University of Nevada, Reno

CAVEMANDER: An Approach and Software Platform for Building Command and Control Applications in CAVE

submitted in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy in
Computer Science and Engineering

by

Sermsak Buntha

Dr. Sergiu M. Dascalu/Dissertation Advisor

August, 2009

THE GRADUATE SCHOOL

We recommend that the dissertation
prepared under our supervision by

SERMSAK BUNTHA

entitled

Cavemander: An Approach And Software Platform For Building Command And Control Applications In Cave

be accepted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

Dr. Sergiu Dascalu, Advisor

Dr. Wendy Calvin, Committee Member

Dr. Frederick C. Harris, Jr., Committee Member

Dr. Eelke Folmer, Committee Member

Dr. Janet Usinger, Graduate School Representative

Marsha H. Read, Ph. D., Associate Dean, Graduate School

August, 2009

Abstract

Command and control systems play a vital role in displaying information about various operational situations, thus helping decision makers to thoroughly understand them and make related decisions in a timely and correct manner. In military operations, pictures of such situations have been traditionally displayed on large tactical boards and/or vertical maps. Although for several decades computers have been used to replace these traditional displays, the pictures of the situations have largely been presented on 2D media only, such as on PC monitors or wall screens. Due to several recognized advantages of 3D visualizations, combined with the power of immersion in virtual worlds, we believe that in command and control applications 3D immersive environments such as the CAVE Automatic Virtual Environment (in short, CAVE) could significantly improve the understanding of the situation and the decision-making performance of the commanders. This dissertation investigates existing solutions for developing software for CAVE and proposes a new approach and related toolset for creating command and control applications in CAVE. CAVEMANDER, the proposed approach and its related supporting software tools, is aimed at improving the development of command and control applications for CAