75. **Ross** - what??
76. **Christoph** - that grey misobject,
77. **Christoph** - the design is a mess, gonna be a labyrint
78. **Eddie** - where is the entrance??
79. **Christoph** - guess?i don't know it's on the lifeft side, from my point of view, i will highlight it
80. **Christoph** - that??!!!!
81. **Christoph** - you see?
82. **Eddie** - that roof is floating
83. **Christoph** - navigation mode would be really nice, like in airplanes, like 2 o'clock high, and a definite no9rthpoint
84. **Ross** - do we need the large wall on the left??
85. **Christoph** - where?
86. **Ross** - there
87. **Eddie** - minimalists
88. **Christoph** - who's creatring these massive walls
89. **Christoph** - all the time?
90. **Eddie** - i made one of them.
91. **Ross** - i made 1
92. **Christoph** - skyhigh, lets go 3d
93. **Christoph** - anybody, please answere, what's that on the left?
94. **Christoph** - must be a wall with negatyive thickness including windows?
95. **Ross** - the wall that i just deleted??
96. **Ross** - ?the roof is floating now
97. **Eddie** - i think it should be lower
98. **Ross** - should it sit on the cylinder??
99. **Eddie** - is the roof a creche then??
100. **Christoph** - youyoulook at it, kiosk??
101. **Eddie** - there is no entrance to the park??!?
102. **Ross** - did we decide on an entrance yet??
103. **Eddie** - no , but it needs one i think
104. **Ross** - where??
105. **Eddie** -
106. **Eddie** - nice door sould be wider though
107. **Ross** - □???????is that ok??
108. **Eddie** - like it ys
109. **Christoph** - other side towards the buildings
110. **Ross** - what do we have happening on the street??
111. **Ross** - ???
112. **Eddie** - what have i missed??
113. **Ross** - what is happening??
114. **Eddie** - i thi9nk christoph is going mad
115. **Ross** - is it going to be open to the street??
116. **Eddie** - is it ??

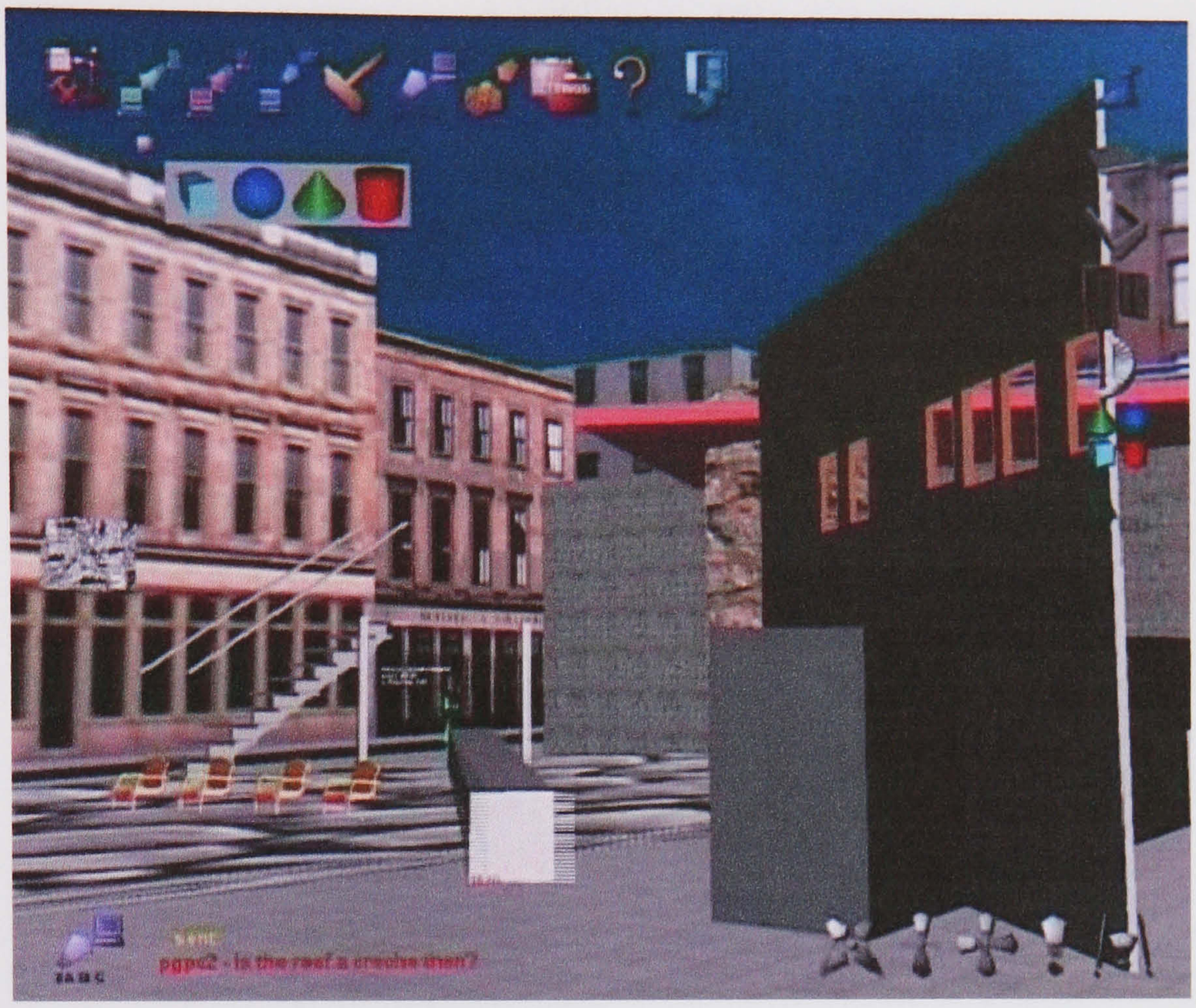
117. **Ross** - ?what do u think??
118. **Eddie** - what happened to the wall with the windows??
119. **Eddie** - timber?!
120. **Ross** - for the wall on left??
121. **Eddie** - yeah
122. **Ross** - pannels???
123. **Eddie** - yes thats good
124. **Eddie** - did anyone really ant the slide??
125. **Ross** - not me
126. **Ross** - should the back wall be higher??
127. **Eddie** - just a bit
128. **Eddie** - maybe make a balustrade to the roof
129. **Eddie** - or maybe not
130. **Christoph** - single storey for a kiosk,please
131. **Eddie** - lighten up Christoph
132. **Eddie** - it's falling water now
133. **Ross** - ?:?)
134. **Eddie** - how about it doubles as a beer garden??
135. **Ross** - in a kiosk??
136. **Eddie** - why not??
137. **Ross** - should there be something on the leftside??
138. **Eddie** - hmmm just a small intervention
139. **Eddie** - whoose is the cylinder??
140. **Ross** - which 1????
141. **Eddie** - the big un
142. **Ross** - me
143. **Ross** - should it go??
144. **Eddie** - i think we should buy up some of the park for a new site then
145. **Eddie** - put it on stilts
146. **Eddie** - where is Christoph??
147. **Eddie** - i was just getting used to that
148. **Eddie** - not really
149. **Giuseppe** - guys 10 min to go from now
150. **Christoph**: ?i crashed again
151. **Giuseppe** - got it ???
152. **Eddie** - are we keeping this monstrosity??
153. **Giuseppe** - I repeat 10 mins to go from now ok ?
154. **Giuseppe** - chris is crashed again...sgi is bit unstable
155. **Ross** - guys hang on 4 a min
156. **Christoph** - hi guys, finally back
157. **Giuseppe** - ok now everything is fine u have 5 min
158. **Giuseppe** - repeat again 5 mins to go, ok ??
159. **Eddie** - ok
160. **Christoph** - ok
161. **Ross** - ?what is in the middle of the park now???
162. **Christoph** - back again
163. **Ross** - ?ok
164. **Christoph** - are we not takiung that alittle bit far?
165. **Eddie** - is that the motorway??
166. **Ross** - thats what i thought
167. **Giuseppe** - I'm deleting all objects now...
168. **Giuseppe** - that's you guys...thankx very much 4 your time !!
169. **Ross** - ok
170. **Christoph** - 0k
171. **Eddie** - ok!

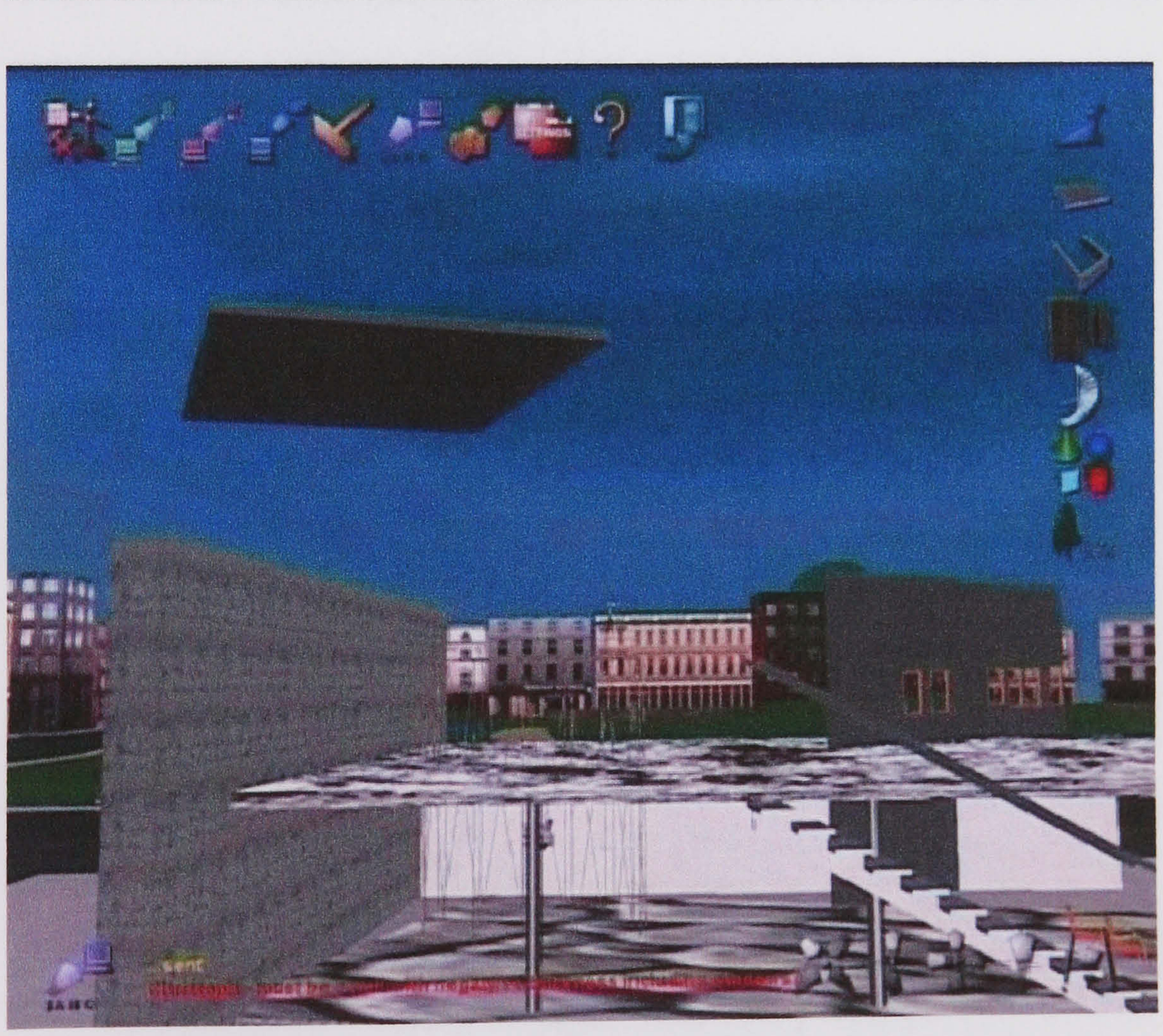
Appendix B: Screenshots from the Experiment

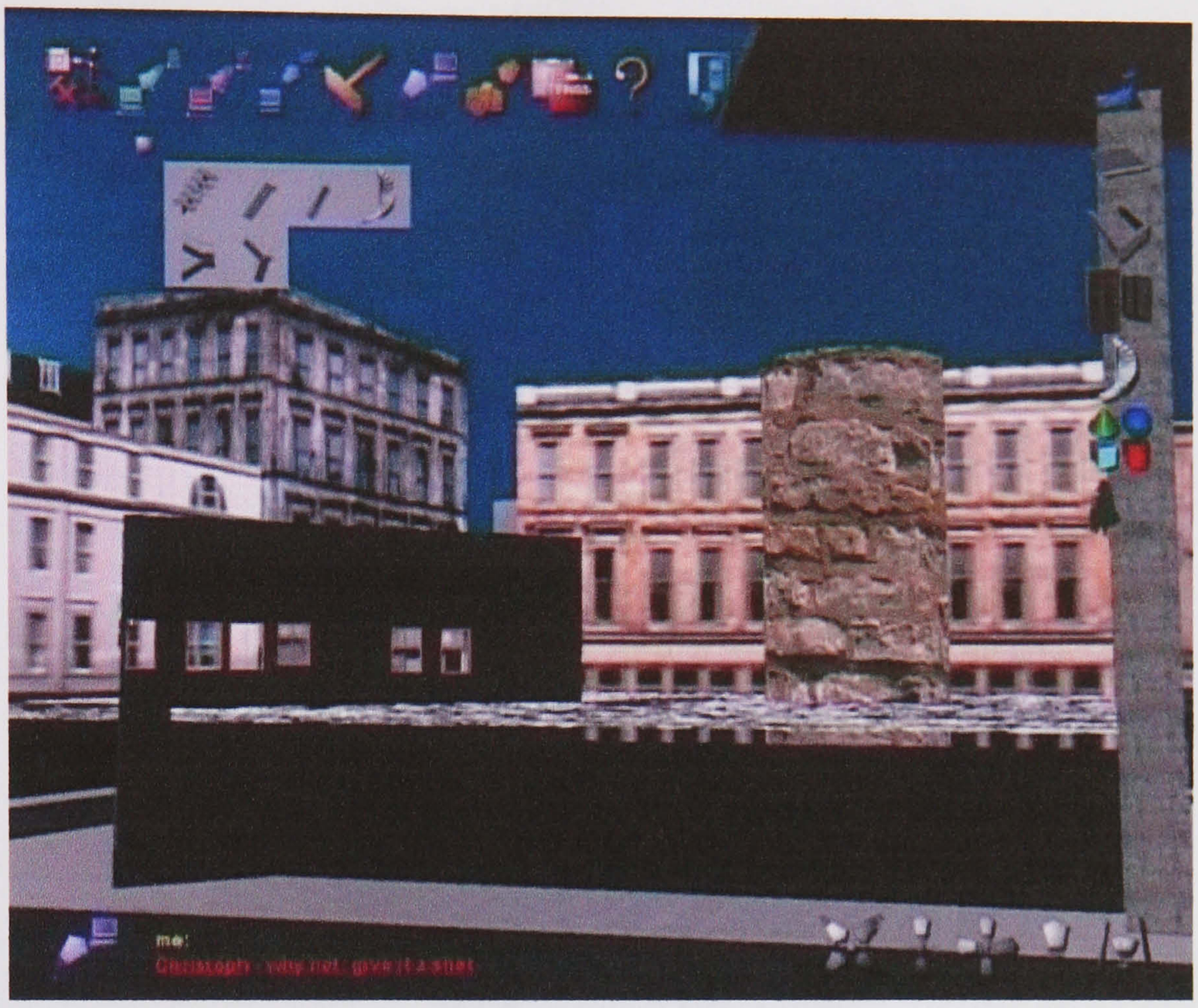


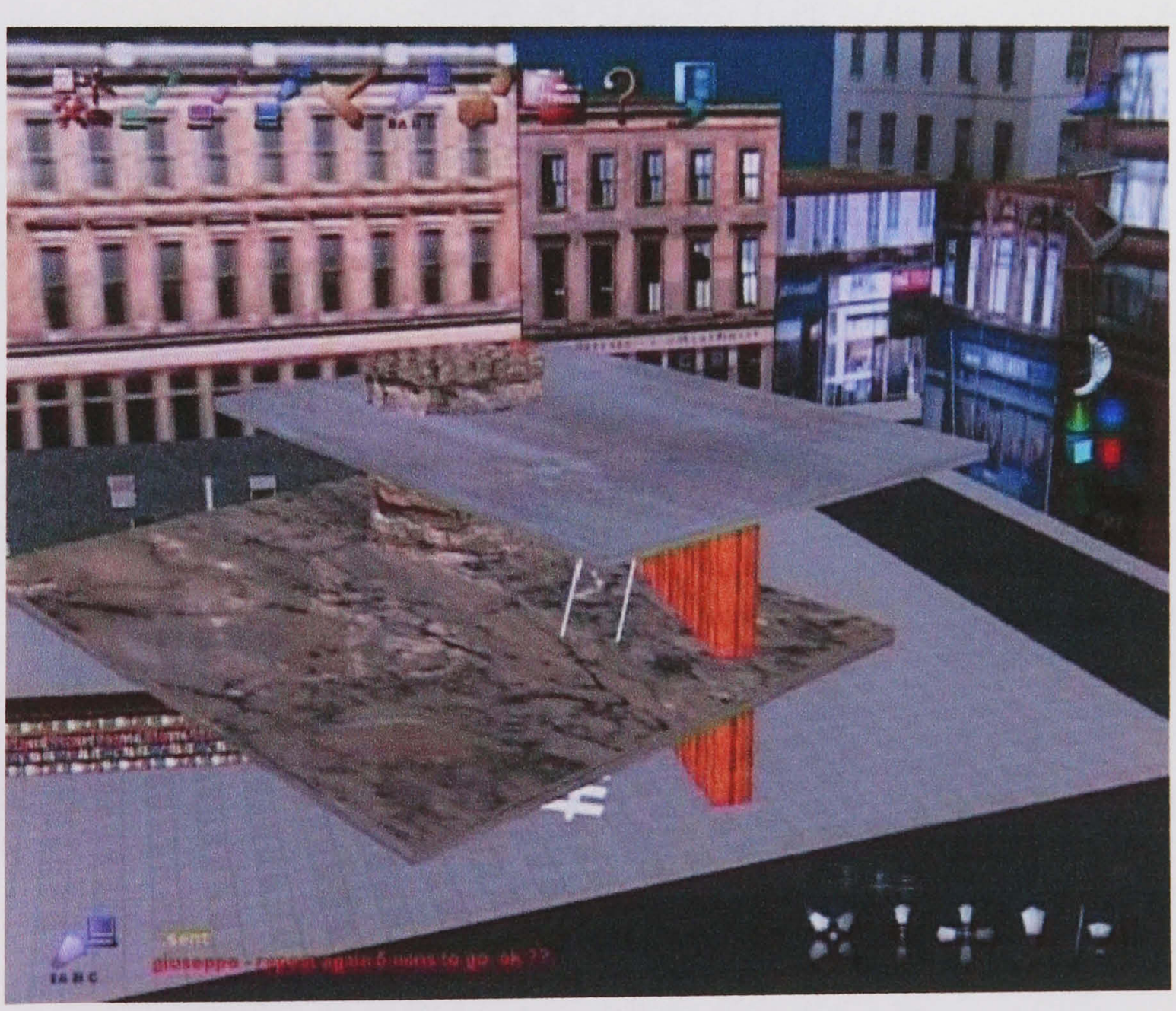



material: default material
cost $:500.00
surface m2 :0.92

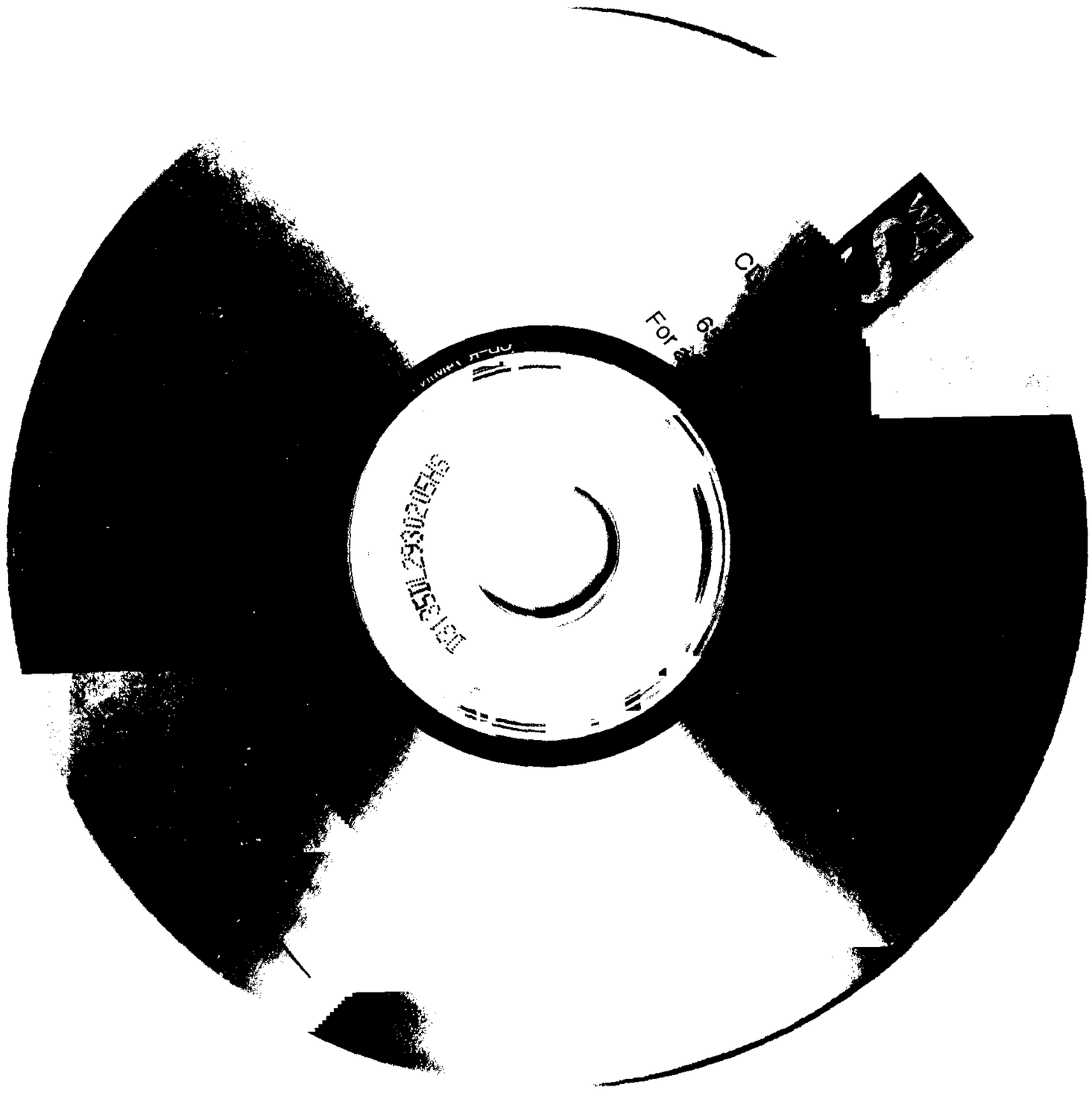





me:


Sent

Appendix C: The CD-ROM with the Video of the Experiment



Appendix D: Specification of Hardware and Software used for the Experiment

D.1 Specification Computer n.1 (client)

Hardware

- Model: Dell Precision 530
- Processor(s): 2 x Intel® Xeon™ 1.4 GHz
- Memory: 512 MB Ram
- Graphic Card: nVidia Quadro2 EX with 32Mb Video Memory

Software

- Operative System: Windows 2000 SP2
- Java™ 2 SDK Environment, Standard Edition 1.3.1_01
- Java 3D™ 1.2.1_03 SDK (OpenGL Version)

D.2 Specification Computer n.2 (client)

Hardware

- Model: Viglen Genie
- Processor(s): Intel® Pentium™ III 933 MHz
- Memory: 256 MB Ram
- Graphic Card: nVidia RIVA TNT2-Model64 with 32Mb Video Memory

Software

- Operative System: Windows 2000 SP2
- Java™ 2 SDK Environment, Standard Edition 1.3.1_01
- Java 3D™ 1.2.1_03 SDK (OpenGL Version)

D.3 Specification Computer n.3 (client)

Hardware

- Model: Sgi Onyx2
- Processor(s): 12 x MIPS R12000 400MHz
- Memory: 6GB
- Graphic Card: 2 x InfiniteReality

Software

- Operative System:
- Java™ 2 SDK 1.3.1 for Sgi Irix: MR Release
- Java 3D™ 1.2.1 SDK for Sgi Irix: MR Release

D.4 Specification Computer n.4 (server)

Hardware

- Model: ® O2
- Processor(s): MIPS R5000 300MHz
- Memory: 128 MB
- Graphic Card: CRM graphics

Software

- Operative System:
- Java™ 2 SDK 1.3.1 for Sgi Irix: MR Release
- Java 3D™ 1.2.1 SDK for Sgi Irix: MR Release

D.5 External Libraries used in JCAD-VR

- Prominence library (Hughes et al., 1997)
- Xj3D loader release 3 (Web3D Consortium, 2002)

Appendix E: Questionnaires Proposed for the Cross-University Experiment

QUESTIONNAIRE - Group 1 - (*JCAD-VR Collaborative*)

Part 1:

- Session: ☐ Morning
  ☐ Afternoon
- Year of study:.....
- What kind of software do you normally use? ☐ Microsoft Office

  ☐ CAD/CAAD packages
  If selected please specify:

.....

.....

☐ Other
If selected please specify:

.....

.....

- Have you had any experience with *Virtual Reality* before?
  - ☐ Yes, many times
  - ☐ Yes, few times
  - ☐ Yes, once
  - ☐ No

Part 2:

1. How do you find navigating within the virtual environment using this software?
   - ☐ Very easy
   - ☐ Easy
   - ☐ Fairly easy
   - ☐ Difficult – specify why.....

.....

.....

☐ Very difficult – specify why.....

.....

.....

2. How did you find creating objects using this software?

☐ Very easy

☐ Easy

☐ Fairly easy

☐ Difficult – specify why.....

.....

.....

☐ Very difficult – specify why.....

.....

.....

3. How did you find modifying objects using this software?

☐ Very easy

☐ Easy

☐ Fairly easy

☐ Difficult – specify why.....

.....

.....

☐ Very difficult – specify why.....

.....

.....

4. How did you find working with your partner simultaneously?

☐ Very easy

☐ Easy

☐ Fairly easy

☐ Difficult – specify why.....

.....

.....

☐ Very difficult – specify why.....

.....

.....

5. Did you find it interesting to work with a partner over the net?

☐ Yes, a lot

☐ Yes

☐ Not so much – specify why.....

.....

.....

☐ No – specify why.....

.....

.....

☐ No, absolutely – specify why.....

.....

.....

6. In your view did you and your partner have enough exchange of information in order to accomplish the design task?

☐ Yes, a lot

☐ Yes

☐ Not so much – specify why.....

.....

.....

☐ No – specify why.....

.....

.....

☐ No, absolutely – specify why.....

.....

.....

7. Do you think that your partner could have misinterpreted your design ideas during the collaboration?

☐ Yes, many times we had problems and we could not explain our ideas

Thanks for your kind help. The information that you have provided will be very useful for us.

QUESTIONNAIRE - Group 2 - (*JCAD-VR Stand-alone*)

Part 1:

- Session: ☐ Morning
  ☐ Afternoon
- Year of study:.....
- What kind of software do you normally use? ☐ Microsoft Office

  ☐ CAD/CAAD packages
  If selected please specify:

.....

.....

  ☐ Other
  If selected please specify:

.....

.....

- How frequently do you use e-mails:
  - ☐ Very often
  - ☐ Often
  - ☐ Sometimes
  - ☐ Seldom
  - ☐ Never

- Have you had any experience with *Virtual Reality* before?
  - ☐ Yes, many times
  - ☐ Yes, few times
  - ☐ Yes, once
  - ☐ No

Part 2:

1. How do you find navigating within the virtual environment using this software?

   ☐ Very easy

   ☐ Easy

   ☐ Fairly easy

   ☐ Difficult – specify why.....

.....

.....

   ☐ Very difficult – specify why.....

.....

.....

2. How did you find creating objects using this software?

   ☐ Very easy

   ☐ Easy

   ☐ Fairly easy

   ☐ Difficult – specify why.....

.....

.....

   ☐ Very difficult – specify why.....

.....

.....

3. How did you find modifying objects using this software?

   ☐ Very easy

   ☐ Easy

   ☐ Fairly easy

   ☐ Difficult – specify why.....

.....

.....

   ☐ Very difficult – specify why.....

.....

.....

4. How did you find the collaboration with your partner?

☐ Very easy

☐ Easy

☐ Fairly easy

☐ Difficult – specify why.....

.....

.....

.....

☐ Very difficult – specify why.....

.....

.....

5. Did you find it interesting to work with a partner over the net?

☐ Yes, a lot

☐ Yes

☐ Not so much – specify why.....

.....

.....

☐ No – specify why.....

.....

.....

.....

☐ No, absolutely – specify why.....

.....

.....

6. In your view did you and your partner have enough exchange of information in order to accomplish the design task?

☐ Yes, a lot

☐ Yes

☐ Not so much – specify why.....

.....

.....

☐ No – specify why.....

.....

.....

☐ No, absolutely – specify why.....

.....

.....

7. In your view is the collaborative method you have used in this experiment (software+e-mails) a feasible approach for collaborative design?

   ☐ Yes, I found it effective

   ☐ Yes, but.....

.....

.....

   ☐ No, but.....

.....

.....

   ☐ No, I found it unpractical

8. Do you think that your partner could have misinterpreted your design ideas during the collaboration?

   ☐ Yes, many times we had problems and we could not explain our ideas

   ☐ Yes, but eventually we succeeded in explaining our ideas

   ☐ No, but the communication media available were just enough to show our ideas

   ☐ No, the communication media available allowed us to show our ideas

9. Do you think that this software could help in proposing design solutions at an initial, conceptual, stage in relation with the urban context?

   ☐ Yes, a lot

   ☐ Yes

   ☐ Not so much – specify why.....

.....

.....

☐ No – specify why.....

.....

.....

☐ No, absolutely – specify why.....

.....

.....

10. We warmly encourage any kind of comment on the software and on the experiment itself:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thanks for your kind help. The information that you have provided will be very useful for us.

QUESTIONNAIRE - Group 3 - (*Sketching)*

Part 1:

- Session: ☐ Morning
  ☐ Afternoon
- Year of study:.....

- Are you confident in using:

a printer: ☐ Yes
☐ Fairly
☐ No

a scanner: ☐ Yes
☐ Fairly
☐ No

- How frequently do you use e-mails:
  ☐ Very often
  ☐ Often
  ☐ Sometimes
  ☐ Seldom
  ☐ Never

Part 2:

1. How did you find expressing your design ideas and solutions sketching?
   ☐ Very easy
   ☐ Easy
   ☐ Fairly easy
   ☐ Difficult – **specify why**.....
   .....
   .....
   ☐ Very difficult – **specify why**.....

.....

.....

2. How did you find the collaboration with your partner?

☐ Very easy

☐ Easy

☐ Fairly easy

☐ Difficult – specify why.....

.....

.....

☐ Very difficult – specify why.....

.....

.....

3. Did you find it interesting to work with a partner over the net?

☐ Yes, a lot

☐ Yes

☐ Not so much – specify why.....

.....

.....

☐ No – specify why.....

.....

.....

☐ No, absolutely – specify why.....

.....

.....

4. In your view did you and your partner have enough exchange of information in order to accomplish the design task?

☐ Yes, a lot

☐ Yes

☐ Not so much – specify why.....

.....

.....

☐ No – specify why.....

.....

.....

☐ No, absolutely – specify why.....

.....

.....

.....

5. Do you think that your partner could have misinterpreted your design ideas during the collaboration?

   ☐ Yes, many times we had problems and we could not explain our ideas

   ☐ Yes, but eventually we succeeded in explaining our ideas

   ☐ No, but sketches were just enough to show our ideas

   ☐ No, sketches allowed us to show our ideas

6. In your view is the collaborative method you have used in this experiment (sketches+e-mails) a feasible approach for collaborative design?

   ☐ Yes, I found it effective

   ☐ Yes, but.....

.....

.....

   ☐ No, but.....

.....

.....

   ☐ No, I found it unpractical

7. We warmly encourage any kind of comment on the collaborative method and on the experiment itself:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Thanks for your kind help. The information that you have provided will be very useful for us.

Appendix F: Notes for Tutors and Troubleshooting

F.1 Notes for tutors

A General Record

In both universities it would be appropriate to record the experiment with a video camera (preferably a digital one) and to take some pictures with a digital camera.

Other Notes

- It would be useful to set up a communication line between the tutors via ICQ or similar chat-lines during the experiment (addresses would be required).
- Both universities have to provide email access for groups 2 and 3, and also a printer, a scanner at A4, A4 paper and sketching materials for group 3.
- At the end of **each session** tutors should copy the content of the following folders where all the screenshots, back-up files, final images and emails are located:

  1. *JCAD-VR\automatic_save,*
  2. *JCAD-VR\scene*
  3. *JCAD-VR\capture*
  4. Folder with the emails of group 2
  5. Folder with the emails of group 3

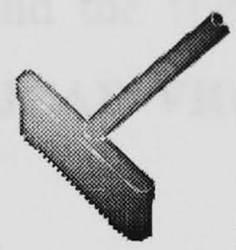
Before deleting the files contained in these directories their content has to be saved in another directory in order to start the second session with empty folders. This will avoid mixing up the files produced by the different students in the two sessions.

F.2 Notes and Troubleshooting for group 1

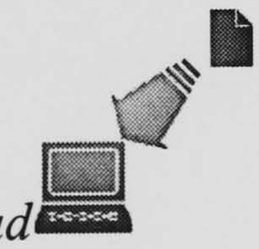
Loading jcad Files when Starting JCAD-VR

JCAD-VR will automatically load the environment to be shared by the students working synchronously. This means that new objects created within JCAD-VR will be visualized in every computer joining the concurrent design session. These are the instructions to follow to load the scene at the beginning of the exercise:

1. Start JCAD-VR
2. During the loading of JCAD-VR a window will appear to allow the selection of a VRML file, click on CANCEL
3. After few seconds the scene will be loaded, and when all the partners joining the session are ready, every object created will be visualized and shared with the others.

IMPORTANT NOTE: Before starting to create objects in the collaborative mode all the users have to be connected with the application running and connected to the server. **Make sure that all your partners in the collaborative session are already connected properly before starting to create objects**. The best way to do thsi is simply by monitoring the text of the chat; if you are able to chat with your partners you will be also able to share objects. If geometries were already created before your partners were connected we suggest you to delete all the objects by clicking on the *delete all*  icon, unless you wish to continue by sharing the objects with your partners.

Loading Files without Restarting JCAD-VR

1. Load your new file by clicking on the *load*  icon

2. If the system is in collaborative mode, once you load the file the other user connected will receive the scene. Therefore this operation has to be done **by only one person involved in the collaborative session**. Double check with your partner by chat to see if they have the scene upgraded before starting to work.

In case of the Crashing or Interruption of JCAD-VR while a Student is Working with a Partner:

1. Close JCAD-VR by clicking on the *exit*  icon or using the CTRL+C button sequence on the MS-DOS console.
2. Restart JCAD-VR
3. During the loading of JCAD-VR a window will appear to allow selection of a file, click on CANCEL
4. Wait for some seconds and the software will automatically upgrade the scene.

Saving with JCAD-VR

In order to make back-up files, JCAD-VR will automatically save students' work every 5 minutes and the files will be placed (with a name and sequential number) in the directory **JCAD-VR\automatic_save**. Nevertheless students can also save their design by clicking on the *save* icon  and by giving these names to their files: **group1_01**, **group1_02, ...group1_20** etc... in the **JCAD-VR\scene** directory.

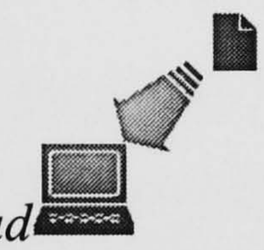
IMPORTANT LAST NOTE: *For normal reasons JCAD-VR can also crash or halt if some unexpected operations have been done. We would therefore recommend saving your work frequently.*

F.3 Notes and Troubleshooting for group 1

Loading jcad Files when Starting JCAD-VR

JCAD-VR automatically loads the environment where you have to place your design. You need only to start JCAD-VR, and the environment will appear.

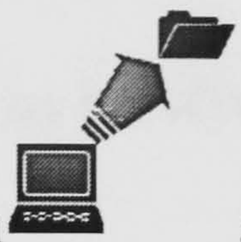
Loading Files without Restarting JCAD-VR

The loading routine of JCAD-VR is such that more than one jcad file cannot be loaded together. You can load your new file by clicking on the *load*  icon and by selecting the file to open (jcad and wrl formats are supported).

In case of the Crashing or Interruption of JCAD-VR

1. Close JCAD-VR by clicking on the *exit*  icon or using the CTRL+C button sequence on the MS-DOS console.
2. Restart JCAD-VR
3. During the loading of JCAD-VR a window will appear to allow the selection of a file, click on CANCEL
4. Load the jcad file you were working on again (for instructions on how to load a file in JCAD-VR see *Loading jcad files when starting JCAD-VR* on the previous page).

Saving with JCAD-VR

Students can save their design by clicking on the *save* icon  and by giving these names to their files: **group2_01**, **group2_02, …group2_20** etc… in the **JCAD-VR\scene** directory.

IMPORTANT LAST NOTE: *For normal reasons JCAD-VR can also crash or halt if some unexpected operations have been done. We would therefore recommend saving your work frequently.*

Appendix G: The JCAD-VR Project's Publications

The following pages include conference papers reporting on the JCAD-VR project:

1. Conti, G, Ucelli, G. and Maver, T. (2001). JCAD-VR: Java Collaborative Architectural Design Tool in VR. In H. Penttilä (Ed.), *Proceedings of Architecture Information Management, 19th Conference of eCAADe, Helsinki, 2001* (pp. 454-459). Espoo, Finland: Otamedia Oy.

2. Ucelli, G., Conti, G., Petric, J. and Maver, T. (2002). Real Experiences of Virtual Worlds. In D. Marjanović (Ed.), *Proceedings of the Design 2002, 7th International Design Conference, 2002, Cavtat* (pp. 561-566). Zagreb, Croatia: Faculty of Mechanical Engineering and Naval Architecture, Zagreb and The Design Society, Glasgow, UK.

3. Petric, J., Maver, T., Conti, G. and Ucelli, G. (2002). Virtual Reality in the Service of User Participation in Architecture. In K. Agger, P. Christiansson and R. Howard (Eds.), *Distributing Knowledge in Building. Proceedings of CIB W78 Conference, Aarhus* (pp. 217-224). Aarhus, Denmark: Aarhus School of Architecture and Centre for Integrated Design.

4. Petric, J., Ucelli, G. and Conti, G (2002). Real Teaching and Learning through Virtual Reality. In K. Koszewski and S. Wrona (Eds.), *Design e-ducation. Proceedings of the 20th Conference of eCAADe, Warsaw, 2002* (pp. 72-79). Warsaw, Poland: Drukarnia Braci Ostrowskich.

5. Conti, G., Ucelli, G. and Petric, J. (2002). JCAD-VR: a collaborative design tool for architects. In *Proceedings of the 4th international conference on Collaborative virtual environments, Bonn, Germany, 2002* (pp. 153-154). New York: ACM Press.

6. Petric, J., Ucelli, G. and Conti, G. (2002). Participatory Design in Collaborative Virtual Environments. Accepted paper at the 6th SIGRADI Conference, Caracas.

7. Petric, J. Conti, G. and Ucelli, G. (2003). Designing Within Virtual Worlds. Accepted paper at the CAAD Futures 2003 Conference, Tainan, Taiwan

Conti, G, Ucelli, G. and Maver, T. (2001). JCAD-VR: Java Collaborative Architectural Design Tool in VR. In H. Penttilä (Ed.), ***Proceedings of Architecture Information Management, 19th Conference of eCAADe, Helsinki, 2001*** **(pp. 454-459). Espoo, Finland: Otamedia Oy.**

Go to contents 16

JCAD-VR: Java Collaborative Architectural Design Tool in Virtual Reality

A Java3D based scalable framework for real-time, multi-platform VR environments

CONTI, Giuseppe; UCELLI, Giuliana; MAVER, Tom
ABACUS, University of Strathclyde, Glasgow, UK
http://www.strath.ac.uk/Departments/Architecture/

This paper proposes a framework that provides the architect with a tool that uses Virtual Reality (VR) as part of the design path. It offers the possibility to deploy a system capable of assisting the design profession during the early stages of the design process. This way VR becomes the means for a new experience where the architect can, free from constraints of the 2D world, create and manipulate the space she/he is designing.
The idea upon which JCAD-VR is being built is that all the users present in the virtual world have to be able to share the same virtual environment in a "transparent fashion" where the user interface, instead of the traditional menu/windows based layout, it is part of the virtual world itself.
The aim is to provide the designer with a tool for creating 3D-shapes in a shared VR environment, thus allowing the design to be shared as it evolves.
Keywords: *Collaborative Design, Virtual Reality, Java 3D, Distributed Environment.*

Introduction

Traditionally many architects have experienced the need to prove their design proposals using physical models: even the most accurate 2D paper representations are usually not suitable enough to explain and transmit the complexity of some architectonic ideas.

If the use of a third dimension is nowadays part of the daily practice, the "CAAD community" is only now experiencing the move from static representation, based on 2D renderings or pre-recorded animations (considered as a sequence of 2D images), to dynamically generated 3D representations. Real-time navigation and interaction, typical of VR environments, provide just that fluent interface and that entirety in the exploration of the design proposal that is the main lack in all CAD packages commonly in use.

Furthermore, the increasing growth of computational resources and hardware power eases the access to desktop VR applications making it a truly feasible approach in everyday practice.

Although VR is a quite mature technology, it is seldom utilised throughout the design process: often in fact it is just used as a more powerful presentation technique. Moreover, the recent growth of network-based virtual communities and the use of avatars have brought a new level of complexity to the meaning of virtuality, providing the technology for remote presence and collaborative experiences.

Background

In two case studies part of a previous research project of the authors (Ucelli, Conti, Lindsay and Ryder, 2000), the research team worked closely with both architects and engineers experimenting the use of VR in a real ongoing project.

During the creation of 3D models continuous access to all the information regarding the design progress and changing priorities were provided. Architects and engineers were invited to evaluate and discuss the project in a Virtual Reality lab.

The outcome of the research confirmed what other authors had stated (Dorta and LaLande, 1998): "the design process is made up of two activities which are graphic ideation or conceptual design and communication [...] the first is a formative process dealing with creating and evolving ideas [active phase]; the second is a descriptive process aiming at presenting to others fully-formed ideas [passive phase] "[1].

Unfortunately the result of the experience, if on the one side it has proved the huge potential of VR for reducing designs "flaws", on the other side it has highlighted that the present use of VR is limited only to the last phase of the design process, what we call the passive phase, when the visualization takes place and the result of the design process is eventually shown to the client (fig 1).

It is evident from the Figure 1 that the use of modeling and Rapid Prototyping (RP) is confined, with only a few exceptions (such in the innovative architectural practice of Frank O. Gehry) at the end of the creation process after which all the design choices have been taken.

The idea upon which the JCAD-VR framework is founded is to anticipate the use of VR within the active phase thus taking full advantage of the technology. The aim is to provide the designer with a tool for creating 3D-shapes in a shared VR environment, thus allowing the design to be shared as it evolves. This paper will report the present state of the JCAD-VR framework and will highlight its future development.

JCAD-VR: a framework

The idea upon which JCAD-VR is being built is that all the users present in the virtual world have to be able to share the same virtual environment in a "transparent fashion" where the user interface (UI), instead of the traditional menu/windows based layout, it is part of the virtual world itself.

The entire project is based on client-server architecture where every user logs into a virtual world and starts sharing designing tasks with other users. The entire structure is organised in an object-oriented

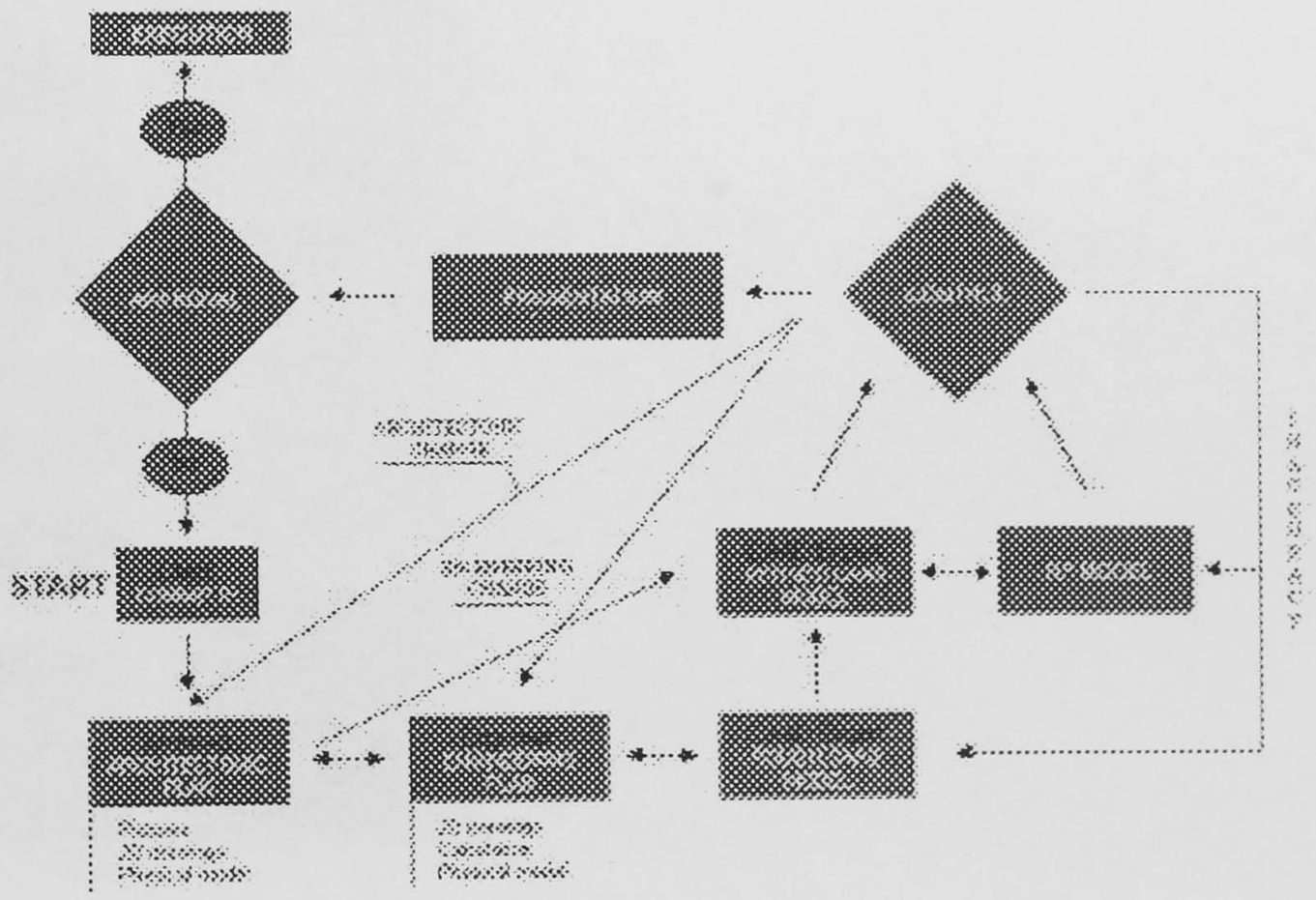


Figure 1. The role of visualisation inside the design process where the graphic ideation (active phase) is represented with the architectonic and engineering planning and the communication (passive phase) is represented by the CAAD/RP section.

fashion, where each module is able to fulfill to a certain task and it is independently coded. This approach has allowed the delivery of an initial functioning core of the JCAD-VR system whose capabilities will be expanded in the near future with several modules currently under development (fig 2).

The entire framework is handling the virtual environment through two closely connected parts: a 3D engine and a services unit each made of different modules. In a human body analogy the former might represent the heart while the latter might be considered as a nervous system.

The 3D engine unit

The 3D engine is the broad part of the framework that handles all the information regarding the "visible" aspects of the virtual world. It includes the code necessary to create and modify geometric entities (**geometry core**), to show the interface (**interface core**) and to deal with several different display devices (**visual core**).

The first module of the **geometry core** handles the creation of 2D and 3D objects: the last ones being both geometric primitives (cones, boxes, spheres etc.) and architectural entities (walls, slabs etc.). Although from the visualisation point of view a wall might be seen as a box, the system treats it in a completely different way. In fact while a box is just regarded as a simple shape without any further quality, the wall instead is handled as an entity owning "topological" properties: it is first of all made of two different surfaces (internal and external faces) and a core. It can be the parent of another object (such as a window or a door) and it can hold other types of information such as number of windows attached to it.

Quite obviously the *geometry module* will also provide the means for attaching materials to objects and add lights and objects from a library to the virtual world through the *database module*. Further development will implement real-time shape recognition routines. This would allow the user the freedom of drawing shapes in the virtual world that

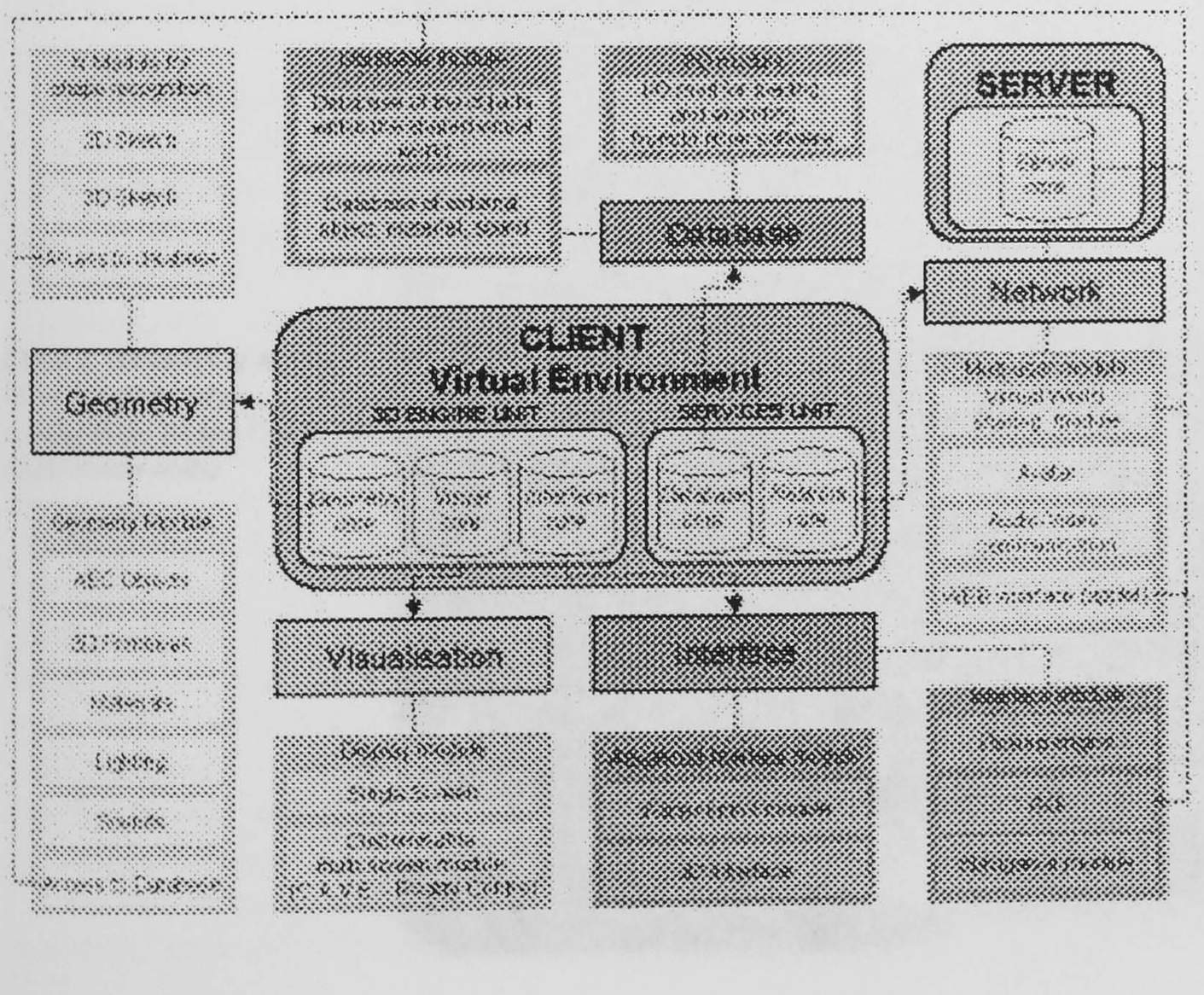


Figure 2. The general overview of the different modules

Go to contents 16

would be converted in 3D objects by the system. The complexity of such a process requires some Artificial Intelligence (AI) routines and it has convinced the authors to code the AI module only in a second stage of development.

The **interface core**, as the name itself says , obviously takes care of the graphic user interface (GUI). As previously mentioned the system aims to achieve an interface as "transparent" as possible. Here the term "transparency" is to be considered as the interface which is not a separate part of the 3D world but as an integral part of the virtual world itself. The idea behind it is that instead of the traditional menus and toolbars the user is immersed in an environment providing the means for the interaction itself. 3D menus pop up showing 3D icons and the 3D menus themselves can be moved for the convenience of the user. Visual feedback is provided, for example, in the form of rulers showing the size of objects or 3D icons showing the operation to be done on the object. A voice driven interface is due to be coded to help push further the level of "transparency" and enhance user friendliness of the interface. For the sake of completeness a traditional window based control panel is provided for advanced settings. On the first functioning core traditional pointing devices are used but support for 6-degree of freedom/virtual glove will be coded in the next future.

The **visual core** is the part of the framework that allows interfacing with the visualization devices. For the sake of flexibility the entire framework has been coded in a multi-platform language (Java). The obvious computational constrains imposed by the use of different hardware is solved by creating a structure that is flexibly scalable and can deliver images for a range of viewing devices, from the simple desktop monitor to the more complex tessellated screen [2] for immersive environments. The user can switch between two different modes according to the machine it is running on. When the software is loaded the user is asked to choose whether to work on a single screen or in multiple screen mode. At the present stage, on common PCs the video card displays the virtual world on a traditional window or on full screen. The system also runs on a Sgi supercomputer whose display is a Reality Centre. The internal architecture of this module is entirely flexible such that it might be easily adapted to allow use of other VR devices (C.A.V.E, H.M.D. etc.) (fig 3).

The services unit

The services unit is the part of the framework that handles all the information regarding the "management" of the virtual world. It is the backbone of the interconnection between users: it manages network connections and exchange of data between users (**network core**), it keeps tracks of the state of

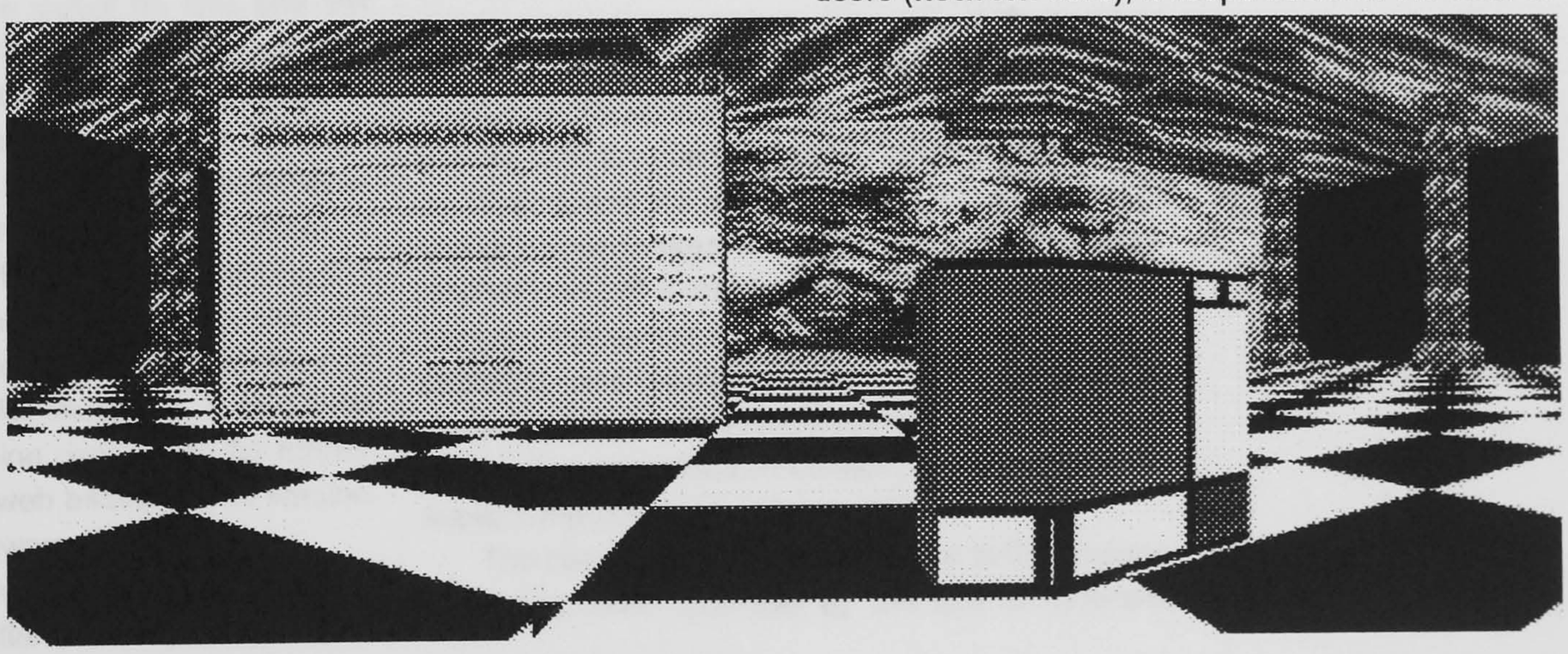


Figure 3. An image of JCAD-VR as it appears when running in the Reality Centre (multi-screen mode)

Go to contents 16

the virtual world in a database from which it also retrieves information from libraries of objects (**database core**).

As already mentioned the entire framework is based on a client/server architecture.

The services unit is indeed developed across two independently coded packages of the framework - *client* and *server*- and the **network core** is allowing the transmission of data between the two (fig 4).

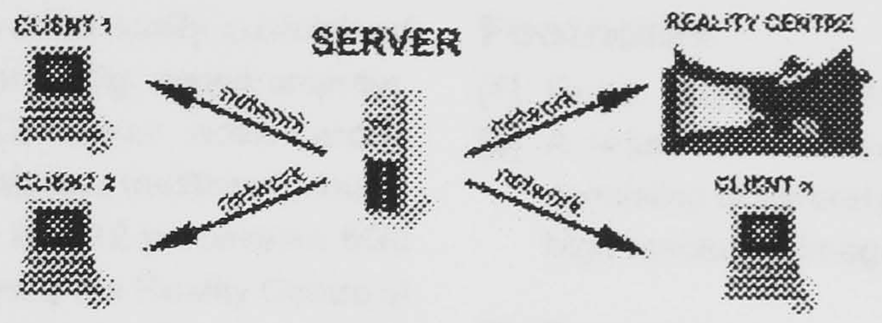


Figure 4. The client/server architecture of JCAD-VR where the server broadcasts to several clients including the Reality Centre.

The **network core** is thus based on a server, several clients and the network allowing the communication. The server is the data-delivering unit that looks after the information to be broadcast. The clients are the users themselves who perform actions and queries, when active, and when passive, rely on the server for receiving data update. The intrinsic multiplatform nature of JCAD-VR, inherited from the language used to code it, allows the server to transmit data to a broad range of machines, from normal PCs to the supercomputer running the Reality Centre, and leaves the research team the freedom to test the software with several operating systems. The communication channel ensures the link between server and clients through a TCP/IP network. As an independent part of the framework the server has an autonomous and simpler interface that provides primarily information about the network system. A number of components are envisaged such as the communication status, the users on line and VR shared environments. Since the clients are communicating through independent processes in the future a further enhancement will allow the server to be capable of dealing with several VR environments simultaneously. At the present stage the network module is supporting use of avatars representing the users inside the virtual world, and it gives also the possibility to interact between the users through a chat system and a whiteboard for sketching. Support for voice communication is being considered for further development as well as a web based applet version of the client side of the software.

The **database core** includes an internal database, that keeps track of the creation or manipulation of the objects in the virtual scene, and an external database through which users will be able to retrieve geometric primitives, materials, lights etc. The internal database is closely coupled with the **network core**. It is not only keeping track of what is happening in the user's virtual world but, most importantly, it receives, through the network, information sent by other users' internal databases. If a new object is created or its status is changed the system will upgrade the internal database of each user no matter who is doing the action. For the convenience of the user an I/O module will allow import/export of objects from/to other packages.

Technical overview

The entire system is coded in Java'. The choice, even if less efficient in term of performances if compared with some other languages, offered indeed great flexibility, true scalability and last but not least fully multi-platform support. Moreover the use of Java' programming language became a natural choice when its 3D suite was released (Java3D'). Its network-centric nature, its multimedia integration together with the use of native hardware acceleration (OpenGL) and multi-processors support (in the case of Sgi workstation) makes it the perfect choice for the development of a real-time multimedia collaborative system. Furthermore thanks to Java"s performance scalability and hardware independence the concept of CAAD has been pushed even further creating a VR environment that can co-exist between high-end supercomputers and common PCs.

The client application, in response to the obvious hardware limits imposed by the use of different

hardware, has been written to be easily customised to run on PCs as well as on a Sgi supercomputer. The former are normal PCs whose video-card is displaying the virtual world only on a traditional window or at full screen. The latter is a 12-processors 6Gb Ram Sgi Onyx2 system running the Reality Centre at ABACUS, University of Strathclyde, Glasgow. When the JCAD-VR is launched on the Sgi it can take advantage of its computational power to stretch itself on a 5 metre wide 2 metre high tassellated screen where 3 Barco projectors create a 160 degree panoramic image.

Conclusions and further developments

The multidisciplinarity of this research is giving the opportunity to investigate collaborative design issues, the role of interfaces inside CAAD packages, the design process in the first stage of its conception, the use of Virtual Reality in architecture and last but non least, a number of technical issues.

As already pointed out, JCAD-VR is an ongoing project and several enhancements are planned for the next releases aiming to get that feeling of intuitiveness and that control over the design that should be the goal of every application using Virtual Reality.

Footnotes

[1] Dorta, T. and LaLande, P., 1998, p.144-148.

[2] A tessellated screen is a composite screen consisting of several projectors creating an unique high resolution image.

References

Ucelli, G., Conti, G., Lindsay, M. and Ryder, G.: 2000, from "Soft" to "Hard" Prototyping: A Unique Combination of VR and RP for Design, in proceedings from UkVRSig 2000, Glasgow.

Bertol, D.: 1997, Designing Digital Spaces, An architect's guide to Virtual Reality, John Wiley & Sons, Inc.

Dorta, T. and LaLande, P.: 1998, The impact of Virtual Reality on the design process, in proceedings from ACADIA Conference, Quebec City.

Hughes, M., Hughes, C., Shoffner, M. and Winslow, M.: 1997, Java Network Programming, Manning, Greenwich.

Barrilleaux, J.: 2000, 3D User Interfaces with Java 3D, a guide to computer-human interaction in three dimensions, Manning, Greenwich.

Sowizral, H. A. and Deering, M. F.: 1999, The Java 3D API and Virtual Reality, IEEE Computer Graphics and Applications, May/June.

Ucelli, G., Conti, G., Petric, J. and Maver, T. (2002). Real Experiences of Virtual Worlds. In D. Marjanovic (Ed.), *Proceedings of the Design 2002, 7th International Design Conference, 2002, Cavtat* (pp. 561-566). Zagreb, Croatia: Faculty of Mechanical Engineering and Naval Architecture, Zagreb and The Design Society, Glasgow, UK.

INTERNATIONAL DESIGN CONFERENCE - DESIGN 2002
Dubrovnik, May 14 - 17, 2002.

REAL EXPERIENCES OF VIRTUAL WORLDS

G. Ucelli, G. Conti, J. Petric and T.W. Maver

Keywords: Virtual Reality, Collaborative Design, Distributed Environment

1. Introduction

The present use of 3D simulations or more effective virtual worlds has provided the designer with new media capable of storing several levels of information traditionally obtained only with the help of multiple media, usually more time and resource-consuming.
Virtual models in particular can store information about planning issues, geometric design, material choices or even furniture and lighting conditions. This level of representation provides the designer with all the necessary tools to represent an architectural environment and facilitate the research of potentially good design solutions.
The use of Virtual Reality (VR) within the design process has not only enabled the designer to store more information than with the use of the traditional media and to check the design solutions more efficiently but furthermore it has enhanced the level of simulation providing:

- **Immersion**: Users are completely surrounded by the environment.
- **Presence**: Being surrounded the participant has actually the sensation of being *in* the environment. The Virtual Environment becomes then a place on its own and its perception is similar to real environments.
- **Interactivity**: This is surely the most important *feature* provided by VR: the environment allows the participant to be involved and the result of the actions done by the participant is visualized in the VE.
- **Autonomy**: Participants are neither constrained in paths nor in views preset by others but have the freedom and autonomy to explore any single part of the environment.
- **Collaboration**: Multiple users are able to take part and to interact in the same VE.

The use of VR can also broaden the boundaries of traditional perception to give the experience of worlds not necessarily real or material and to give the freedom to safely simulate dangerous or expensive condition for training purposes. In fact some applications can simulate something completely different from anything we have ever directly experienced such as the visualisation of the ebb and flow of the world's financial markets or the information of a large corporate database. Other applications provide ways of viewing from an advantageous perspective not possible or too expensive in the real world, like scientific simulators, tele-presense systems and air traffic control systems.
The speed at which technology is evolving is making the application of VR within the design professions a feasible approach. AEC companies have already started to evaluate how time consuming the traditional presentation path can be where animations or walkthroughs are used to show designs solutions to their clients. In fact traditional CAD/CAAD systems are used as rendering tools more than design tools. Any change on design solutions is subject to the inevitable delay of having to step back to the CAD/CAAD systems and then the result must be rendered again to be eventually visualized. This approach is obviously not only unconvenient but time consuming and therefore costly. The consequence of these issues is that some design and manufacturing companies have already started to

investigate how VR can be used within the design process.

2. Background

Since the presentation of The CAVE at the SIGGRAPH conference in 1992 all the media have been presenting VR technologies as the new tools that designers were waiting for and in particular they have been erroneously assumed that it would have triggered the architects' interest for its power to communicate design ideas.
Unfortunately VR is far from being used on a large scale during the architectural design activities although it would provide a natural and easy-to-understand interface between practitioners and clients. Moreover VR would enable architects to test the design functionality and to see whether the design solutions reach the clients expectations; it would increase the possibilities to design a better-built environment by:

- addressing sustainability through environmental simulations and appraisal
- engaging design creativity through immersive design

We have identified two important reasons why VR has not been widely used in the architectural context: the lack of interfaces designed for architects and the wrong positioning of the VR phase inside the design process.
If we observe the software available on the market from a user-centric point of view it is clear that none of the current packages used by architects provides an easy interface to generate virtual environments during the creation phase of the design process. CAAD packages are often complex rendering tools more than design tools and the funtionalities related to VR they provide are subordinated to the creation of 3D models and exportation. Consequently any change on design solutions is subject to the inevitable delay of going back to the CAAD system to refine the 3D model. Thus the recreation of 3D models is so impractical and time-consuming that becomes worth doing only when every design decision has been already taken. In these circumstances the use of VR would just increase costs.
Furthermore the lack of interfaces suitable for architects has as a consequence the wrong positioning of VR technologies in the design process. In fact if the virtual environments created using CAAD packages are generated as refinements, adjustments and exportation from traditional 3D scenes, it is clear that practitioners will consider their use only as presentation tools. Keeping up to date 3D models is an expensive task and obviously even more expensive is to upgrade VEs generated from them.
Therefore we could say that VR is relegated to the end of the design process rather then being used to engage design creativity through immersive design.

Figure 1. Traditional Schema

Having identified these problems behind the difficulty of using VR from the very beginning of the design phase, the research group has thought to engage itself in the development of a VR system that provides a flexible user-friendly immersive environment to support collaborative design on a synchronous co-editing base, being called JCAD-VR.
This paper will report the present state of the JCAD-VR system and will highlight its future development.

3. The JCAD-VR schema

The idea upon which the JCAD-VR framework is founded is to anticipate the use of VR within the creation phase thus taking full advantage of VR technology. The system in fact allows the creation of simple virtual environments through a user-friendly interface without forcing the user to model it with

traditional CAAD packages. The use of CAAD packages is therefore left to the final stage of the project, where further refinements are needed.
It creates simple parametric 3D-shapes directly in a co-edit VR environment, thus allowing the design to be shared as it evolves.

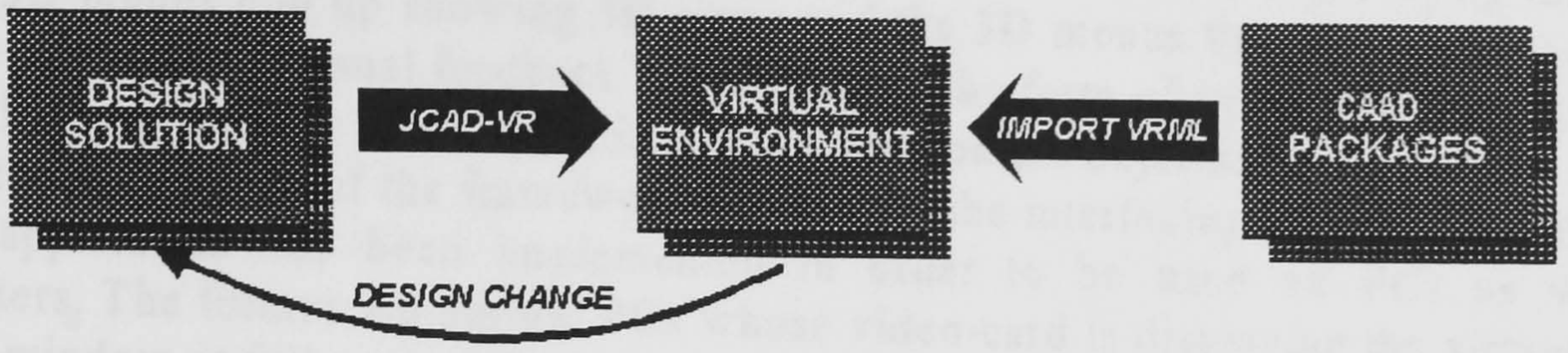


Figure 2. JCAD-VR Schema

To allow constant collaboration between several users the entire project is based on client-server architecture where every user accesses the virtual world, interacts with the VE and shares design tasks. The whole framework is organised in an object-oriented fashion, where each module fulfils a certain task and it is independently coded. This approach has allowed the delivery of an initial functioning core of the system, whose capabilities will be expanded in the near future.

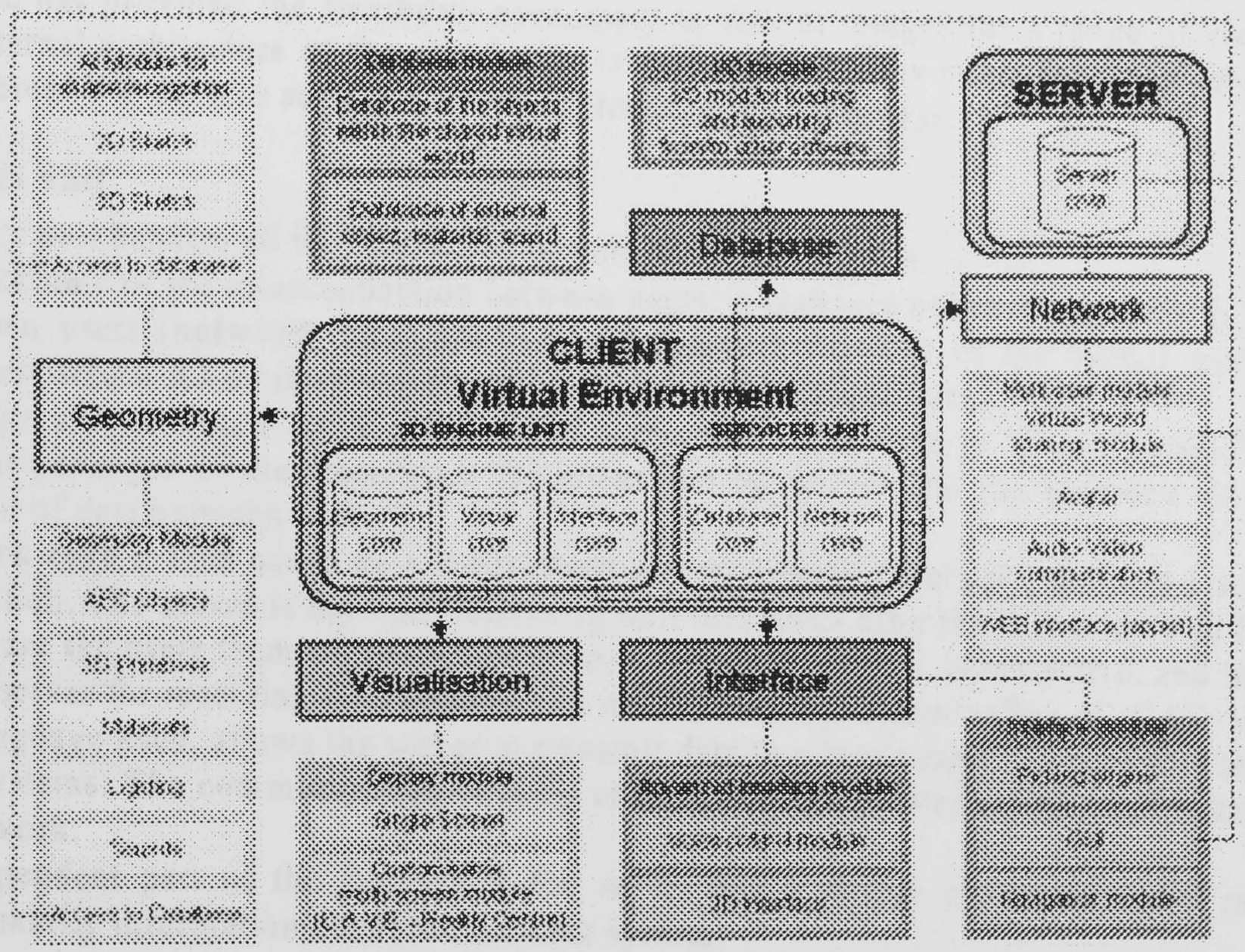


Figure 3. The JCAD-VR framework schema

From the implementation point of view JCAD-VR handles the VE through two closely connected sections: a *3D engine* and a *services unit* each made of several *modules*.

3.1 3D engine unit

The 3D engine handles all the information regarding the visual aspects of the VE. It includes the code necessary to create and modify geometric entities (**geometry core**), to run the 3D-interface (**interface core**) and to deal with several different output devices (**visual core**).
The first module of the **geometry core** handles the creation of 3D objects: both geometric primitives (cones, boxes, spheres etc.) and architectural entities (walls, slabs etc.). To the architectural entities some extra properties were provided such as: information on internal and external faces or windows and doors attached to them.
The *geometry module* will also provide the means for attaching materials to objects and add lights and

objects from a library to the virtual world through the *database module*.
The **interface core** does not implement a traditional graphic user interface (GUI): JCAD-VR has been provided with a 3D interface that is an integral part of the virtual world itself. The idea behind it is that instead of the traditional menus and toolbars the UI is immersed in VE providing the means for the interaction: 3D menus pop up showing 3D icons and the 3D menus themselves can be moved for the convenience of the user. Visual feedback is provided in the form of rulers showing the size of objects or 3D icons helping the user in the operations to be done on the objects.
The **visual core** is the part of the framework that allows the interfacing with the visualization devices. The client application has been implemented in order to be used on PCs as well as on Sgi supercomputers. The former are normal PCs whose video-card is displaying the virtual world only on a traditional window at full screen, the latter is a 12-processors 6Gb Ram Sgi Onyx2 system running a Reality Centre. When JCAD-VR is launched on the system running the Reality Centre it can take advantage of the increased computational power stretching its visual output on a 5 metre wide 2 metre high tassellated screen where 3 projectors create a 160 degree panoramic image.
For the sake of flexibility the entire system is coded in Java™. The choice, even if less efficient in terms of performances if compared with some other languages, offered indeed great flexibility, true scalability and last but not least fully multi-platform support. Moreover the use of Java™ programming language became a natural choice when its 3D suite was released (Java3D™).
This choice has provided the flexibility neceessary to deliver images for a range of viewing devices and the internal architecture of the visual core is such that modules might be easily adapted to allow use of different VR devices such as CAVEs or Headmounted Displays.

3.2 Services unit

The *services unit* handles all the circulation of data within the system.
It is the backbone of the interconnection between users: it manages network connections, it exchanges data between users (**network core**) and it keeps track of the state of the virtual world through a database from which it also retrieves objects information (**database core**).
The *services unit* is based on a client/server architecture therefore it is implemented across two independent packages of the framework the *client* and the *server* and the **network core** allows the transmission of data between them.
The **network core** is thus based on a multi-client server, several clients and the network allowing the communication. The server is the data-delivering unit that looks after the information to be broadcast. The clients are the users themselves who perform actions and queries, when active, and when passive, rely on the server for receiving data update. The intrinsic multiplatform nature of JCAD-VR, inherited from the language used, allows the server to transmit data to a broad range of machines across several operating systems. The communication channel ensures the link between server and clients through a TCP/IP network.
As an independent part of the framework the server has a simple and autonomous interface that provides primarily information about the network system.
At the present stage the *network module* supports:

- Broadcasting of new geometries in the VE
- Notification of creation of new geometries in every user's internal database and broadcasting of their numerical information
- Broadcasting of modifications applied on geometries in the VR scene
- Notification of changes on geometries in every user internal database
- Checking for user priority on the objects through a distributed locking mechanism
- *Avatars* representing multiple clients in the VE
- Interaction between users through a chat system and a whiteboard for freehand sketching in 2D.

It will be soon expanded to include new functionalities such as the transfer of voice and video accross users.

As already pointed out, JCAD-VR is an ongoing project and several enhancements are planned for the next releases aiming to get that feeling of intuitiveness and that control over the design that should be the goal of every application using Virtual Reality.

References

Barrilleaux, J., "3D User Interfaces with Java 3D-A guide to computer-human interaction in three dimensions", Manning, Greenwich, 2000.

Bertol, D., "Designing Digital Spaces", John Wiley & Sons, 1997.

Conti, G., Ucelli, G. and Maver, T., "JCAD-VR - A Collaborative Design Tool in Virtual Reality", Proceedings of ECAADE 19. Helsinki, 2001, pp 454 – 459.

Dorta, T. and LaLande, P., "The Impact of VR on the Design Process", Proceedings of ACADIA '98, Quebec, 1998, pp 139 – 161.

Hughes, M., Hughes, C., Shoffner, M. and Winslow, M., " Java Network Programming", Manning, Greenwich, 1997.

Maver, T., Harrison, C. and Grant, M., "Virtual Environments for Special needs", Proceedings of CAAD Futures 2001. Eindhoven, 2001, pp 151 – 159.

Petric, J., Ucelli, G. and Conti, G., "Educating the Virtual Architect", Proceedings of ECAADE 19. Helsinki, 2001, pp 388 – 393.

G. Ucelli
ABACUS Unit,
Architecture and Building Science,
University of Strathclyde,
131 Rottenrow
G4 0NG Glasgow, UK
Tel. +44 141 548 3997
Fax. +44 141 552 3996
Email: giuliana.ucelli@strath.ac.uk

Petric, J., Maver, T., Conti, G. and Ucelli, G. (2002). Virtual Reality in the Service of User Participation in Architecture. In K. Agger, P. Christiansson and R. Howard (Eds.), ***Distributing Knowledge in Building. Proceedings of CIB W78 Conference, Aarhus*** **(pp. 217-224). Aarhus, Denmark: Aarhus School of Architecture and Centre for Integrated Design.**

Theme: Arial, size 10 (will be completed by the organising committee)

Title: **Virtual Reality in the Service of User Participation**

Author(s): J Petric, T Maver, G Conti, G Ucelli
Institution(s): University of Strathclyde
E-mail(s): j.petric@strath.ac.uk
t.w.maver@strath.ac.uk
giuseppe.conti@strath.ac.uk
gucelli@strath.ac.uk

Abstract:

The issue of user participation in the processes of building and urban design is enjoying renewed attention following its relative neglect over the last 20 years due, in large measure, to significant advances in emerging information technologies, particularly multimedia, virtual reality and internet technologies.

This paper re-established the theoretical framework for participatory design evolved in the late sixties and early seventies as part of the movement towards a more explicit design methodology and attempts an explanation of why the concept failed to gain commitment from the architectural and urban design professionals.

The paper then gives an account of two significant developments in the evolution of the application of information technologies with which the authors have been engaged. These are:

i. *a responsive and interactive interface to wholly immersive and realistic virtual reality representations of proposed buildings and urban neighbourhoods.*
ii. *an intuitive and platform-independent VR modelling environment allowing collaborative evolution of the scheme from within the virtual world.*

The impact of these IT developments is demonstrated in the context of the design of a leisure facility for a community of users with physical impairment.

Keywords: *user, design, participation, VR, CAAD*

Design Decision Making

Architectural design is a multi-faceted occupation which requires, for its successful performance, a mixture of intuition, craft skills and detailed knowledge of a wide range of practical and theoretical matters. It is a cyclical process in which groups of people work towards a somewhat ill-defined goal in a series of successive approximations. There is no 'correct' method of designing and, although it is recognised that the process can be divided into separate phases, there is no generally accepted sequence of work that might guide design teams in the direction of achieving a satisfactory solution. Indeed, there are no solutions to design problems in the way that there are solutions to mathematical problems: the best that can be hoped for is an outcome which satisfies the maximum number of constraints which bound the area of concern. Furthermore, design is not an algorithmic process in which the desired conclusion can be reached by the application of step-by-step procedures - first finalising this aspect, then that. It is a fluid, holistic process wherein at any stage all the major parts have to be manipulated at once. In this sense, it is less like solving a logical puzzle and more like riding a bicycle, blindfold, whilst juggling.

Despite the complexity of the design decision-making process the emerging new generation of computer-based models is already having an impact on how design is performed and, hence, on the quality of design. The impact stems from the fact that the new models, as opposed to paper-based plans and elevations or other conventional forms, are predictive rather than descriptive; dynamic rather than static; explicit rather than implicit and, above all, permit a more-or-less continuous and interactive assessment of the effects of a developing design on cost and performance.

Evidence is growing of the advantages offered by the application of computers in design, and these can be summarised as follows:

Widening the Search for Solutions

Access to programs which dynamically predict the cost and performances characteristics of optional design proposals can increase the scope of search for good solutions by as much as ten-fold. Not only is the search coverage extended, it is also more purposefully directed because designers are able to compare the quality of any one tentative solution against the quality of all previous solutions.

Greater Integration in Decision-Making

In conventional working, a great deal of design time is lost as proposals are passed to and fro between the architect (who tends to be the originator) and the other specialist members of the design team (who tend to the "checkers"). Quite frequently the scheme on which the architect has lavished time and effort is found by one or other of the specialists to be infeasible. With access to appropriate appraisal techniques embodied in computer programs, it is possible to check a proposal against a wide range of criteria from the outset of the design activity. Moreover, it is entirely practical (though not yet a widespread working method) for all members of the design team to have access to, and operate on, the common design model whether or not they share a design office. The models, then, can provide a strong integrating force in design team working.

Improving Design Insights

Apart from the use of appraisal programs to search for better designs, the programs can be used in a research and development context to provide insights into the way in which particular design decisions affect cost and performance. Typically, a designer working in this mode would select an existing building for study, then, keeping all other design variables constant (insofar as this is possible), systematically vary one factor while recording the cost/performance output from the program. In this manner, the architect can establish sets of causal relationships which provide powerful insights into structure of design decision-making.

Differentiation of Objective and Subjective Judgements

Contrary to the early fears of many architectural practitioners, the use of CAAD techniques focuses increased attention on subjective value judgements rather than less. As measurable attributes of optional designs are made more explicit, the necessary value judgements are forced to the surface of design activity and thereby, themselves become more explicit. The effect of this is to make it clear to designers and their clients, which judgements are based on quantifiable criteria and which on subjective and intuitive concepts.

Evidence of the degree to which computer-generated cost/performance information promotes effective value judgement, throws into sharp focus the crucial question: **whose** value judgement? This question was, for the first time, seriously addressed in the Design Participation Conference in Manchester in 1971 (1). At that time, however, the human-machine interface was too primitive for the concept of useful participation by the users of buildings to be achieved. The new technologies of VR and Multimedia give real prospects for participation.

Virtual Reality

The present use of 3D simulations or more effective virtual worlds has provided the designer and user participants with new media capable of storing several levels of information traditionally obtained only with the help of multiple media, usually more time and resource-consuming.
Virtual models in particular can store information about planning issues, geometric design, material choices or even furniture and lighting conditions. This level of representation provides the designer with all the necessary tools to represent an architectural environment and facilitate the research of potentially good design solutions.
The use of Virtual Reality (VR) within the design process has not only enabled the designer to store more information than with the use of the traditional media and to check the design solutions more efficiently but furthermore it has enhanced the level of simulation providing:

?? **Immersion**: Users are completely surrounded by the environment.
?? **Presence**: Being surrounded the participant has actually the sensation of being *in* the environment. The Virtual Environment becomes then a place on its own and its perception is similar to real environments.
?? **Interactivity**: This is surely the most important *feature* provided by VR: the environment allows the participant to be involved and the result of the actions done by the participant is visualized in the VE.
?? **Autonomy**: Participants are neither constrained in paths nor in views preset by others but have the freedom and autonomy to explore any single part of the environment.
?? **Collaboration**: Multiple users are able to take part and to interact in the same VE.

The use of VR can also broaden the boundaries of traditional perception to give the experience of worlds not necessarily real or material and to give the freedom to safely simulate dangerous or expensive condition for training purposes. In fact some applications can simulate something completely different from anything we have ever directly experienced such as the visualisation of the ebb and flow of the world's financial markets or the information of a large corporate database. Other applications provide ways of viewing from an advantageous perspective not possible or too expensive in the real world, like scientific simulators, tele-presence systems and air traffic control systems.

The speed at which technology is evolving is making the application of VR within the design professions a feasible approach. AEC companies have already started to evaluate how time consuming the traditional presentation path can be where animations or walkthroughs are used to show designs solutions to their clients. In fact traditional CAD/CAAD systems are used as rendering tools more than design tools. Any change on design solutions is subject to the inevitable delay of having to step back to the CAD/CAAD systems and then the result must be rendered again to be eventually visualized. This approach is obviously not only inconvenient but time consuming and therefore costly. The consequence of these issues is that some design and manufacturing companies have already started to investigate how VR can be used within the design process.

The JCAD-VR Prototype

In the Department of Architecture and Building Science at the University of Strathclyde, the ABACUS group has been building a prototype design decision support system known as JCAD-VR (2).

The idea upon which the JCAD-VR framework is founded is to anticipate the use of VR within the creation phase thus taking full advantage of VR technology. The system in fact allows the creation of simple virtual environments through a user-friendly interface without forcing the user to model it with traditional CAAD packages. The use of CAAD packages is therefore left to the final stage of the project, where further refinements are needed.
It creates simple parametric 3D-shapes directly in a co-edit VR environment, thus allowing the design to be shared as it evolves.

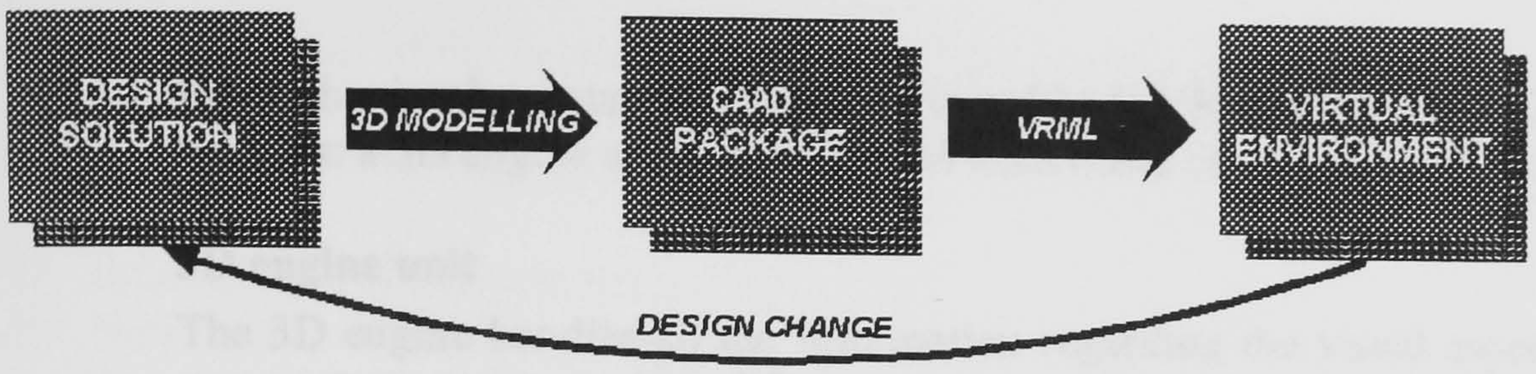


Figure 1. Traditional Schema.

Figure 2. JCAD-VR Schema.

To allow constant collaboration between several users the entire project is based on client-server architecture where every user accesses the virtual world, interacts with the VE and shares design tasks. The whole framework is organised in an object-oriented fashion, where each module fulfils a certain task and it is independently coded. This approach has allowed the delivery of an initial functioning core of the system, whose capabilities will be expanded in the near future.

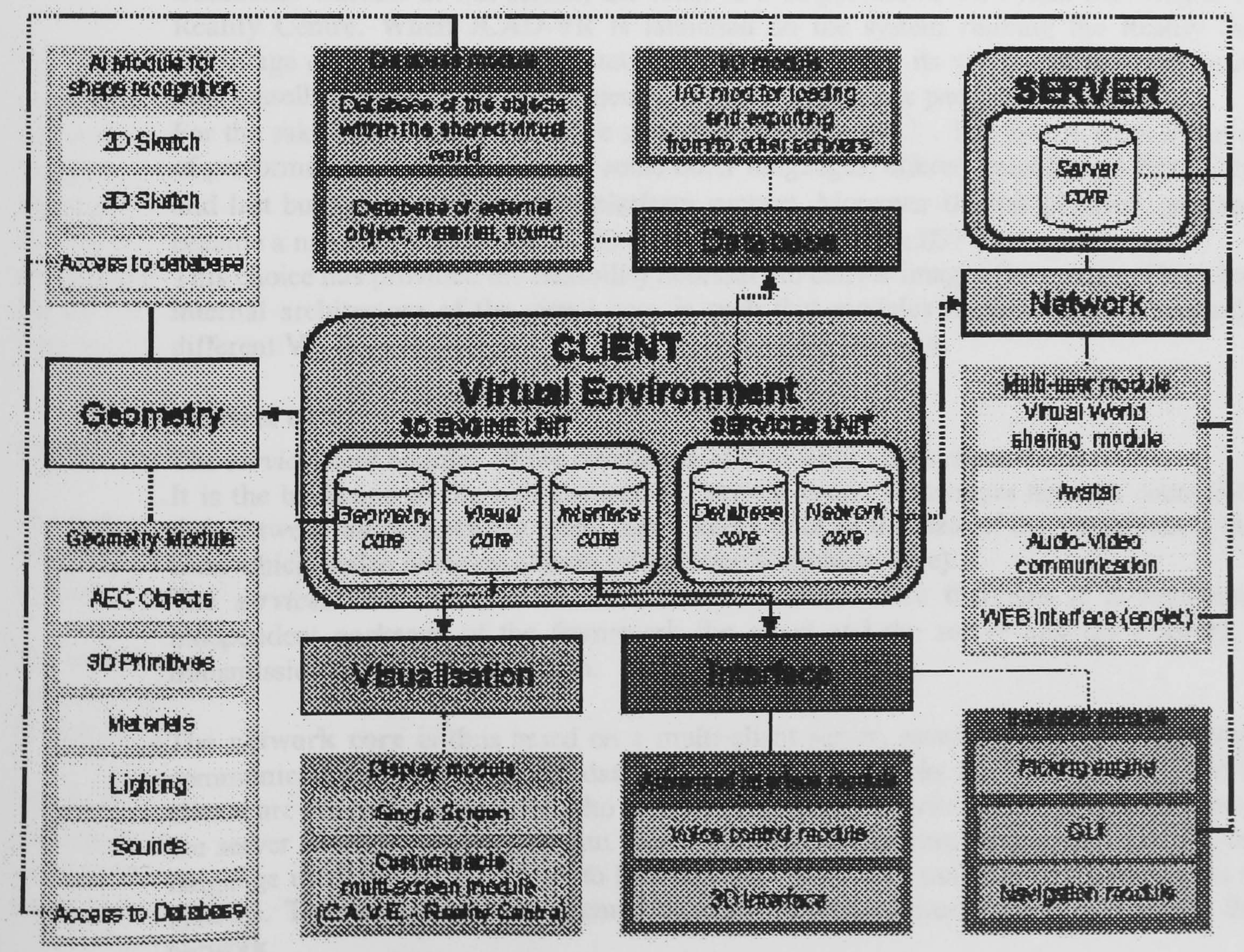


Figure 3. The JCAD-VR framework schema

From the implementation point of view JCAD-VR handles the VE through two closely connected sections: a *3D engine* and a *services unit* each made of several *modules.*

3D engine unit

The 3D engine handles all the information regarding the visual aspects of the VE. It includes the code necessary to create and modify geometric entities (**geometry core**), to run the 3D-interface (**interface core**) and to deal with several different output devices (**visual core**).

The first module of the **geometry core** handles the creation of 3D objects: both geometric primitives (cones, boxes, spheres etc.) and architectural entities (walls, slabs etc.). To the architectural entities some extra properties were provided such as: information on internal and external faces or windows and doors attached to them.
The *geometry module* will also provide the means for attaching materials to objects and add lights and objects from a library to the virtual world through the *database module.*

The **interface core** does not implement a traditional graphic user interface (GUI): JCAD-VR has been provided with a 3D interface that is an integral part of the virtual world itself. The idea behind it is that instead of the traditional menus and toolbars the UI is immersed in VE providing the means for the interaction: 3D menus pop up showing 3D icons and the 3D menus themselves can be moved for the convenience of the user. Visual feedback is provided in the form of rulers showing the size of objects or 3D icons helping the user in the operations to be done on the objects.

The **visual core** is the part of the framework that allows the interfacing with the visualization devices. The client application has been implemented in order to be used on PCs as well as on SGI supercomputers. The former are normal PCs whose video-card is displaying the virtual world only on a traditional window at full screen, the latter is a 12-processors 6Gb Ram SGI Onyx2 system running a Reality Centre. When JCAD-VR is launched on the system running the Reality Centre it can take advantage of the increased computational power stretching its visual output on a 5 metre wide 2 metre high tassellated screen where 3 projectors create a 160 degree panoramic image.
For the sake of flexibility the entire system is coded in Java? . The choice, even if less efficient in terms of performances if compared with some other languages, offered indeed great flexibility, true scalability and last but not least fully multi-platform support. Moreover the use of Java? programming language became a natural choice when its 3D suite was released (Java3D?).
This choice has provided the flexibility necessary to deliver images for a range of viewing devices and the internal architecture of the visual core is such that modules might be easily adapted to allow use of different VR devices such as CAVEs or Headmounted Displays.

Services unit

The *services unit* handles all the circulation of data within the system.
It is the backbone of the interconnection between users: it manages network connections, it exchanges data between users (**network core**) and it keeps track of the state of the virtual world through a database from which it also retrieves objects information (**database core**).
The *services unit* is based on a client/server architecture therefore it is implemented across two independent packages of the framework the *client* and the *server* and the **network core** allows the transmission of data between them.

The **network core** is thus based on a multi-client server, several clients and the network allowing the communication. The server is the data-delivering unit that looks after the information to be broadcast. The clients are the users themselves who perform actions and queries, when active, and when passive, rely on the server for receiving data update. The intrinsic multiplatform nature of JCAD-VR, inherited from the language used, allows the server to transmit data to a broad range of machines across several operating systems. The communication channel ensures the link between server and clients through a TCP/IP network.

As an independent part of the framework the server has a simple and autonomous interface that provides primarily information about the network system.
At the present stage the *network module* supports:

- ?? Broadcasting of new geometries in the VE
- ?? Notification of creation of new geometries in every user's internal database and broadcasting of their numerical information
- ?? Broadcasting of modifications applied on geometries in the VR scene
- ?? Notification of changes on geometries in every user internal database
- ?? Checking for user priority on the objects through a distributed locking mechanism
- ?? *Avatars* representing multiple clients in the VE
- ?? Interaction between users through a chat system and a whiteboard for freehand sketching in 2D.

It will be soon expanded to include new functionalities such as the transfer of voice and video across users.

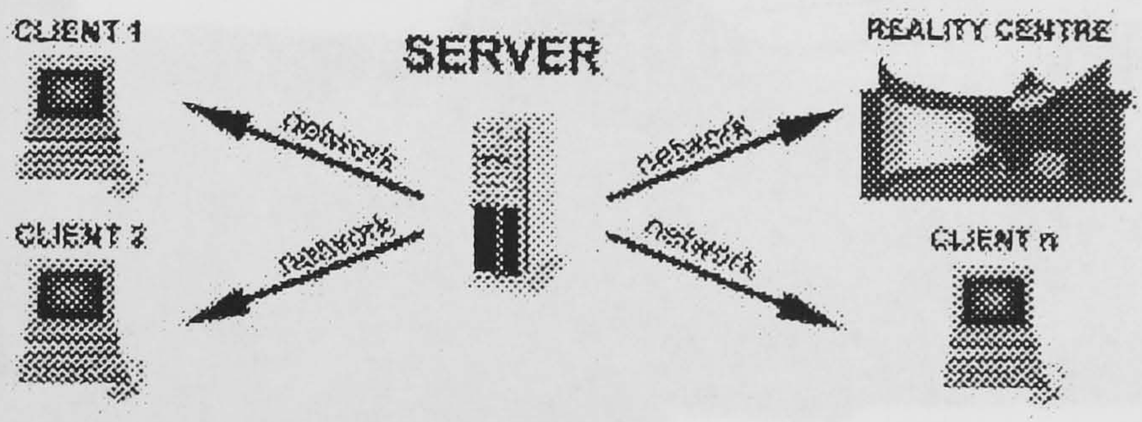


Figure 4. The client/server architecture of JCAD-VR where the server broadcasts to several clients including the Reality Centre

The **database core** includes an internal database, that keeps track of the numerical parameters of the geometries created or modified within the virtual scene, and an external database through which users will be able to retrieve more complex 3D shapes, AEC objects, materials, lights etc.
The internal database is closely coupled with the **network core**. Not only it keeps track of what it is happening in the user's virtual world but also, most importantly, it receives, through the network, information sent by other users' internal databases. If a new object is created or its geometric parameters are changed the system will upgrade the internal database of each user no matter who is doing the action.
For the convenience of the user an *I/O module* supports loading of external files thus allowing import from traditional CAAD packages.

Initial Trials

The current prototype version of JCAD-VR is starting trials of its simultaneous use at the Technical University of Eindhoven and the University of Strathclyde. However, an earlier version was piloted with a group of students in the BSc (Architectural Studies) at the University of Strathclyde.

The brief for the design project associated with the workshop was quite demanding: A Sailing Club for the Disabled located on a canal site in Glasgow City Centre. A real client community agreed to be involved in the project and in the assessment of its outcomes (3).

Within the overall JCAD-VR system, students were encouraged to use initially a standard CAAD package for initial creation of the geometry and then to refine the design using VRML. Event management in their virtual worlds was done through the user of sensors and connectivity. These include touch sensors, proximity sensors, time sensors and anchors.

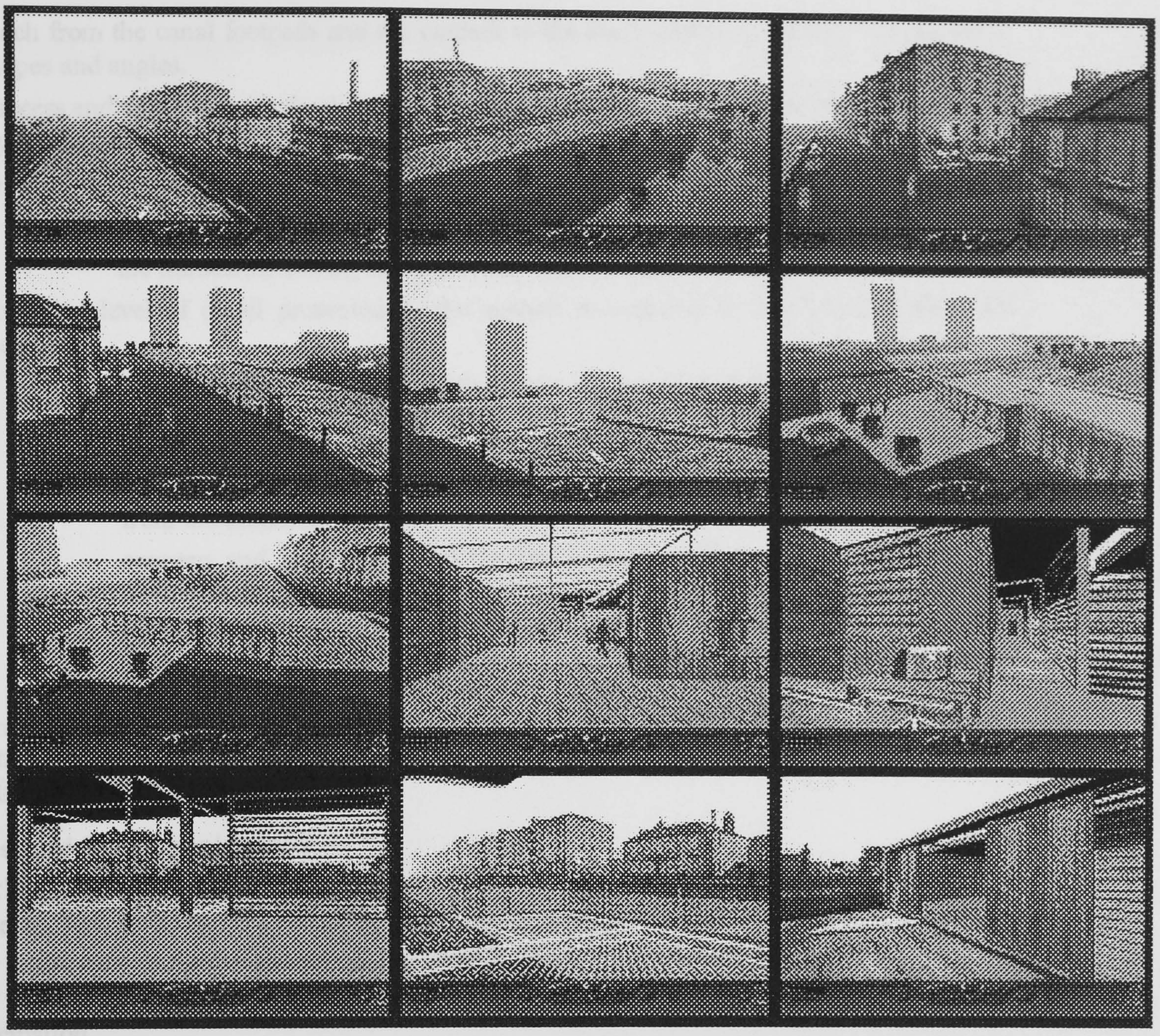


Figure 5. Sailing Club for the Disabled at Spier's Wharf, Glasgow.

The design outcomes

The outcome of the experiment – although not statistically measurable – was nonetheless considered to be remarkable by both the tutors and the client community. One second year student in particular made the most effective use of the full range of functionality of the system.

In relation to the site and exterior of the building these were:

?? Good balance between modelling of 3D geometry and texture mapping to provide a thoroughly convincing, large scale model of the site, which includes existing buildings of importance to the intervention as well as the wider urban issues such as the adjacent motorway.

?? Ability to approach the site, as would a user, by sailing along the canal or, as a wheel chair user, to open gates and wheel along foot-paths to the building entrance.

?? Understanding of the urban site by "flying" nearer or further from the adjacent motorway (along which cars are speeding) to check the attenuation of noise pollution.

In relation to the building itself, the contribution of the student's ability to "visit" his design, during the evolution, as would a wheelchair user, was clearly evident in the quality of the design solution and was manifested in a number of subtle but important ways, for example:

?? The approach from the canal footpath and the carpark to the front door was carefully considered in terms of slopes and angles.

?? The door access and view lines of wheel chair users, on entering the facility (including signage) were completely thought through.

?? The elegance of articulation of the building into two zones - wet and dry - was the evident result of:

- the Boolean operations performed by the student on the volumes;
- the immediate testing of these in the virtual environment.

?? The unparalleled level of detail presented by the student in response to user requirements, was exemplified by :

- the transparency of the balustrades on the ramps and the sophisticated louvre system on the external glazing - a direct result of the designer's perception, from the user viewpoint - of what is important to someone in a wheelchair. Through every window in the building the student was able to show the appropriate view from the building.
- concern and commitment to the real experience for a wheelchair user. The sliding door towards the canal could be opened and the user wheeled out into a mesh deck in order to get a first-hand experience of being "on" the water before perhaps, two ramps down, actually getting into a canoe, the mechanism illustrated by an elegant animation.

The client community group were presented with the work of the students, and were impressed by the sensitivity with which the brief was addressed. They are featuring the outcome on their website (http://www.fcccp.org.uk).

Conclusions and future aspirations

As in the earliest days of the introduction of computers into architectural design, the quantum jump is made by students. The work reported here, and which will be shown during the conference is, we believe, the epitome - in the current state of the art - of excellent practice. It makes a breakthrough, we believe, in the evolution of good design ideas, modelled offline but appraised interactively and offers a real prospect for user participation.

There is some way to go, of course, to *design* interactively in a virtual environment. The next step which we envisage is to link to the 3D model the emerging and sophisticated software for the thermal, lighting and acoustics properties of the building. This would allow the user to visualise, dynamically, airflow, temperature gradients, lighting levels and to experience the actual acoustic characteristics of the space as she/he moves through it.

The other exciting development is for representatives of the client/user group to "join" the designer within the virtual environment and to participate directly in the evolution of the design concept. The recent development of a wheel-chair motion platform for immersive virtual environments (4) will allow the future users of buildings like the Sailing Club for the Disabled to navigate themselves, in their own wheelchair, through the virtual building.

References

1. Nigel Cross, Design Participation. Proceedings of the Design Research Society, Academy Editions (1971)
2. Guiliana Ucelli et al Real Experiences of Virtual Worlds. Design 2002, International Design Conference, Dubrovnic (2002)
3. Jelena Petric et al Educating the Virtual Architect. Promise and Reality. Proceedings of ECAADE Conference, Weimar [Ed Donath, D]
4. Tom Maver et al Virtual Environments for Special Needs. Proceedings of CAAD Futures 2001, (2001)

Petric, J., Ucelli, G. and Conti, G (2002). Real Teaching and Learning through Virtual Reality. In K. Koszewski and S. Wrona (Eds.), *Design e-ducation. Proceedings of the 20th Conference of eCAADe, Warsaw, 2002* (pp. 72-79). Warsaw, Poland: Drukarnia Braci Ostrowskich.

Real Teaching and Learning through Virtual Reality

Jelena Petric, Giuliana Ucelli, Giuseppe Conti
ABACUS, University of Strathclyde, Glasgow, UK
http://www.strath.ac.uk/Departments/Architecture

Abstract. This paper addresses an articulated vision of Virtual Reality which lends itself to design collaboration in teaching and learning and communication of architectural design ideas among students, design professionals and client body during the early stages of the design process.

Virtual Reality (VR) has already acquired a new degree of complexity through development of network-based virtual communities and the use of avatars. The intrinsic quality of VR technology is to support collaborative design experience.

The design tools developed for this experiment are capable of creating 3D objects in a shared VR environment, thus allowing the design and its evolution to be shared. The choice of programming language (Java™) reflects the desire to achieve scalability and hardware independence, which in turn allows for creation of a VR environment that can co-exist between high-end supercomputers and standard PCs. The prototype design environment was tested using PC workstations and an SGI system running a Reality Centre.

The research and teaching/learning experience in the collaborative design environment reported in this paper describe the development and application of software that aims to increase the opportunity for architects to collaborate within virtual worlds which enable effective and transparent information exchange.

Keywords
VR, Collaborative Design, Virtual Environment, Interface, Architecture, Experiment, Design Process.

Introduction

In the recent decades the design profession has been deeply affected by the *digital revolution* and the use of Computer Aided Architectural Design (CAAD) tools is nowadays part of the daily practice in most architectural firms. But in the last few years the 'CAAD community' is experiencing a new revolution that is leading the move from static representation, based on 2D renderings or pre-recorded animations (considered as a sequence of 2D images), to dynamically generated 3D representations. Real-time navigation and interaction, typical of VR environments, provide just that fluent interface enabling the exploration of the design proposals that architects have not been able to get with any other media.

The increasing growth of computational resources and hardware power is probably preparing the anticipated transition to desktop VR applications, making them truly feasible tools for everyday practice. Furthermore, the recent growth of network-based virtual communities has brought a new level of complexity to the notion of virtual spaces, turning the profession of architect into something that might now resembles the one of the *virtual architect*.

Although VR is nowadays a quite mature technology, it is seldom used in architecture throughout the design process, but more often it is merely used as a powerful presentation technique. Design methodologists in the past agreed on the need for iterative cycles between several phases of the design process. From studies concerning designers' behaviour [6] many authors observed an indefinite number of return loops from the moment when gathering of information and structuring of the design problem take place (known as *analysis*) to the one when design solutions are generated (known as *synthesis*).

The use of Virtual Reality within the design process provides the designer with an appropriate, quick and practical feedback which facilitates search for design solutions. Moreover, it enables the capture of more information than would be possible to capture with the use of the traditional media and makes the checking of the design solutions more efficient by enhancing simulation capabilities. The use of VR in design broadens the boundaries of traditional perception by

providing experiences of worlds not necessarily real or material. In short VR is the perfect simulation medium for architects investigating design solutions.

It is then highly predictable that in the near future VR will become the interface for the next generation of computer aided design (CAD) applications and we can anticipate the change of its use from a mere presentation medium to a more powerful and effective design tool.

At present Virtual Environments (VE) are often created using CAAD packages for refinements, adjustments and exportation based on traditional 3D scenes. Documenting the evolution and development of design by constant updating of 3D models is an expensive business, and obviously even more expensive is the upgrading of the VEs generated from these models. In most existing design scenarios the decision making process does not take advantage of the technology, but relegates the use of VR to the end of the process as a more convincing tool to impress contractors and clients. In such a scenario, only once the final solution has been achieved is it worth investing time into more powerful visualisation media.

With this background knowledge the research task we set ourselves was to develop the VR system which would help designers in the initial stages of the design process to take advantage of the VR as a new design tool.

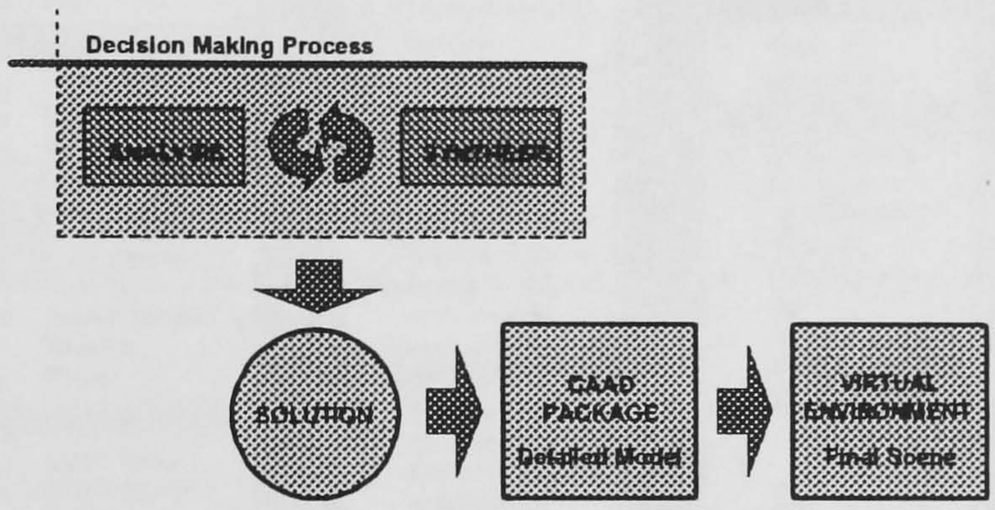


Figure 1. The current position of VR within the design

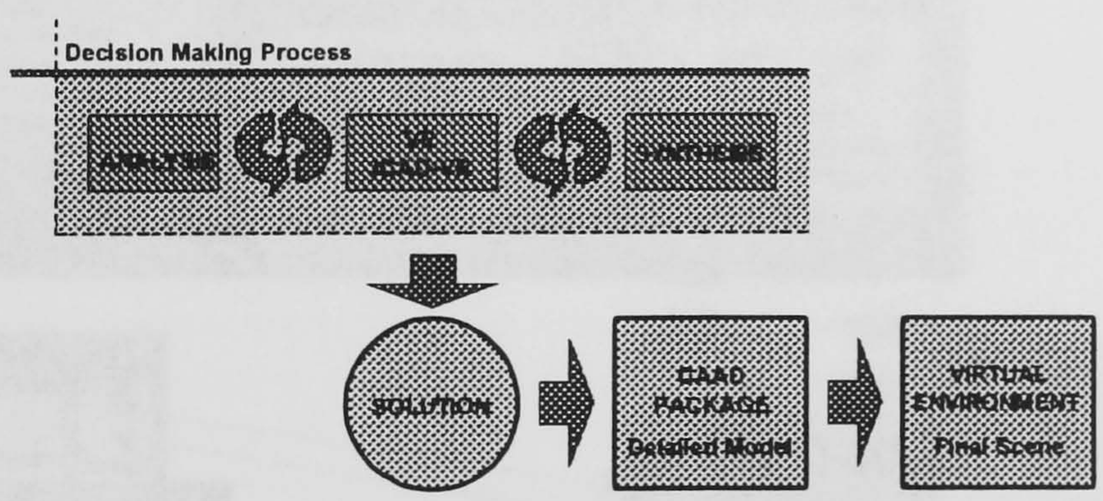


Figure 2. Proposed scenario

The system named JCAD-VR provides a flexible user-friendly immersive environment to support collaborative design on a synchronous base. It can be thought of as an investigation tool that allows the designer to sketch freely 3D objects within the virtual context. Moreover design solutions are shared in a synchronous fashion with other participants through the system's network-based architecture. Figure 2 shows the proposed scenario using JCAD-VR within the decision making process.

Here JCAD-VR provides the means for a more effective use of VR bridging between the phases of *analysis* and *synthesis*. VR is now employed at the very beginning of the decision making process when it is most likely to help in finding better design solutions. Once a desired solution is achieved the task of the creation of a very detailed 3D model and the final VR scene is given to appropriate CAAD packages. Moreover the participants are able to investigate design solution through concurrent design and synchronous collaboration.

JCAD-VR: The System Architecture

The two ideas upon which JCAD-VR is being built are:

?? that all the users present in the virtual world have to be able to share the same virtual environment in a "transparent fashion"
?? user interface (UI), instead of the traditional menu/windows based layout, is part of the virtual world itself. Any element of the interface becomes an object belonging to the 3D world and therefore it is treated as any other object. Each element of the interface can then be moved or scaled according to the user's needs.

The entire project is based on client-server architecture where every user logs into a virtual world and starts sharing design tasks with other users.

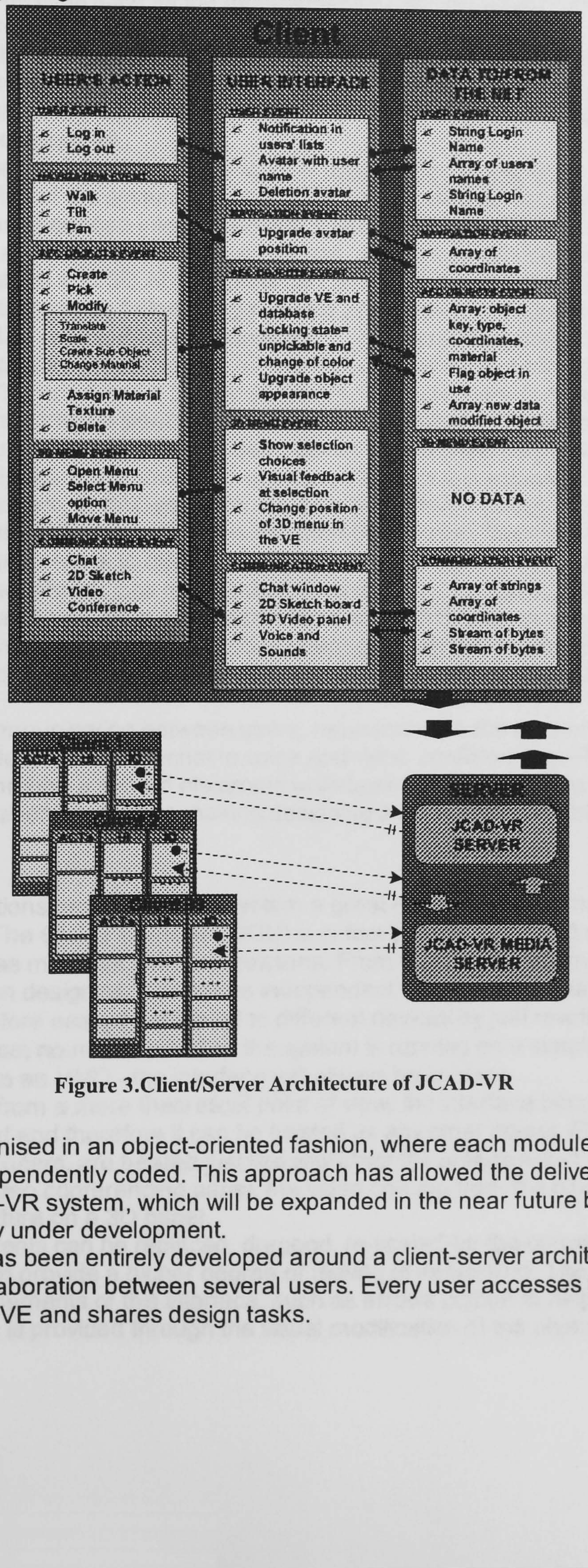


Figure 3.Client/Server Architecture of JCAD-VR

JCAD-VR is organised in an object-oriented fashion, where each module is able to fulfil certain task and it is independently coded. This approach has allowed the delivery of an initial functioning core of the JCAD-VR system, which will be expanded in the near future by adding several modules currently under development.

The system has been entirely developed around a client-server architecture to allow constant synchronous collaboration between several users. Every user accesses the virtual world, interacts with the VE and shares design tasks.

The Collaborative Approach

When JCAD-VR is initiated, the user is asked for a login name for the session and through an options panel he/she can decide which server to connect to and through which server port. This name will be used to identify participants and to communicate within the virtual world.

The system can be initiated in single mode or multiple screen mode. Single mode is set for the display device which consists of a standard computer screen; the multiple screen option has been included to allow devices such as the multi-projector display system processing the visual output of a Reality Centre which was used for the experiments.

In this phase it is also possible to activate or de-activate video conferencing facilities for the session. In instances when video conferencing is activated, support for video capturing device recognition and checking is provided. JCAD-VR provides also a stand-alone option in case collaboration is not required.

Once the system is initialised every window disappears freeing the space for the 3D graphic user interface (GUI) of the system. A set of 3D menus and icons appear on the screen and through them each user can interact with the system and with the other participants. A number of functions can be accessed through these menus, such as navigation and creation of objects. A number of 3D shapes and 3D AEC objects can be created and shared with other participants. The objects created can be following: geometric primitives (cones, boxes, spheres etc.) and architectural entities (walls, doors, windows etc.). The system routinely checks for constrains and allows only the possible modifications; for example a door cannot be moved onto or too close to another door. A "3D ruler" and a 3D panel close to the object are constantly providing the user with feedback related to the parameters which can be edited such as size, materials and cost.

The architecture of the system has been developed to allow every object created in the system to be assigned with a unique id-number. The ID is a combination of local ID and a user ID assigned by the server. In this way each object is attributed a unique number consistent for all the users in the system. When any object is selected by the user, this object is *locked* and such event is sent through the network to other users. Every time the user is about to modify an object this is checked against a network lock mechanism. This mechanism controls that several participants are not editing the same object at the same time and is designed in order to ensure consistency throughout the system. The system notifies every user internal database of any creation or modification of geometric objects within the virtual scene and broadcasts their numerical information.

To ensure communication between users, represented in the 3D world by avatars, different means are provided, from basic chat to voice and video conferencing. Freehand sketching in 2D is also possible through a shared electronic whiteboard. This architecture allows a real synchronous collaborative design making designing a true multi-user collaborative experience.

The Interface

Besides the functions provided by the system a great deal of attention has been put into the visual interface. The GUI or perhaps 3DGUI is in fact thought of as part of the virtual world itself.

This choice was made for two main reasons. From the technical point of view once the interface has been designed, it becomes independent from the visualisation device used. The system can therefore easily be adapted to different devices by just rewriting the code that is handling the device; no matter whether the system is running on a simple screen, on a Reality Center or linked to an HMD - the interface will always be in place.

Furthermore, from a more theoretical point of view, the interface becomes one of the elements of the virtual world and therefore it can be treated as any other object. Elements of the GUI such as panels, icons, rulers, are treated just like any other 3D entities within the VE. For instance, in the case of the video conferencing panel, the video coming from the other users is continuously rendered as a texture on a 3D panel.

All these elements can be replaced, dragged, re-scaled for the convenience of the user and perhaps they even provide a higher degree of feeling of immersion. The user interacts with the objects through elements of this interface, such as arrows placed to help the user editing the object. Feedback is provided through the visual modification of the object itself in the scene.

The 3D engine just renders all the possible changes of the VE: movements of avatars, video conferencing streams rendered on 3D panels and, most importantly, creation and modification of objects created within JCAD-VR.

Server Package

The server is made of two parts: a module which looks after the VE information to be broadcast, and another module which takes care of media streams and video conferencing tools. Both parts constitute the server system and they are closely linked to each other.

As an independent part of the framework the server has an autonomous and simpler interface that provides primarily information about the network status and transfer rate. A number of components are envisaged such as the communication status, the users on line, VR shared environments and the quality of the broadcast video for conferencing. Since the clients are communicating through independent processes, a future enhancement will allow the server to deal with several VR environments simultaneously.

The intrinsic multi-platform nature of JCAD-VR, inherited from the language used for coding, allows the server to transmit data to a broad range of platforms, from normal PCs to the supercomputer running a Reality Centre, and leaves the research team the freedom to test the software with several operating systems. The communication channel ensures the link between server and clients through a TCP/IP network.

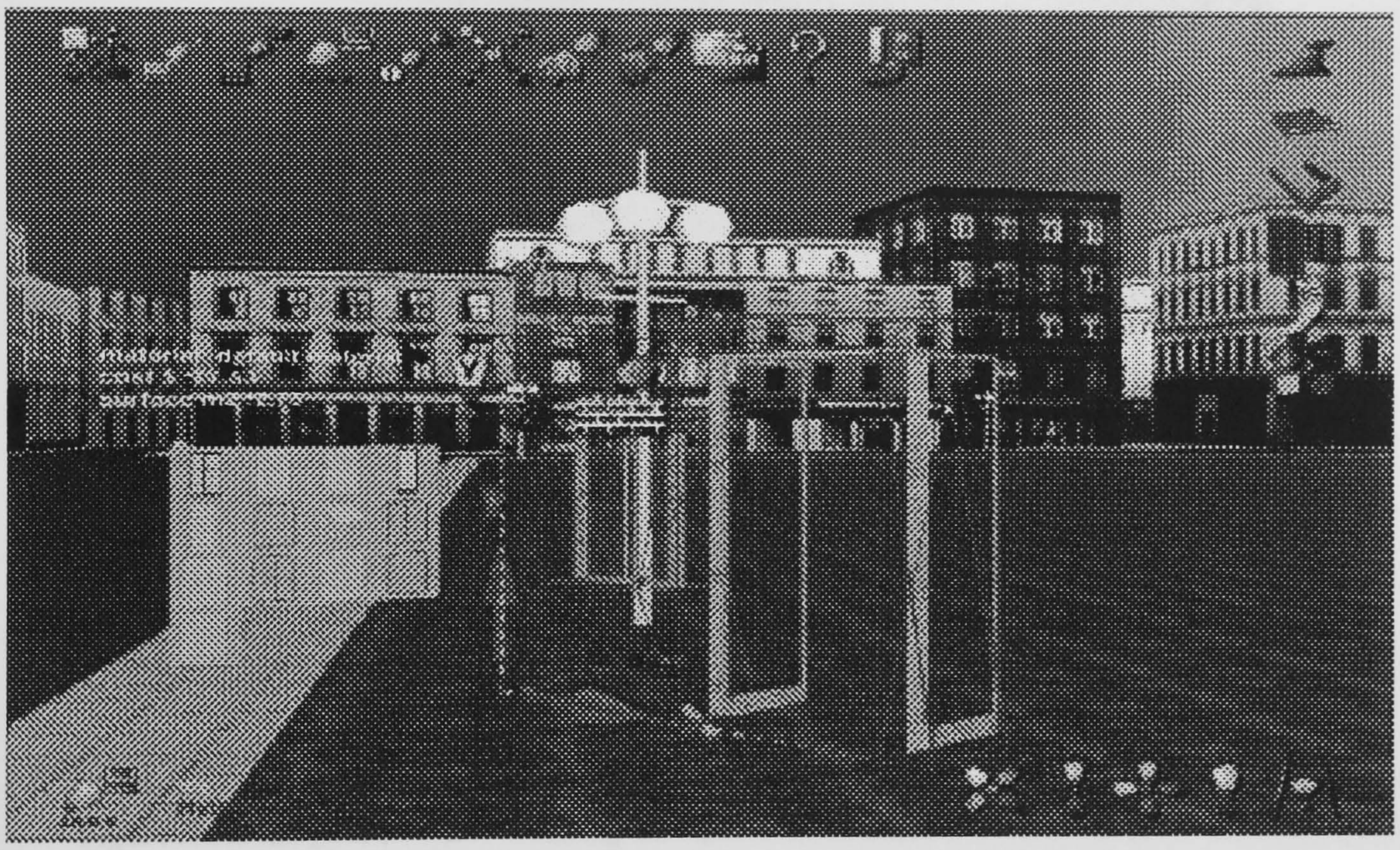


Figure 4. A screenshot from JCAD-VR

System Implementation and Hardware used

The whole framework of JCAD-VR was organised to allow concurrent software development, in a modular fashion, by individual members of the R+D team [3]. To facilitate this, an object oriented approach was identified as the most suitable one and the entire system was coded in Java? . The choice, even if less efficient in term of performances if compared with other choices, indeed offered great flexibility, true scalability and, last but not least, fully multi-platform support. Its network-centric nature, its multimedia integration together with the use of native graphic hardware and multi-processor support made it the obvious choice for the development of such real-time multimedia collaborative system.

The client application, in response to the obvious hardware limits imposed by the use of different hardware, has been written so that it can be easily customised to run on PCs as well as

on a Sgi supercomputers. The former are normal PCs whose video-card displays the virtual world only on a traditional window or at full screen. The latter is a 12-processors Sgi Onyx2 system running the Reality Centre at ABACUS, University of Strathclyde, Glasgow. When JCAD-VR is launched on the Sgi it can take advantage of its computational power to stretch itself on a 5 metre wide 2 metre tall tassellated screen where 3 Barco projectors create a 160 degree panoramic image.

The internal architecture of JCAD-VR is such that modules might be easily adapted to allow use of different VR devices such as CAVEs or Head-Mounted Displays, as well as several pointing devices such as a joystick, 3D mouse and VR Gloves. Further developments will include support for some of these devices.

From the collaborative point of view JCAD-VR is highly scalable and several communication media options are provided depending on the hardware limitations of the computer on which it is running.

The video conferencing facility has been coded using the Java™ Media Framework (JMF) which enables cross-platform capture, playback and streaming of audio and video at different transfer rates and resolutions. A great deal of effort has been expended by the research group to integrate the 3D module with the multimedia one.

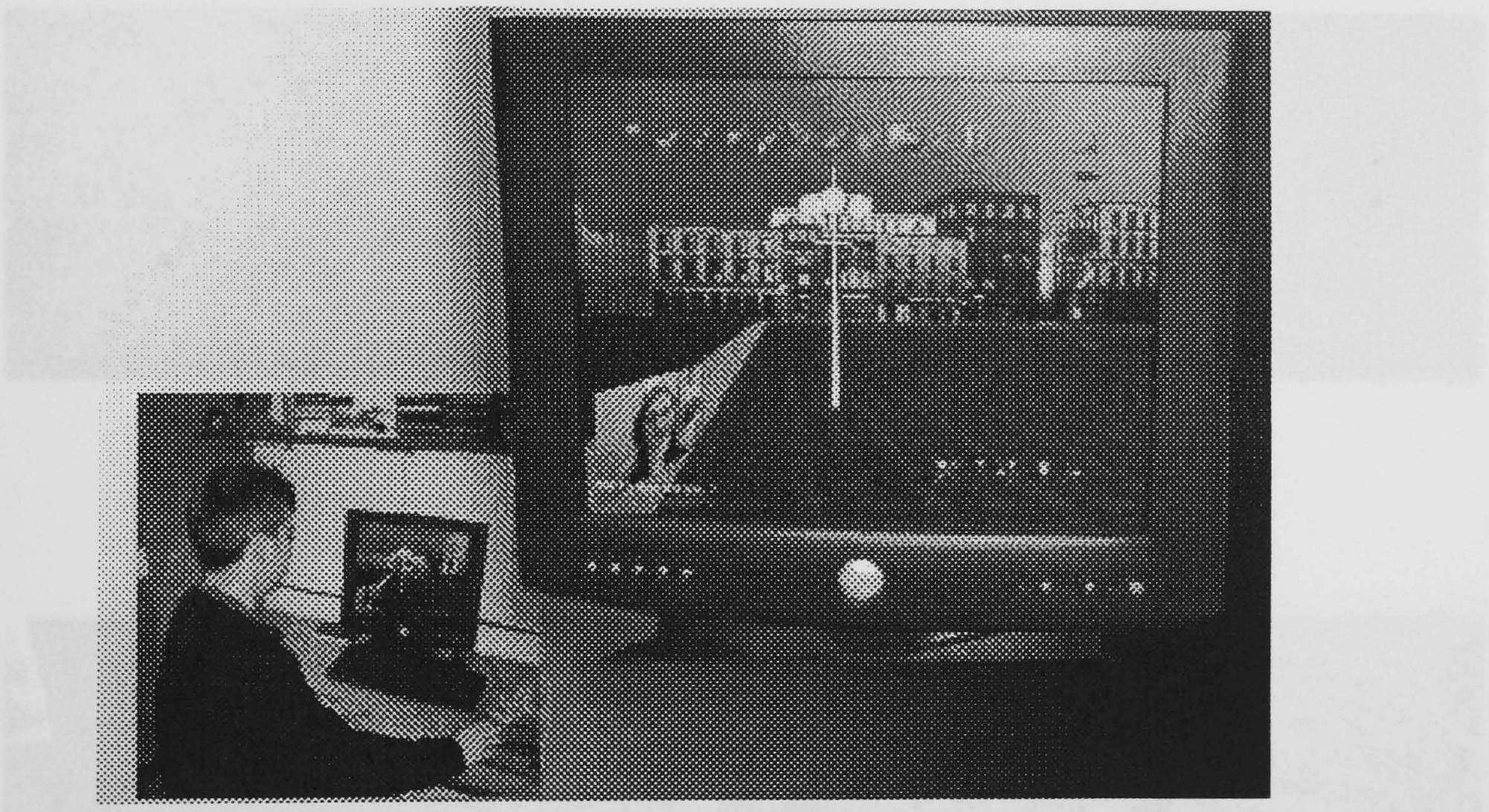


Figure 5. An image of JCAD-VR during an experiment of collaborative design

Collaborative Experiment

The obvious first line of inquiry regarding the usability and usefulness of the emerging system was in the academic environment within which it had been created. For the past three years, the academic with overall responsibility for CAAD teaching had offered an optional class, to fourth year students (and to students from less senior years with exceptional commitment and skills in CAAD) in the design application of innovative VR technologies. In Session 2001/2 three of thebstudents taking the class were introduced to JCAD-VR and invited to put the system to its first serious test.

Students Christoph Ackermann, Ross Marshall and Edward Wright were located, each with an appropriate workstation, in three different areas within the Department of Architecture, with fixed and hand-held video cameras covering the actions and observations of the students. Over the two-hour design session, the three students were invited to design an information centre in a public square and in a given urban context of Glasgow. The introduction to the project and to the specifics of the interface to JCAD-VR lasted a mere 30 minutes. This meagre introduction was purposeful and intended to test how intuitive (or not) the system was.

The in-house experiment was a revelation to the authors of this paper. Over the two-hour design period there was:

- ?? Fast and furious interaction amongst the three design participants within the common design environment; some 60/70 design scenarios were commonly generated, modified and agreed.
- ?? Both satisfaction and frustration amongst the participants was noted regarding the high degree of mutuality in the interactive process.
- ?? A real sense of having experienced a wholly immersive and shared design experience which heralds a future way of exploring and determining the configuration of the built environment.

The entire outcome of this limited experiment - with all its local and 'non-scientific' constraints - in common with the experiments in the late 1960' - is, thankfully, to stimulate further trials, tests and transformations.

The notion of a distributed design environment within which *all* can contribute, has, the authors claim, been significantly advanced.

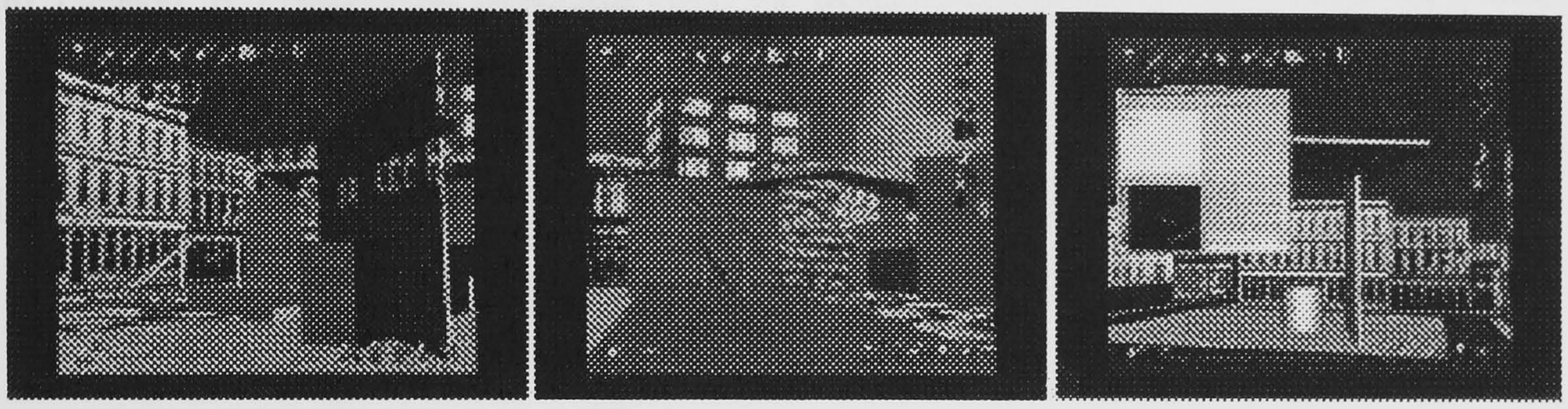


Figure 5. Screenshots from experiment

Figure 6. Pictures from the experiment

Conclusions and Further Developments

The prototype JCAD-VR system makes some steps toward the change of VR usage from mere presentation medium to a more powerful and effective design tool, and establishes the feasibility of VR becoming the interface for the next generation of computer aided design (CAAD) applications for architecture. Several enhancements are being considered for further development if the system including:

- ?? A voice driven interface enhancing friendliness of the user interface
- ?? Support for driving devices such as 6-degrees of freedom virtual glove

?? Implementation of a multi-environments server capable of dealing with several VR environments simultaneously.

References

[1] Barrilleaux, J. (2000) 3D User Interfaces with Java 3D, a guide to computer-human interaction in three dimensions. Manning, Greenwich.

[2] Bertol, D. (1997) Designing Digital Spaces, An architect's guide to Virtual Reality. John Wiley & Sons, Inc.

[3] Conti, G., Ucelli, G. and Maver, T. (2001) JCAD-VR - A Collaborative Design Tool in Virtual Reality, Proceedings of ECAADE 19. Helsinki, 2001, pp 454 – 459.

[4] Dorta, T. and LaLande, P. (1998) The impact of Virtual Reality on the design process, Proceedings from ACADIA Conference, Quebec City, 1998, pp 139 – 161

[5] Hughes, M., Hughes, C., Shoffner, M. and Winslow, M. (1997) Java Network Programming. Manning, Greenwich.

[6] Lawson, B. (1990) How designers think: the design process demystified. University Press, Cambridge.

[7] Sowizral, H. A. and Deering, M. F. (1999) The Java 3D API and Virtual Reality, IEEE Computer Graphics and Applications, May/June.

[8] Petric, J., Ucelli, G. and Conti, G. (2001) Educating the Virtual Architect, Proceedings of ECAADE 19. Helsinki, 2001, pp 388 – 393.

Conti, G., Ucelli, G. and Petric, J. (2002). JCAD-VR: a collaborative design tool for architects. In *Proceedings of the 4th international conference on Collaborative virtual environments, Bonn, Germany, 2002* (pp. 153-154). New York: ACM Press.

JCAD-VR: a collaborative design tool for architects

Giuseppe Conti
ABACUS
University of Strathclyde
Glasgow G4 0NG, Scotland, UK
Tel. +44 141 548 3407
giuseppe.conti@strath.ac.uk

Giuliana Ucelli
ABACUS
University of Strathclyde
Glasgow G4 0NG, Scotland, UK
Tel. +44 141 548 3407
giuliana.ucelli@strath.ac.uk

Jelena Petric
ABACUS
University of Strathclyde
Glasgow G4 0NG, Scotland, UK
Tel. +44 141 548 3407
j.petric@strath.ac.uk

ABSTRACT

Today the development of network-based virtual communities and the use of avatars have brought a new level of complexity to the meaning of virtuality, providing the technology for remote presence and collaborative experiences. In this project the intention was to pursue this articulated vision of Virtual Reality (VR) in order to assist all the participants - professional and client body - during the early stages of the design process. The objective was to provide a tool that is capable of creating 3D shapes in a shared VR environment, thus allowing the design and its evolution to be shared. The use of the Java™ programming language was a natural choice for this project. Because of Java™'s performance scalability and hardware independence the concept of CAAD has been extended, making it possible to create a VR environment that can co-exist between high-end supercomputers and standard PCs. The project is currently being tested using PCs workstations and an SGI system running a Reality Centre.

The research reported in this paper describes the development and application of software that aims to increase the opportunity for architects to collaborate within virtual worlds and enable effective and transparent information exchange.

Categories and Subject Descriptors

D.2.2 [**User Interface**]: 3D User Interface, H.5.3. [**Computer-supported cooperative work**] Collaborative Design, I.3.7 [**Virtual Reality**] Design system for Virtual Reality, J.6 [**Computer-aided design (CAD)**]

General Terms

Management, Design, Human Factors.

Keywords

Synchronous Collaborative Design, Virtual Environment, Interface, Architecture, Design Process.

1. INTRODUCTION

Historically architects have experienced the need to prove their design proposals using physical models. Due to the intrinsic spatial nature of the act of designing, even the most accurate 2D paper representation is usually not suitable to deliver the complexity of some architectonic ideas.

In the last decades the profession of the architect has been deeply affected by the *digital revolution* and the use of Computer Aided Architectural Design (CAAD) tools is nowadays part of the daily practice in most architecture firms. But in the last few years the "CAAD community" is experiencing a new revolution that is leading the move from static representation, based on 2D renderings or pre-recorded animations (considered as a sequence of 2D images), to dynamically generated 3D representations. Real-time navigation and interaction, typical of VR environments, provide just that fluent interface enabling the exploration of the design proposals that architects have not been able to get with any other media.

The increasing growth of computational resources and hardware power is probably preparing the anticipated transition to desktop VR applications, making them truly feasible tools for everyday practice. Furthermore, the recent growth of network-based virtual communities has brought a new level of complexity to the notion of virtual spaces, turning the profession of architect into something that might now resembles the one of the *virtual architect*.

Although VR is nowadays a quite mature technology, it is seldom used in architecture throughout the design process, but more often it is merely used as a powerful presentation technique.

Design methodologists in the past agreed on the need for iterative cycles between several phases of the design process. From studies concerning designers' behaviour [6] many authors observed an indefinite number of return loops from the moment when gathering of information and structuring of the design problem take place (known as *analysis*) to the one when design solutions are generated (known as *synthesis*).

The use of Virtual Reality within the design process could give to the designer an appropriate quick and practical response to his/her need of iteration and search for design solutions. Moreover it enables the capture of more information than would be possible to capture with the use of the traditional media and makes the checking of the design solutions more efficient by enhancing simulation capabilities. Furthermore VR broadens the boundaries of traditional perception by providing experiences of worlds not necessarily real or material. In short it is the perfect simulation medium for architects investigating design solutions.

It is then highly predictable that in the near future VR will become the interface for the next generation of computer aided drawing (CAD) applications and we can anticipate the change of its use from a mere presentation medium to a more powerful and effective design tool.

CAD/CAAD packages are very powerful but often complex rendering tools, which were not meant to be investigation tools, and therefore generating 3D models is often impractical and time-consuming. Therefore such models are usually employed when every design decision has been already taken.

Virtual environments (VE) are often created using CAAD packages for refinements, adjustments and exportation based on traditional 3D scenes. Documenting the evolution and development of design by constant updating of 3D models is an expensive business, and obviously even more expensive is the upgrading of the VEs generated from these models.

In these circumstances the use of VR would just increase costs. As result VR is relegated to the end of the design process rather then being used to engage design creativity through immersive design.

In the traditional scenario the decision making process does not take advantage of the technology but relegates the use of VR to the end of the process as a more convincing tool to impress contractors and clients. Only once the final solution has been achieved it is worth investing time into more powerful visualisation media.

Being aware of this background the research group engaged itself in the development of a VR system, named JCAD-VR, to help designers in the initial stages of the design process to take advantage of the VR as a new design tool.

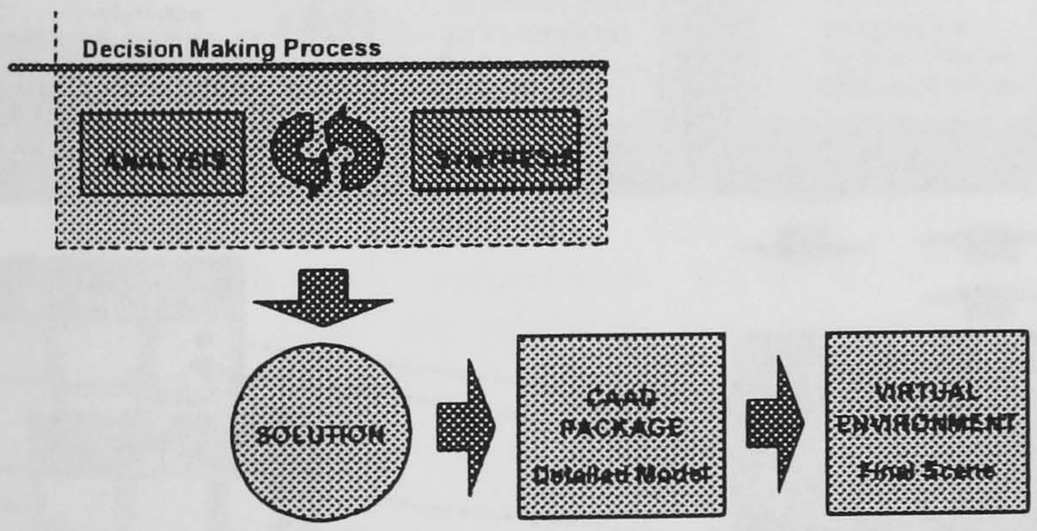


Figure 1. The current position of VR within the design process.

JCAD-VR provides a flexible user-friendly immersive environment to support collaborative design on a synchronous base.

It can be thought of as an investigation tool that allows the designer to sketch freely 3D shapes within the virtual context. Moreover design solutions are shared in a synchronous fashion with other participants through the system's network-based architecture. Figure 2 shows the proposed scenario using JCAD-VR within the decision making process.

Here JCAD-VR provides the means for a more effective use of VR bridging between the phases of *analysis* and *synthesis*. VR is now employed at the very beginning of the decision making process when it is most likely to help in finding better design solutions. Once a desired solution is achieved the task of the creation of a very detailed 3D model and the final VR scene is given to appropriate CAAD packages.

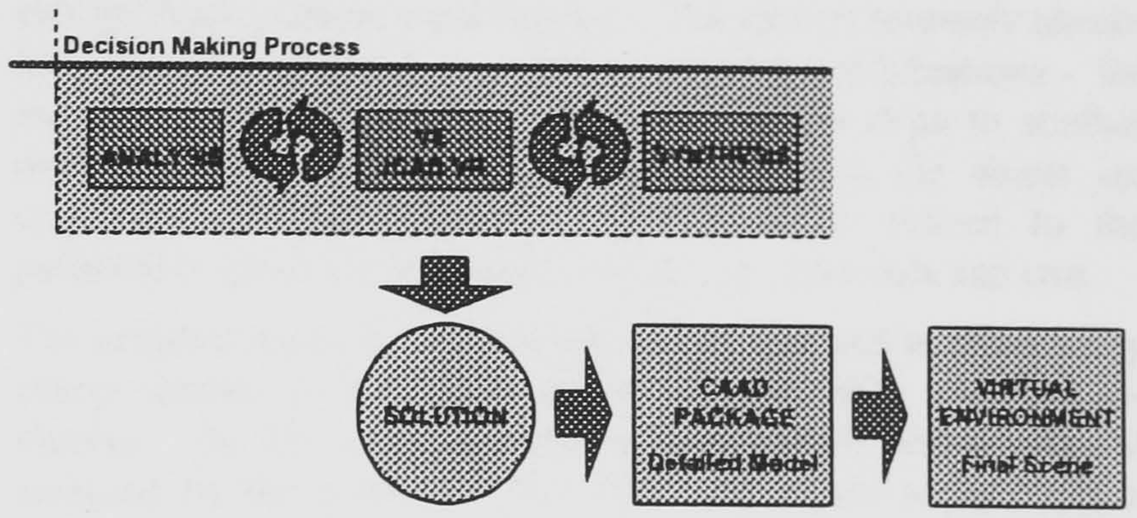


Figure 2. Proposed scenario

Moreover the participants are able to investigate design solutions through concurrent design and synchronous collaboration.

This paper reports the present state of the JCAD-VR system (Java™ based Collaborative Architectural Design tool in VR) and will highlight its future development and testing.

2. THE SYSTEM

The two ideas upon which JCAD-VR is being built are:

- that all the users present in the virtual world have to be able to share the same virtual environment in a "transparent fashion"
- user interface (UI), instead of the traditional menu/windows based layout, is part of the virtual world itself. Any element of the interface becomes an object belonging to the 3D world and therefore it is treated as any other object. Each element of the interface can then be moved or scaled according to the user's needs.

The entire project is based on client-server architecture where every user logs into a virtual world and starts sharing design tasks with other users.

JCAD-VR is organised in an object-oriented fashion, where each module is able to fulfil certain task and it is independently coded. This approach has allowed the delivery of an initial functioning core of the JCAD-VR system which will be expanded in the near future by adding several modules currently under development.

3. SYSTEM ARCHITECTURE

The system has been entirely developed around a client-server architecture to allow constant synchronous collaboration between several users. Every user accesses the virtual world, interacts with the VE and shares design tasks.

3.1 Client Side

3.1.1 The Collaborative approach

When JCAD-VR is initiated, the user is asked for a login name for the session and through an options panel he/she can decide which server to connect to and through which server port. This name will be used to communicate within the virtual world.

The system can be initiated in single mode or multiple screen mode. Single mode is set for the display device which consists of a standard computer screen; the multiple screen option has been included to allow devices such as the multi-projector display system processing the visual output of a Reality Centre used for the experiments.

In this phase it is also possible to activate or de-activate video conferencing facilities for the session. In instances when video conferencing is activated, support for video capturing device recognition and checking is provided.

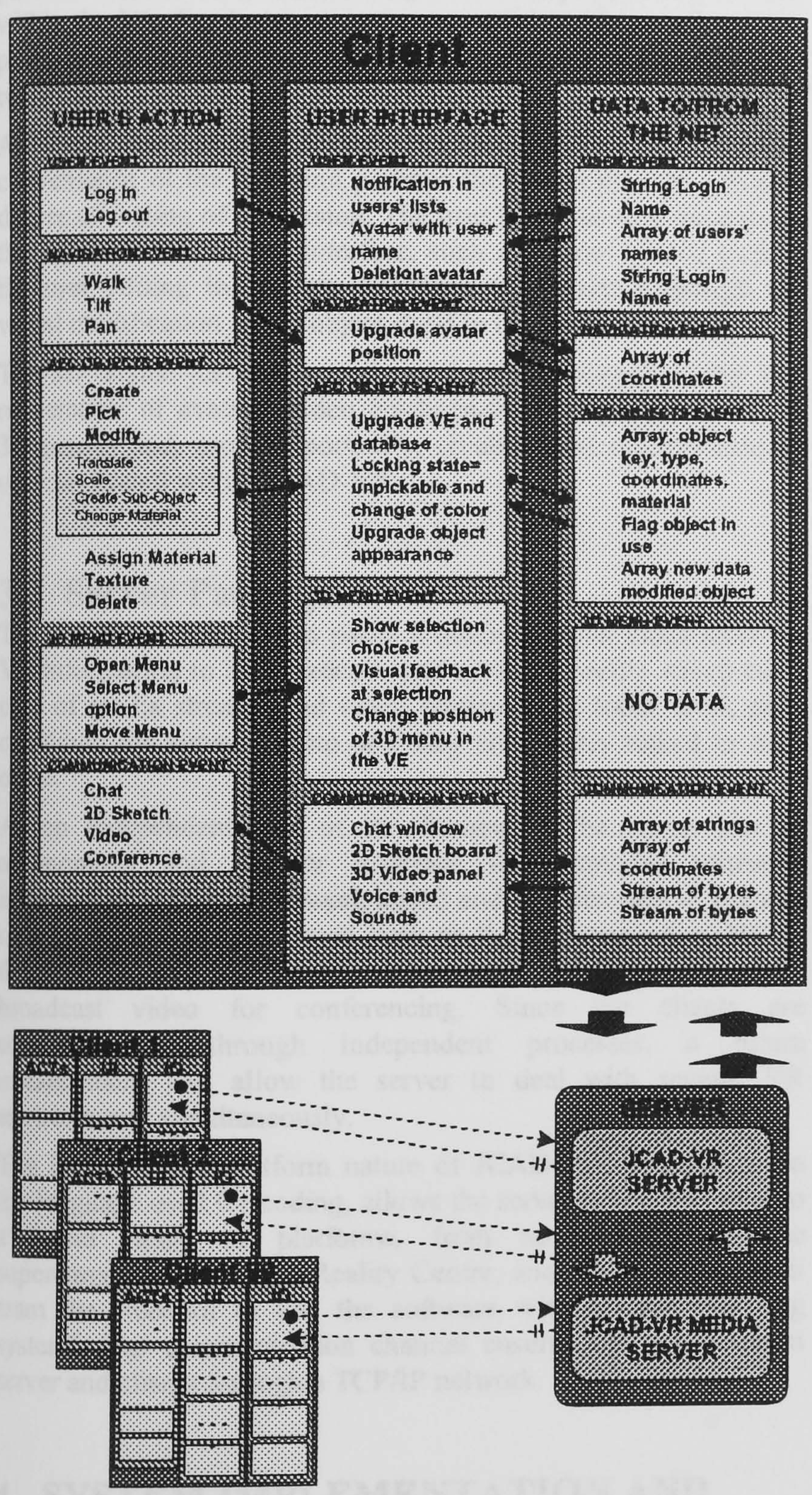


Figure 3.Client/Server Architecture of JCAD-VR

JCAD-VR provides also a stand-alone option in case collaboration is not required.

Once the system is initialised every window disappears freeing the space for the 3D graphic user interface (GUI) of the system. A set of 3D menus and icons appear on the screen and through them each user can interact with the system and with the other participants. A number of functions can be accessed through these menus, such as navigation and creation of objects. A number of 3D shapes and 3D AEC objects can be created and shared with other participants. The objects created can be the following: geometric primitives (cones, boxes, spheres etc.) and architectural entities (walls, doors, windows etc.). The system routinely checks for constrains and allows only the possible modifications - for example a door cannot be moved onto or too close to another door. A "3D ruler" and a 3D panel close to the object are constantly providing the user with feedback related to the parameters which can be edited such as size, materials and cost.

The architecture of the system has been developed to allow every object created in the system to be assigned with a unique id-number. The ID is a combination of local ID and a user ID assigned by the server. In this way each object is attributed a unique number consistent for all the users in the system. When any object is selected by the user, this object is *locked* and such event is sent through the network to other users. Every time the user is about to modify an object this is checked against a network lock mechanism. This mechanism controls that several participants are not editing the same object at the same time and is designed in order to ensure consistency throughout the system.

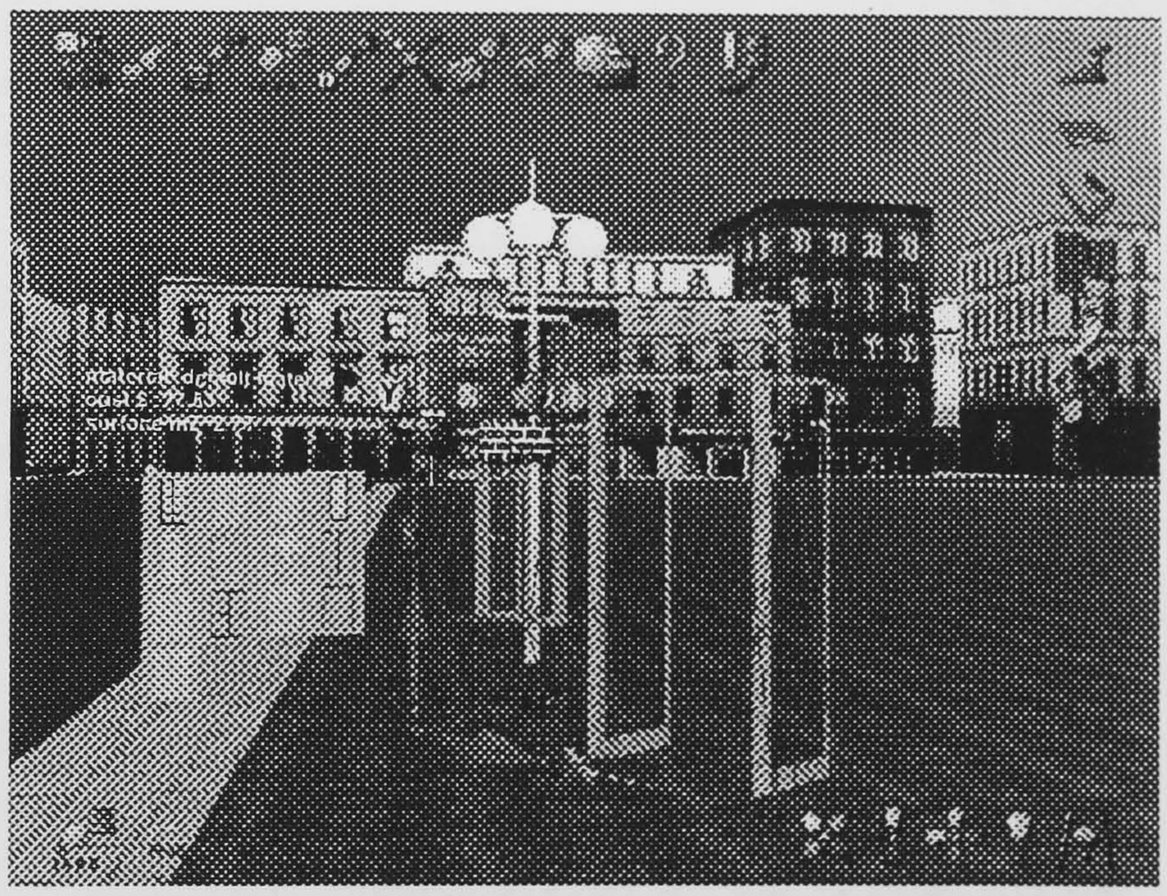


Figure 4. A screenshot from JCAD-VR

The system notifies every user internal database of any creation or modification of geometric objects within the virtual scene and broadcasts their numerical information.

To ensure communication between users, represented in the 3D world by avatars, different means are provided, from basic chat to voice and video conferencing. Freehand sketching in 2D is also possible through a shared electronic whiteboard. This architecture allows a real synchronous collaborative design making designing a true multi-user collaborative experience.

3.1.2 The interface

Besides the functions provided by the system a great deal of attention has been put into the visual interface. The GUI or perhaps 3DGUI is in fact thought of as part of the virtual world itself.

This choice was made for two main reasons. From the technical point of view once the interface has been designed, it becomes independent from the visualisation device used. The system can therefore easily be adapted to different devices by just rewriting the code that is handling the device; no matter whether the system is running on a simple screen, on a Reality Center or linked to an HMD - the interface will always be in place.

Furthermore, from a more theoretical point of view, the interface becomes one of the elements of the virtual world and therefore it can be treated as any other object. Elements of the GUI such as panels, icons, rulers, are treated just like any other 3D entities within the VE. For instance, in the case of the video conferencing panel, the video coming from the other users is continuously rendered as a texture on a 3D panel.

All these elements can be replaced, dragged, re-scaled for the convenience of the user and perhaps they even provide a higher degree of feeling of immersion. The user interacts with the objects through elements of this interface, such as arrows placed to help the user editing the object. Feedback is provided through the visual modification of the object itself in the scene.

The 3D engine just renders all the possible changes of the VE: movements of avatars, video conferencing streams rendered on 3D panels and, most importantly, creation and modification of objects created within JCAD-VR.

3.2 Server Package

The server is made of two parts: a module which looks after the VE information to be broadcast, and another module which takes care of media streams and video conferencing tools. Both parts constitute the server system and they are closely linked to each other.

As an independent part of the framework the server has an autonomous and simpler interface that provides primarily information about the network status and transfer rate. A number of components are envisaged such as the communication status, the users on line, VR shared environments and the quality of the broadcast video for conferencing. Since the clients are communicating through independent processes, a future enhancement will allow the server to deal with several VR environments simultaneously.

The intrinsic multiplatform nature of JCAD-VR, inherited from the language used for coding, allows the server to transmit data to a broad range of platforms, from normal PCs to the supercomputer running a Reality Centre, and leaves the research team the freedom to test the software with several operating systems. The communication channel ensures the link between server and clients through a TCP/IP network.

4. SYSTEM IMPLEMENTATION AND HARDWARE USED

The whole framework of JCAD-VR was organised to allow concurrent software development, in a modular fashion, by individual members of the R+D team [3]. To facilitate this, an object oriented approach was identified as the most suitable one and the entire system was coded in Java™.

The choice, even if less efficient in term of performances if compared with other choices, indeed offered great flexibility, true scalability and, last but not least, fully multi-platform support. Its network-centric nature, its multimedia integration together with the use of native graphic hardware and multi-processor support made it the obvious choice for the development of such real-time multimedia collaborative system.

The client application, in response to the obvious hardware limits imposed by the use of different hardware, has been written so that it can be easily customised to run on PCs as well as on a Sgi supercomputers. The former are normal PCs whose video-card displays the virtual world only on a traditional window or at full screen. The latter is a 12-processors Sgi Onyx2 system running the Reality Centre at ABACUS, University of Strathclyde, Glasgow. When JCAD-VR is launched on the Sgi it can take advantage of its computational power to stretch itself on a 5 metre wide 2 metre high tassellated screen where 3 Barco projectors create a 160 degree panoramic image.

The internal architecture of JCAD-VR is such that modules might be easily adapted to allow use of different VR devices such as CAVEs or Head-Mounted Displays, as well as several pointing devices such as a joystick, 3D mouse and VR Gloves. Further developments will include support for some of these devices.

From the collaborative point of view JCAD-VR is highly scalable and several communication media options are provided depending on the hardware limitations of the computer on which it is running.

The video conferencing facility has been coded using the Java™ Media Framework (JMF) which enables cross-platform capture, playback and streaming of audio and video at different transfer rates and resolutions. A great deal of effort has been expended by the research group to integrate the 3D module with the multimedia one.

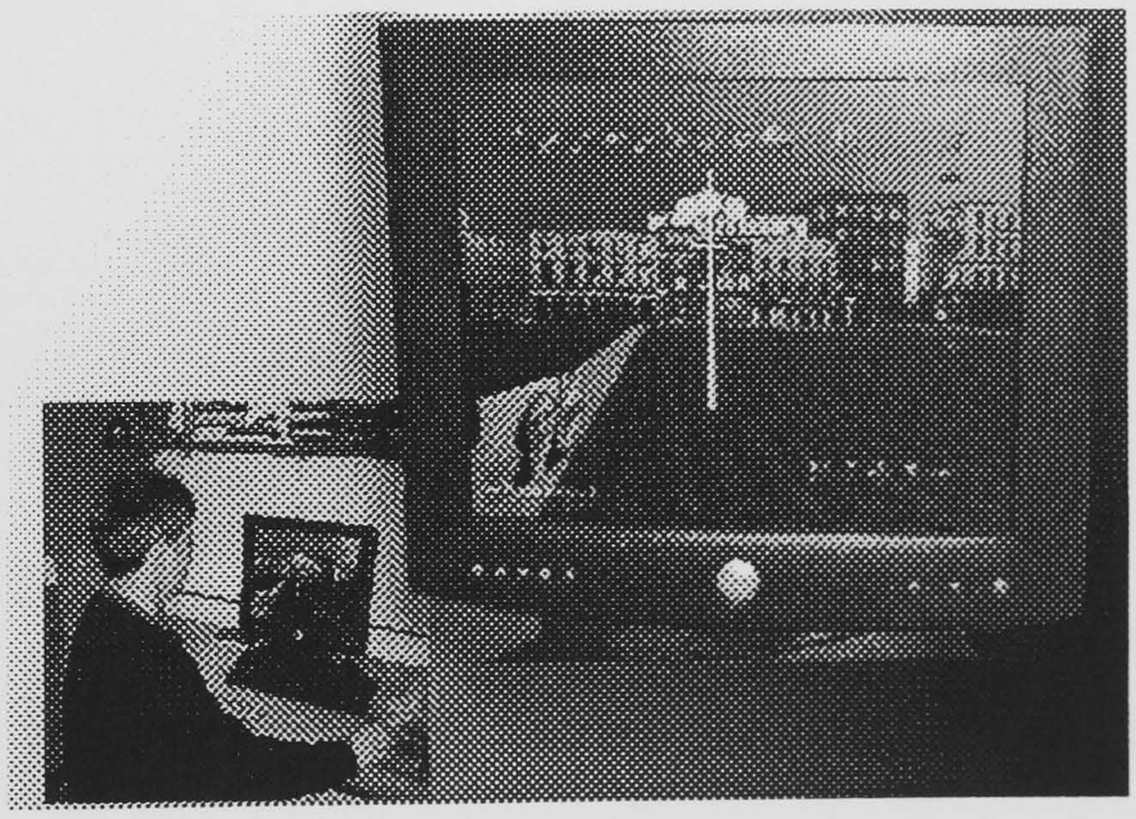


Figure 5. An image of JCAD-VR during an experiment of collaborative design

5. COLLABORATIVE EXPERIMENTS

A major test of the potential of JCAD-VR to contribute to collaborative design is currently underway. Under the auspices of ECAADE, an association of some 150 schools of Architecture within Europe, a bilateral experiment is taking place, between the Technical University of Eindhoven and University of Strathclyde

which seeks to establish, subjectively, if not statistically, the power of synchronous design support systems such as JCAD-VR.

The inter-institutional collaborative experiment will be informed by more local, simulated, distributed design decision exercises featuring committed students within the University of Strathclyde.

6. CONCLUSIONS AND FURTHER DEVELOPMENTS

The prototype JCAD-VR system makes some steps toward the change of VR usage from mere presentation medium to a more powerful and effective design tool, and establishes the feasibility of VR becoming the interface for the next generation of computer aided drawing (CAAD) applications for architecture. Several enhancement are being considered for further development of the system including:

- A voice driven interface enhancing friendliness of the user interface
- Support for driving devices such as 6-degree of freedom/virtual glove
- Implementation of a multi-environments server capable of dealing with several VR environments simultaneously.

7. REFERENCES

[1] Barrilleaux, J. (2000) 3D User Interfaces with Java 3D, a guide to computer-human interaction in three dimensions. Manning, Greenwich.

[2] Bertol, D. (1997) Designing Digital Spaces, An architect's guide to Virtual Reality. John Wiley & Sons, Inc.

[3] Conti, G., Ucelli, G. and Maver, T. (2001) JCAD-VR - A Collaborative Design Tool in Virtual Reality, Proceedings of ECAADE 19. Helsinki, 2001, pp 454 – 459.

[4] Dorta, T. and LaLande, P. (1998) The impact of Virtual Reality on the design process, Proceedings from ACADIA Conference, Quebec City, 1998, pp 139 – 161

[5] Hughes, M., Hughes, C., Shoffner, M. and Winslow, M. (1997) Java Network Programming. Manning, Greenwich.

[6] Lawson, B. (1990) How designers think: the design process demystified. University Press, Cambridge.

[7] Sowizral, H. A. and Deering, M. F. (1999) The Java 3D API and Virtual Reality, IEEE Computer Graphics and Applications, May/June.

[8] Petric, J., Ucelli, G. and Conti, G. (2001) Educating the Virtual Architect, Proceedings of ECAADE 19. Helsinki, 2001, pp 388 – 393.

Petric, J., Ucelli, G. and Conti, G. (2002). Participatory Design in Collaborative Virtual Environments. Accepted paper at the 6th SIGRADI Conference, Caracas.

Jelena Petric, Giuliana Ucelli, Giuseppe Conti
j.petric@strath.ac.uk , giuliana.ucelli@strath.ac.uk , giuseppe.conti@strath.ac.uk
ABACUS, Department of Architecture, University of Strathclyde, Glasgow, UK

Participatory Design in Collaborative Virtual Environments

Abstract

This paper re-establishes the theoretical framework for participatory design evolved in the late sixties and early seventies as part of the movement towards a more explicit design methodology and attempts an explanation of why the concept failed to gain commitment from the architectural and urban design professionals. The issue of user participation in the processes of building and urban design is enjoying renewed attention following its relative neglect over the last 20 years due, in large measure, to significant advances in emerging information technologies, particularly multimedia, virtual reality and internet technologies.

This paper then gives an account of two significant and relevant developments in the evolution of the application of information technologies with which the authors have been engaged. These are:

- a responsive and interactive interface to wholly immersive and realistic virtual reality representations of proposed buildings and urban neighbourhoods.
- an intuitive and platform-independent VR modelling environment allowing collaborative evolution of the scheme from within the virtual world.

The efficacy of these IT developments is tested in the context of a design exercise in which three designers, from distributed locations and using different computer platforms, collaboratively design an Information Centre from within the virtual world.

Resumo

Este trabalho restabelece uma estrutura teórica de projeto participativo originado no final dos anos 60 e início dos anos 70, como parte de um movimento direcionado a uma metodologia mais explícita de projeto e busca uma resposta para as razões do seu fracasso para a obtenção de um envolvimento maior dos profissionais das áreas de arquitetura e desenho urbano. A questão da participação do usuário nos processos projetuais arquitetônicos e urbanísticos está tendo uma renovada atenção após a relativa rejeição dos últimos 20 anos. Este fato deve-se, em grande parte, a avanços significativos em tecnologias da informação emergentes, particularmente multimídia, realidade virtual e tecnologias ligadas à Internet.

Assim, este trabalho pretende demonstrar dois desenvolvimentos significativos na aplicação da tecnologia da informação nos quais os autores estão envolvidos:

- uma interface sensível e interativa, completamente imersiva e representações realísticas de realidade virtual de edifícios e áreas urbanas.
- um ambiente de modelagem de Realidade Virtual, intuitivo e para múltiplas plataformas, permitindo a evolução colaborativa de um projeto a partir de um mundo virtual.

O impacto destes desenvolvimentos de TI é demonstrado a partir da realização de um projeto de uma estrutura de lazer para uma comunidade de usuários com deficiências físicas.

Design Decision Making

Architectural design is a multi-faceted occupation which requires, for its successful performance, a mixture of intuition, craft skills and detailed knowledge of a wide range of practical and theoretical matters. It is a cyclical process in which groups of people work towards a somewhat ill-defined goal in a series of successive approximations. There is no 'correct' method of designing and, although it is recognised that the process can be divided into separate phases, there is no generally accepted sequence of work that might guide design teams in the direction of achieving a satisfactory solution. Indeed, there are no solutions to design problems in the way that there are solutions to mathematical problems: the best that can be hoped for is an outcome which satisfies the maximum number of constraints which bound the area of concern. Furthermore, design is not an algorithmic process in which the desired

conclusion can be reached by the application of step-by-step procedures - first finalising this aspect, then that. It is a fluid, holistic process wherein at any stage all the major parts have to be manipulated at once. In this sense, it is less like solving a logical puzzle and more like riding a bicycle, blindfold, whilst juggling.

Despite the complexity of the design decision-making process the emerging new generation of computer-based models is already having an impact on how design is performed and, hence, on the quality of design. The impact stems from the fact that the new models, as opposed to paper-based plans and elevations or other conventional forms, are predictive rather than descriptive; dynamic rather than static; explicit rather than implicit and, above all, permit a more-or-less continuous and interactive assessment of a developing design on cost and performance.

Evidence is growing of the advantages offered by the application of computers in design, and these can be summarised as follows:

Widening the Search for Solutions

Access to programs which dynamically predict the cost and performances characteristics of optional design proposals can increase the scope of search for good solutions by as much as ten-fold. Not only is the search coverage extended, it is also more purposefully directed because designers are able to compare the quality of any one tentative solution against the quality of all previous solutions.

Greater Integration in Decision-Making

In conventional working, a great deal of design time is lost as proposals are passed to and fro between the architect (who tends to be the originator) and the other specialist members of the design team (who tend to the "checkers"). Quite frequently the scheme on which the architect has lavished time and effort is found by one or other of the specialists to be infeasible. With access to appropriate appraisal techniques embodied in computer programs, it is possible to check a proposal against a wide range of criteria from the outset of the design activity. Moreover, it is entirely practical (though not yet a widespread working method) for all members of the design team to have access to, and operate on, the common design model whether or not they share a design office. The models, then, can provide a strong integrating force in design team working.

Improving Design Insights

Apart from the use of appraisal programs to search for better designs, the programs can be used in a research and development context to provide insights into the way in which particular design decisions affect cost and performance. Typically, a designer working in this mode would select an existing building for study, then, keeping all other design variables constant (insofar as this is possible), systematically vary one factor while recording the cost/performance output from the program. In this manner, the architect can establish sets of causal relationships which provide powerful insights into structure of design decision-making.

Differentiation of Objective and Subjective Judgements

Contrary to the early fears of many architectural practitioners, the use of CAAD techniques focuses increased attention on subjective value judgements rather than less. As measurable attributes of optional designs are made more explicit, the necessary value judgements are forced to the surface of design activity and thereby, themselves become more explicit. The effect of this is to make it clear to designers and their clients, which judgements are based on quantifiable criteria and which on subjective and intuitive concepts.

Evidence of the degree to which computer-generated cost/performance information promotes effective value judgement, throws into sharp focus the crucial question: **whose** value judgement? This question was, for the first time, seriously addressed in the Design Participation Conference in Manchester in 1971. At that time, however, the human-machine interface was too primitive for the concept of useful participation by the users of buildings to be achieved. The new technologies of VR and Multimedia give real prospects for participation.

Virtual Reality

The essence of Virtual Reality (VR) is that the user, instead of looking through the window of his/her computer screen at a virtual world, can in effect, step through the window and enter the virtual world itself. This enhances:

- **Immersion**: Users are completely surrounded by the environment.
- **Presence**: Being surrounded the participant has actually the sensation of being *in* the environment. The Virtual Environment becomes then a place on its own and its perception is similar to real environments.
- **Interactivity**: This is surely the most important *feature* provided by VR: the environment allows the participant to be involved and the result of the actions done by the participant is visualized in the VE.

- **Autonomy**: Participants are neither constrained in paths nor in views preset by others but have the freedom and autonomy to explore any single part of the environment.
- **Collaboration**: Multiple users are able to take part and to interact in the same VE.

The effectiveness of VR for the *presentation* of design proposals is well established but its potential in the *process* of design has yet to be realized. However the speed at which technology is evolving is making the application of VR within the design professions a feasible approach. AEC companies have already started to evaluate how time consuming the traditional presentation path can be where animations or walkthroughs are used to show designs solutions to their clients. In fact traditional CAD/CAAD systems are used as rendering tools more than design tools. Any change on design solutions is subject to the inevitable delay of having to step back to the CAD/CAAD systems and then the result must be rendered again to be eventually visualized. This approach is obviously not only inconvenient but time consuming and therefore costly. The consequence of these issues is that some design and manufacturing companies have already started to investigate how VR can be used within the design process. The research and development reported in this paper hopefully makes a signal contribution to this investigation.

The JCAD-VR Concept

In the Department of Architecture and Building Science at the University of Strathclyde, the ABACUS group has been building a prototype design decision support system known as JCAD-VR.

The system can be initiated in single mode or multiple screen mode. Single mode is set for the display device which consists of a standard computer screen; the multiple screen option has been included to allow devices such as the multi-projector display system processing the visual output of a Reality Centre which was used for the experiments.

In this phase it is also possible to activate or de-activate video conferencing facilities for the session. In instances when video conferencing is activated, support for video capturing device recognition and checking is provided. JCAD-VR provides also a stand-alone option in case collaboration is not required.

Once the system is initialised every window disappears freeing the space for the 3D graphic user interface (GUI) of the system. A set of 3D menus and icons appear on the screen and through them each user can interact with the system and with the other participants. A number of functions can be accessed through these menus, such as navigation and creation of objects. A number of 3D shapes and 3D AEC objects can be created and shared with other participants. The objects created can be following: geometric primitives (cones, boxes, spheres etc.) and architectural entities (walls, doors, windows etc.). The system routinely checks for constrains and allows only the possible modifications; for example a door cannot be moved onto or too close to another door. A "3D ruler" and a 3D panel close to the object are constantly providing the user with feedback related to the parameters which can be edited such as size, materials and cost.

The architecture of the system (Figure 1) has been developed to allow every object created in the system to be assigned with a unique id-number. The ID is a combination of local ID and a user ID assigned by the server. In this way each object is attributed a unique number consistent for all the users in the system. When any object is selected by the user, this object is *locked* and such event is sent through the network to other users. Every time the user is about to modify an object this is checked against a network lock mechanism. This mechanism controls that several participants are not editing the same object at the same time and is designed in order to ensure consistency throughout the system. The system notifies every user internal database of any creation or modification of geometric objects within the virtual scene and broadcasts their numerical information.

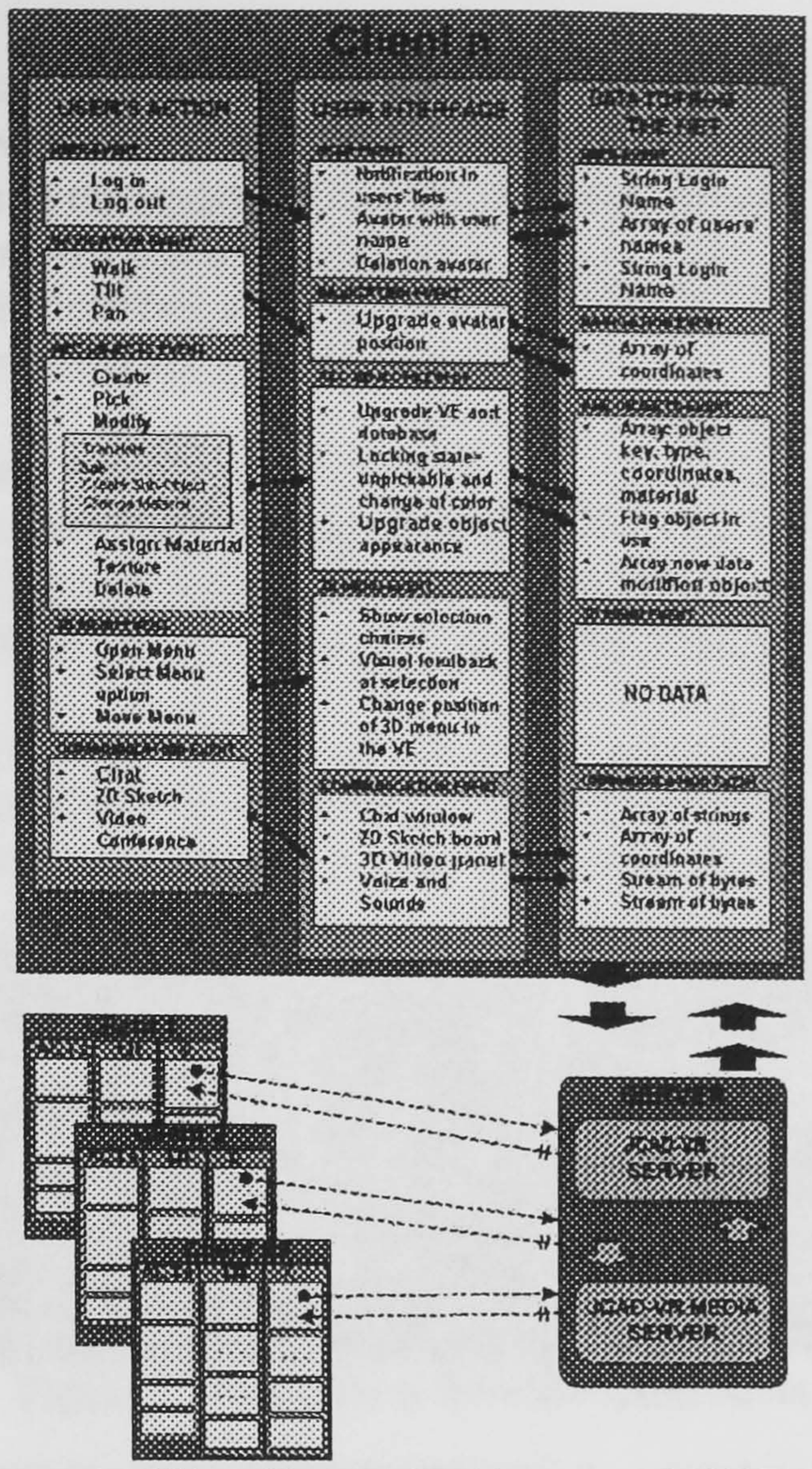


Figure 1 - Client/Server Architecture of JCAD-VR

To ensure communication between users, represented in the 3D world by avatars, different means are provided, from basic chat to voice and video conferencing. Freehand sketching in 2D is also possible through a shared electronic whiteboard. This architecture allows a real synchronous collaborative design making designing a true multi-user collaborative experience (Figure 2).

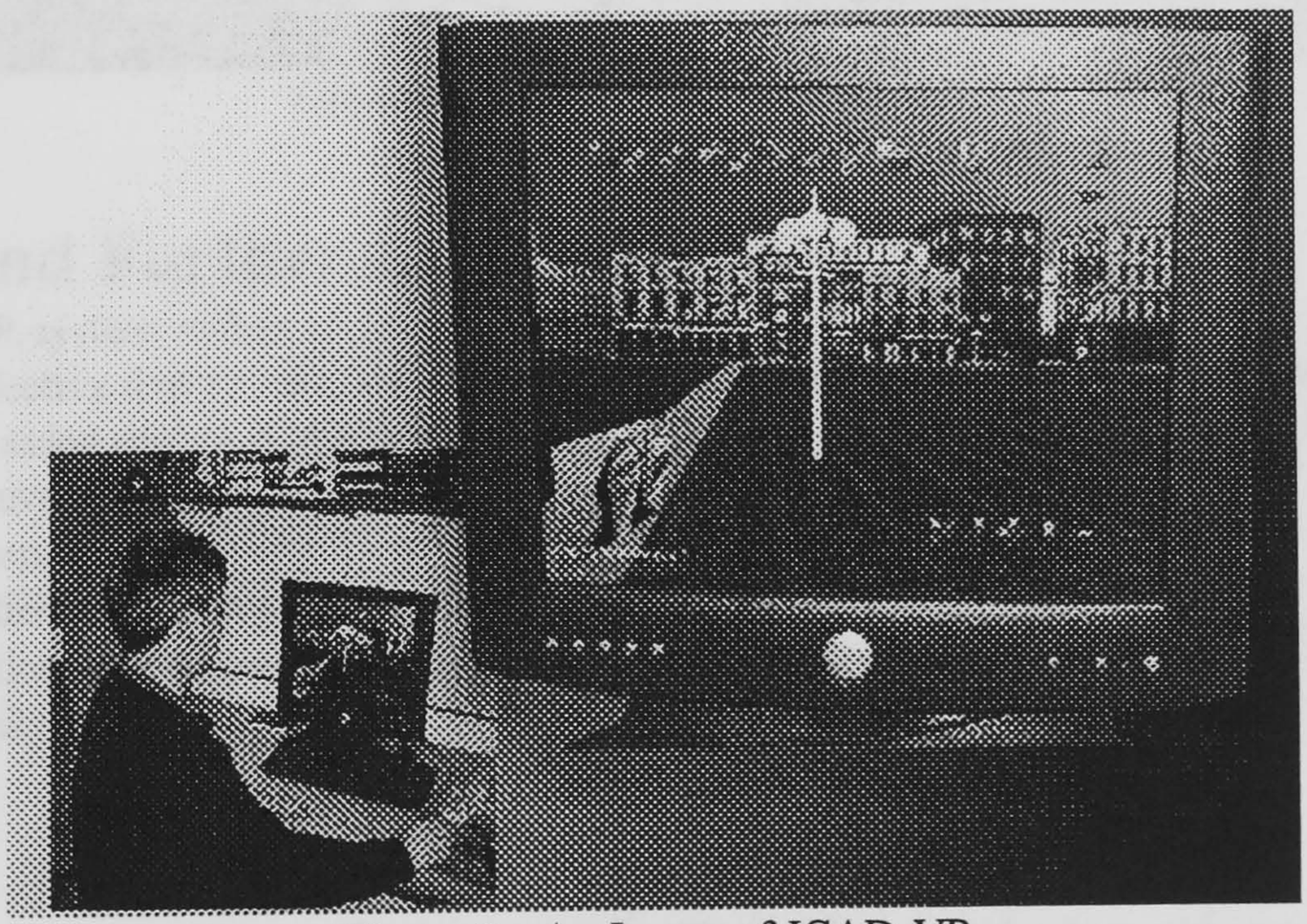


Figure 2 - An Image of JCAD-VR

Collaborative Experiment

The obvious first line of inquiry regarding the usability and usefulness of the emerging system was in the academic environment within which it had been created. For the past three years, the academic with overall responsibility for CAAD teaching had offered an optional class, to fourth year students (and to students from less senior years with exceptional commitment and skills in CAAD) in the design application of innovative VR technologies. In Session 2001/2 three of thebstudents taking the class were introduced to JCAD-VR and invited to put the system to its first serious test.

Students Christoph Ackermann, Ross Marshall and Edward Wright were located, each with an appropriate workstation, in three different areas within the Department of Architecture, with fixed and hand-held video cameras covering the actions and observations of the students. Over the two-hour design session, the three students were invited to design an information centre in a public square and in a given urban context of Glasgow. The introduction to the project and to the specifics of the interface to JCAD-VR lasted a mere 30 minutes. This meagre introduction was purposeful and intended to test how intuitive (or not) the system was.

The in-house experiment was a revelation to the authors of this paper. Over the two-hour design period there was:

- Fast and furious interaction amongst the three design participants within the common design environment; some 60/70 design scenarios were commonly generated, modified and agreed.
- Both satisfaction and frustration amongst the participants was noted regarding the high degree of mutuality in the interactive process.
- A real sense of having experienced a wholly immersive and shared design experience which heralds a future way of exploring and determining the configuration of the built environment.

Screen shots from the experiment are shown in Figure 3 and a frame from the video is shown in Figure 4.

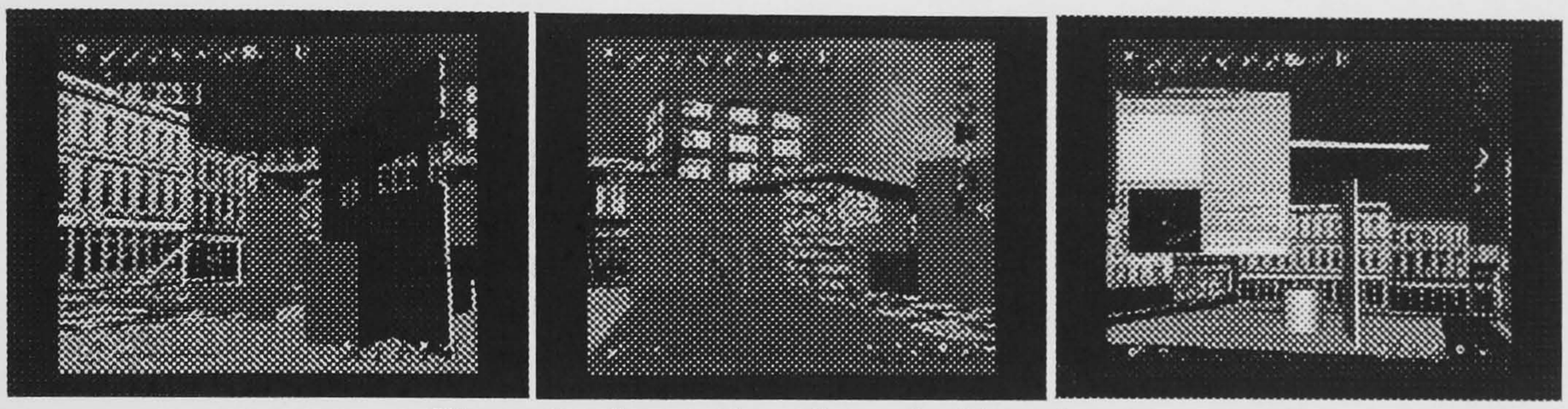


Figure 3 - Screenshots from the Experiment

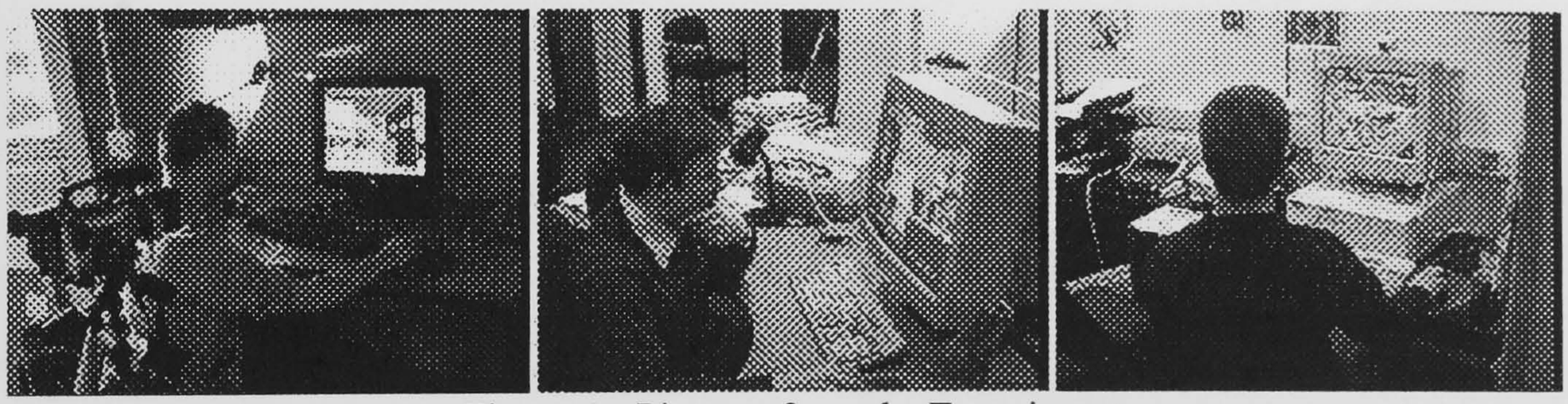


Figure 4 - Pictures from the Experiment

Conclusions and Further Developments

The prototype JCAD-VR system makes some steps toward the change of VR usage from mere presentation medium to a more powerful and effective design tool, and establishes the feasibility of VR becoming the interface for the next generation of computer aided design (CAAD) applications for architecture. Several enhancements are being considered for further development if the system including:

- A voice driven interface enhancing friendliness of the user interface
- Support for driving devices such as 6-degrees of freedom virtual glove
- Implementation of a multi-environments server capable of dealing with multiple VR environments simultaneously.

References

Cross, N. (1971). *Design Participation*, Proceedings of the Design Research Society, Academy Editions.

Barrilleaux, J. (2000). *3D User Interfaces with Java 3D, a guide to computer-human interaction in three dimensions*, Manning, Greenwich.

Bertol, D. (1997). *Designing Digital Spaces, An architect's guide to Virtual Reality*, John Wiley & Sons, Inc.

Conti, G., Ucelli, G. and Maver, T. (2001). "JCAD-VR - A Collaborative Design Tool in Virtual Reality", Proceedings of ECAADE 2001, Helsinki, pp. 454 – 459.

Dorta, T. and LaLande, P. (1998). "The impact of Virtual Reality on the design process", Proceedings from ACADIA '98 Conference, Quebec City, pp. 139 – 161.

Hughes, M., Hughes, C., Shoffner, M. and Winslow, M. (1997). *Java Network Programming*, Manning, Greenwich.

Lawson, B. (1990). *How designers think: the design process demystified*, University Press, Cambridge.

Sowizral, H. A. and Deering, M. F. (1999), "The Java 3D API and Virtual Reality", IEEE Computer Graphics and Applications, May/June 1999.
Petric, J., Ucelli, G. and Conti, G. (2001), "Educating the Virtual Architect", Proceedings of ECAADE 2001, Helsinki pp. 388 – 393.

Petric, J. Conti, G. and Ucelli, G. (2003). Designing Within Virtual Worlds. Accepted paper at the CAAD Futures 2003 Conference, Tainan, Taiwan.

Designing Within Virtual Worlds

Jelena PETRIC, Giuseppe CONTI and Giuliana UCELLI
ABACUS, Department of Architecture, University of Strathclyde, Glasgow, UK

Keywords: Distributed Design, Virtual Environment, Collaboration, Interface.

Abstract: This paper celebrates the successful outcome of a trial of an innovative multi-platform distributed design decision support system in which the shared design environment exists within the virtual world. The outcome is the result of a sustained three-year research and development effort, within an internationally recognised research group. The project set itself a number of ambitious targets within the broad spectrum of distributed design decision support, viz:

- A multi-platform environment: the trial demonstrates inter-operability of different machine platforms - from a home PC to an international standard Virtual Reality Centre.
- A distributed environment: the trial demonstrates the high level of understanding amongst the design team separated by time and space.
- An ability to propose, discuss and agree upon, design decision from **within** the virtual world. Hitherto, virtual environments were viewing galleries; designers had to leave them to effect design changes in a conventional CAD package. The trial described in the paper amply demonstrates the potential to design, collaboratively and, in distributed mode, from **within** the virtual world.

The two ideas upon which the system (known as JCAD-VR) is built are:

- that all the users present in the virtual world have to be able to share the same virtual environment in a "transparent fashion";
- the user interface, instead of the traditional menu/windows based layout, is part of the virtual world itself. Any element of the interface becomes an object belonging to the 3D world and therefore it is treated as any other object. Each element of the interface can then be moved or scaled according to the user's needs.

The entire project is based on client-server architecture where every user logs into a virtual world and starts sharing design tasks with other users.

The authors propose to present a video which demonstrates the positive outcome of the trials to date. More importantly, perhaps, the authors will put the achievements of the R+D into the context of past aspirations and developments in the subject area and, most importantly of all, suggest how these modest achievements will impact on the next decade of increasingly rapid R+D.

1 CONCEPT

Historically, architects have experienced the need to prove their design proposals using physical models. Due to the intrinsic spatial nature of the act of designing, even the most accurate 2D paper representation is usually not suitable to deliver the complexity of some architectonic ideas.

In the last decades the profession of the architect has been deeply affected by the *digital revolution* and the use of Computer Aided Architectural Design (CAAD) tools is nowadays part of the daily practice in most architecture firms. But in the last few years the "CAAD community" is experiencing a new revolution that is leading the move from static representation, based on 2D renderings or pre-recorded animations (considered as a sequence of 2D images), to dynamically generated 3D representations. Real-time navigation and interaction, typical of VR environments, provide just that fluent interface enabling the exploration of the design proposals that architects have not been able to get with any other media.

The increasing growth of computational resources and hardware power is probably preparing the anticipated transition to desktop VR applications, making them truly feasible tools for everyday practice. Furthermore, the recent growth of network-based virtual communities has brought a new level of complexity to the notion of virtual spaces, turning the profession of architect into something that might now resembles the one of the *virtual architect*.

Although VR is nowadays a quite mature technology, it is seldom used in architecture throughout the design process, but more often it is merely used as a powerful presentation technique.

Design methodologists in the past agreed on the need for iterative cycles between several phases of the design process. From studies concerning designers' behaviour many authors observed an indefinite number of return loops from the moment when gathering of information and structuring of the design problem take place (known as *analysis*) to the one when design solutions are generated (known as *synthesis*).

The use of Virtual Reality within the design process could give to the designer an appropriate quick and practical response to his/her need of iteration and search for design solutions. Moreover it enables the capture of more information than would be possible to capture with the use of the traditional media and makes the checking of the design solutions more efficient by enhancing simulation capabilities. Furthermore VR broadens the boundaries of traditional perception by providing experiences of worlds not necessarily real or material. In short it is the perfect simulation medium for architects investigating design solutions.

It is then highly predictable that in the near future VR will become the interface for the next generation of computer aided drawing (CAD) applications and we can anticipate the change of its use from a mere presentation medium to a more powerful and effective design tool.

CAD/CAAD packages are very powerful but often complex rendering tools, which were not meant to be investigation tools, and therefore generating 3D models is

often impractical and time-consuming. Therefore such models are usually employed when every design decision has been already taken.

Virtual environments (VE) are often created using CAAD packages for refinements, adjustments and exportation based on traditional 3D scenes. Documenting the evolution and development of design by constant updating of 3D models is an expensive business, and obviously even more expensive is the upgrading of the VEs generated from these models.

In these circumstances the use of VR would just increase costs. As result VR is relegated to the end of the design process rather then being used to engage design creativity through immersive design.

In the traditional scenario the decision making process does not take advantage of the technology but relegates the use of VR to the end of the process as a more convincing tool to impress contractors and clients. Only once the final solution has been achieved it is worth investing time into more powerful visualisation media.

Being aware of this background the research group engaged itself in the development of a VR system, named JCAD-VR, to help designers in the initial stages of the design process to take advantage of the VR as a new design tool.

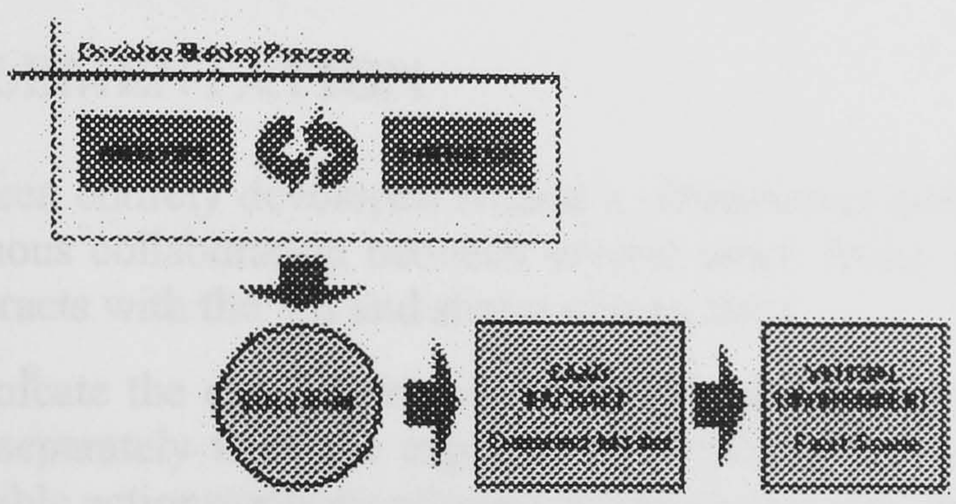


Figure 1 The current position of VR within the design process

JCAD-VR provides a flexible user-friendly immersive environment to support collaborative design on a synchronous base.

It can be thought of as an investigation tool that allows the designer to sketch freely 3D shapes within the virtual context. Moreover design solutions are shared in a synchronous fashion with other participants through the system's network-based architecture. Figure 2 shows the proposed scenario using JCAD-VR within the decision making process.

Here JCAD-VR provides the means for a more effective use of VR bridging between the phases of *analysis* and *synthesis*. VR is now employed at the very beginning of the decision making process when it is most likely to help in finding better design solutions. Once a desired solution is achieved the task of the creation of a very detailed 3D model and the final VR scene is given to appropriate CAAD packages.

Moreover the participants are able to investigate design solutions through concurrent

design and synchronous collaboration.

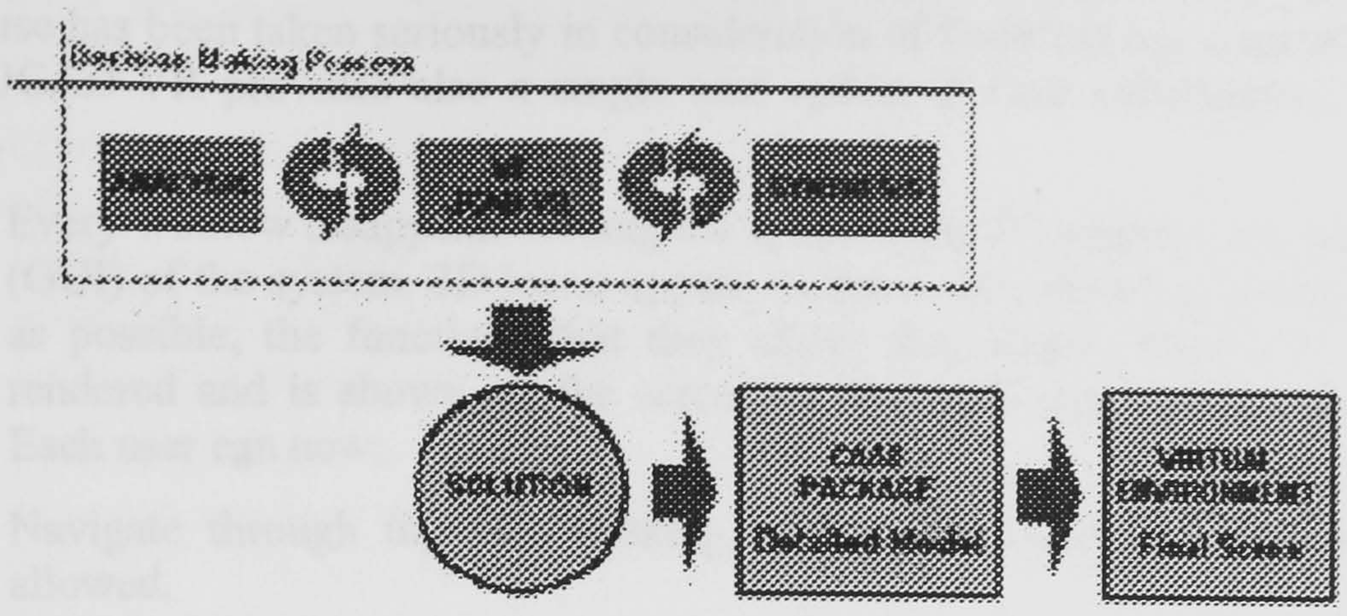


Figure 2 Proposed scenario

This paper reports the present state of the JCAD-VR system (Java™ based Collaborative Architectural Design tool in VR) and will highlight its future development and testing.

2 IMPLEMENTATION

The system has been entirely developed around a client-server architecture to allow constant synchronous collaboration between several users. Every user accesses the virtual world, interacts with the VE and shares design tasks.

To better communicate the capabilities of JCAD-VR, the explanation of the client side is presented separately from the explanation of the server. As if in a software demo, all the possible actions/options allowed by the system will be identified.

2.1 Client Package

1) On initiating JCAD-VR, the user is asked for a login name for the session and through an options panel he/she can:

- Decide which server to connect to and through which server port (in case the software is running on a local server).
- Decide if running in single or multiple screen mode. Single mode is set in case the display device is a normal screen; the multiple screen option has been included to allow devices such as the multi-projector display system processing the visual output of a Reality Centre.
- Load a VRML 97 file as a VR scene, if any.
- Activate or de-activate video conferencing facilities for the session (in case video conferencing facilities have been activated, support for video

capturing device recognition and checking are provided).

Providing the information required for this initial window is quite straight forward. Ease of use has been taken seriously in consideration of those not accustomed to the system. JCAD-VR provides also a single user option in case collaboration is not required.

2) Every window disappears freeing the space to the 3D graphic user interface (GUI) of the system. 3D icons appear on the screen showing, as intuitively as possible, the functions that they allow; the chosen VRML 97 file is rendered and is shown on the screen as in a traditional VRML browser. Each user can now:

- Navigate through the VE: walking, tilting and panning movements are allowed.
- Observe other participants' movements through their 3D avatars in the VE. Avatars are made distinguishable from each other by having, nearby, a 3D text of the users' login name.
- Listen to the other participants' voices through the loudspeakers.
- Watch, in the virtual scene, the 3D panels showing the video captured and sent by other participants. Every 3D panel shows the participants' login name to make them clearly distinguishable. These 3D panels are intrinsic parts of the GUI of JCAD-VR. Instead of being conventional windows they are 3D entities within the VE. These panels, as well as all the elements of the 3D GUI, are moveable for the convenience of the user.
- Check in a monitor the local user's captured video that is streamed out to the server.
- Check the list of users.
- Chat with other participants. This option is provided to assure a certain degree of communication between the participants in case the video conferencing facility is de-activated; this is available even when the video conferencing has been set as active.
- Freehand sketch in 2D on a shared electronic whiteboard, the possibility to set colours is provided in order to ease distinction between participants' contributions.
- Create 3D shapes and 3D AEC objects: both geometric primitives (cones, boxes, spheres etc.) and architectural entities (walls, doors, windows etc.) are available. A "3D ruler" is provided to help the user in constructing objects.

The choice of a 3D GUI was made in response to the possibility that modules supporting several display devices such as CAVEs and head-mounted displays would be included. Not having a traditional windows/menus user interface JCAD-VR can be used freely in every display situation minimising the effort to customise it for each device.

3) The 3D engine of the system renders all the possible changes of the VE: movements of avatars, video conferencing streams on the 3D panels and, most importantly, creation and modification of objects created within JCAD-VR. Each user, concerning 3D objects, can:

- Observe every object created in the VE both those created locally and by other participants. The system upgrades the VE with the objects created by all the participants in a synchronous fashion. An identification routine is provided in order to give each object a unique ID number to avoid interference in each users' local database of objects.

- Pick every *active* object in the virtual scene. *Active* objects here are all the objects created within JCAD-VR and not geometry imported with the VRML 97 file. Objects originally part of the VRML 97 file are considered to be *passive* and are not pickable. Once an object is selected:

  a) User priority on selected objects is set by a distributed locking mechanism. Locked objects will be no longer pickable for other participants until unlocked and their locked status becomes apparent through change into a red colour.

  b) 3D icons, 3D panels with general dimensions and x, y ,z arrows are set visible to help the user operate on the object.

  c) Translation, rotation and scaling in every direction are allowed. These modifications are operated on the object by simple dragging of the arrow representing the x, y or z axis. A 3D ruler and a 3D panel provide simple visual feedback to check the modified dimensions. Some routines to constrain modifications allowed on AEC objects are included, for example a door cannot be moved onto or too close to a second door and vice versa.

  d) Change of material is supported. In the first instance objects are created with a default grey colour but a library of textures is provided.

  e) Deletion of the object is allowed.

- Observe the visual feedback of the locking engine mechanism in case one of the participants has selected an object.

- Observe upgrading of the VE in case any modification on one or more objects has been carried out by any participant.

- Check for information on all *active* objects in the VE through a local database of objects. Objects are divided by type and general geometrical information is provided such as length, width, height, volume, radius, material etc. The system notifies to every users' internal database any creation or modification of geometric objects within the virtual scene and broadcasts their numerical information.

Figure 3 shows, that for every action performed by the user, the consequential visual

feedback and data as sent/received to/from the server through the network.

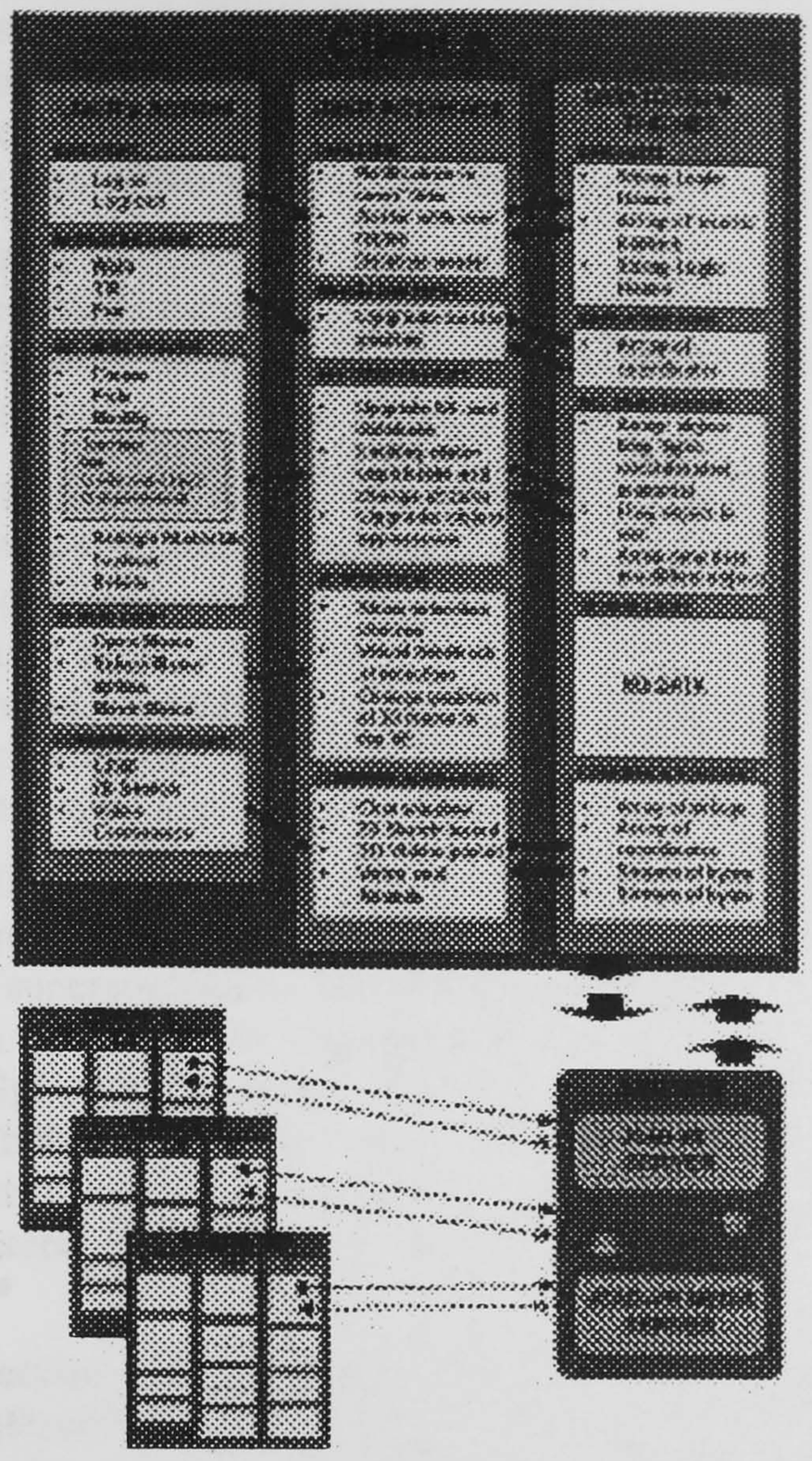


Figure 3 Client/Server Architecture of JCAD-VR

2.2 Server Package

The server is made of two parts: *JCAD-VR Server* which looks after the VE information to be broadcast, and *JCAD-VR Media server* which takes care of media streams and video conferencing tools. Both parts constitute the server system and they are closely linked to each other.

As an independent part of the framework the server has an autonomous and simpler interface that provides primarily information about the network status and transfer rate. A number of components are envisaged such as the communication status, the users on line, VR shared environments and settings and size of video conferencing windows. Since the clients are communicating through independent processes, a future enhancement will allow the server to deal with several VR environments

simultaneously.

The intrinsic multiplatform nature of JCAD-VR, inherited from the language used to code it, allows the server to transmit data to a broad range of platforms, from normal PCs to the supercomputer running a Reality Centre, and leaves the research team the freedom to test the software with several operating systems. The communication channel ensures the link between server and clients through a TCP/IP network.

The whole framework of JCAD-VR was organised to allow concurrent software development, in a modular fashion, by individual members of the R+D team (Conti et al. 2001). To facilitate this, an object oriented approach was identified as the most suitable one and the entire system was coded in Java™.

The choice, even if less efficient in term of performances if compared with some other languages, indeed offered great flexibility, true scalability and, last but not least, fully multi-platform support. Java3D™ was used to code the GUI and everything concerning the VE. Its network-centric nature, its multimedia integration together with the use of native hardware acceleration (OpenGL) and multi-processor support (in the case of Sgi workstation) make it the obvious choice for the development of a real-time multimedia collaborative system.

The client application, in response to the obvious hardware limits imposed by the use of different hardware, has been written to be easily customised to run on PCs as well as on a Sgi supercomputer. The former are normal PCs whose video-card displays the virtual world only on a traditional window or at full screen. The latter is a 12-processors Sgi Onyx2 system running the Reality Centre at ABACUS, University of Strathclyde, Glasgow. When JCAD-VR is launched on the Sgi it can take advantage of its computational power to stretch itself on a 5 metre wide 2 metre high tassellated screen where 3 Barco projectors create a 160 degree panoramic image.

The internal architecture of JCAD-VR is such that modules might be easily adapted to allow use of different VR devices such as CAVEs or Headmounted Displays as well as several pointing devices such as a joystick, 3D mouse and VR Gloves.

From the collaborative point of view JCAD-VR is highly scalable and several communication media options are provided depending on the hardware limitations of the computer on which it is running.

The video conferencing facility has been coded using the Java™ Media Framework (JMF) which enables cross-platform capture, playback and streaming of audio and video at different transfer rate and resolutions. A great deal of effort has been expended by the research group to integrate closely the two sections of JCAD-VR: the 3D module with the multimedia module.

Figure 4 An Image of JCAD-VR

3 TRIALS OF THE SYSTEM

The obvious first line of inquiry regarding the usability and usefulness of the emerging system was in the academic environment within which it had been created. For the past three years, the academic with overall responsibility for CAAD teaching had offered an optional class to fourth year students (and to students from less senior years with exceptional commitment and skills in CAAD) in the design application of innovative VR technologies. In Session 2001/2 three of the students taking the class were introduced to JCAD-VR and invited to put the system to its first serious test.

Students Christoph Ackermann, Ross Marshall and Edward Wright were located, each with an appropriate workstation, in three different areas within the Department of Architecture, with fixed and hand-held video cameras covering the actions and observations of the students. Over the two-hour design session, the three students were invited to design an information centre in a public square and in a given urban context of Glasgow. The introduction to the project and to the specifics of the interface to JCAD-VR lasted a mere 30 minutes. This meagre introduction was purposeful and intended to test how intuitive (or not) the system was. The in-house experiment was rewarding to the authors of this paper. Over the two-hour design period there was:

- Fast and furious interaction amongst the three design participants within the common design environment; some 60/70 design scenarios were commonly generated, modified and agreed.
- Both satisfaction and frustration amongst the participants was noted regarding the high degree of mutuality in the interactive process.
- A real sense of having experienced a wholly immersive and shared design experience which heralds a future way of exploring and determining the configuration of the built environment.

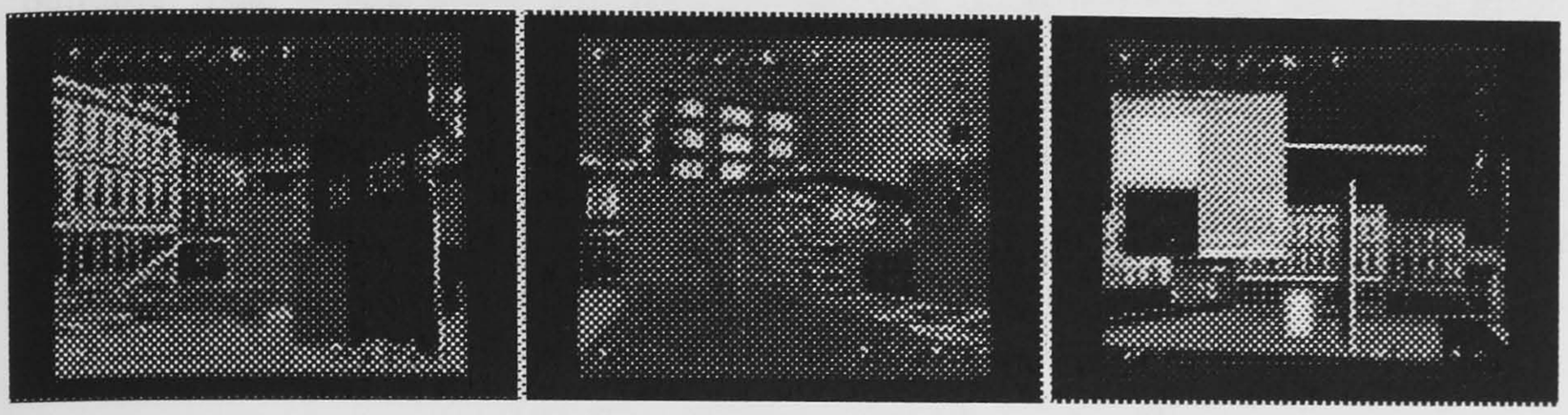


Figure 5 Screenshots from the Experiment

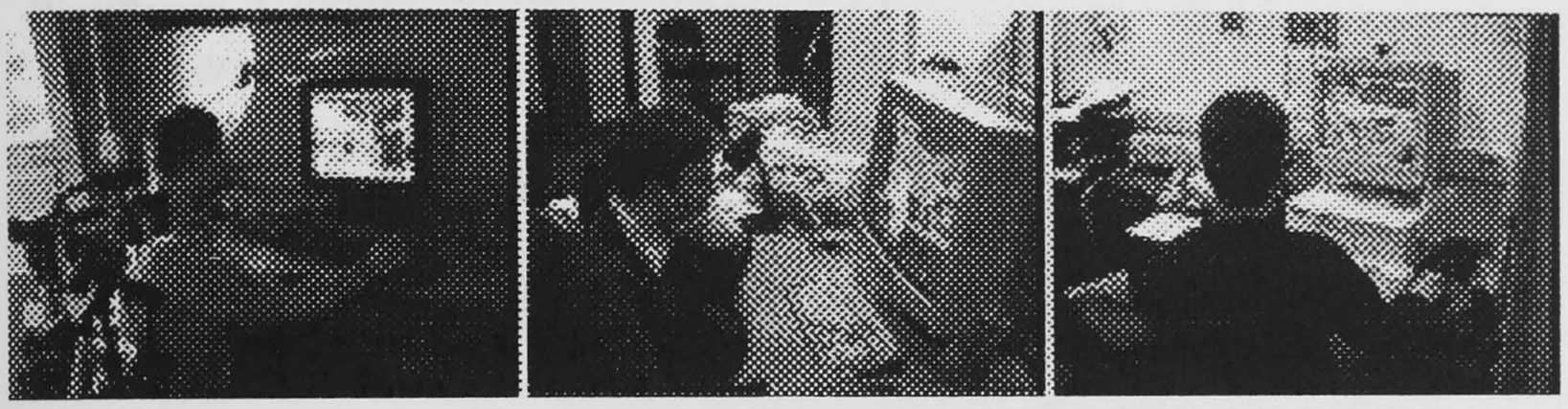


Figure 6 Pictures from the Experiment

4 CONCLUSIONS AND FUTURE DEVELOPMENTS

Assuming the availability of resources, it is intended to conduct further trials and development of the system.

Further trials will be trans-institutional and trans-national and the version of the system will include the recently implemented video-conferencing facility.

Further developments will include:

- A voice driven interface enhancing friendliness of the user interface;
- Support for driving devices such as 6-degrees of freedom virtual glove;

- Implementation of a multi-environments server capable of dealing with several VR environments simultaneously.

As in the earliest days of the introduction of computers into architectural design, the quantum jump is made by students. The work reported here, and which will be shown during the conference is, we believe, the epitome - in the current state of the art - of excellent practice. It makes a breakthrough, we believe, in the evolution of good design ideas and offers a real prospect for user participation.

There is some way to go, of course, to *design* interactively in a virtual environment. The next step which we envisage is to link to the 3D model the emerging and sophisticated software for the thermal, lighting and acoustics properties of the building. This would allow the user to visualise, dynamically, airflow, temperature gradients, lighting levels and to experience the actual acoustic characteristics of the space as she/he moves through it.

The other exciting development is for representatives of the client/user group to "join" the designer within the virtual environment and to participate directly in the evolution of the design concept.

BIBLIOGRAPHY

Barrilleaux, J. (2000). *3D User Interfaces with Java 3D, a guide to computer-human interaction in three dimensions,* Manning, Greenwich.

Bertol, D. (1997). *Designing Digital Spaces, An architect's guide to Virtual Reality*, John Wiley & Sons, Inc.

Conti, G., Ucelli, G. and Maver, T. (2001). JCAD-VR - A Collaborative Design Tool in Virtual Reality, *Proceedings of ECAADE 2001*, Helsinki, pp. 454 – 459.

Dorta, T. and LaLande, P. (1998). The impact of Virtual Reality on the design process, *Proceedings from ACADIA '98 Conference*, Quebec City, pp. 139 – 161.

Hughes, M., Hughes, C., Shoffner, M. and Winslow, M. (1997). *Java Network Programming*, Manning, Greenwich.

Lawson, B. (1990). *How designers think: the design process demystified*, University Press, Cambridge.

Petric, J., Ucelli, G. and Conti, G. (2001). Educating the Virtual Architect, *Proceedings of ECAADE 2001*, Helsinki pp. 388 – 393.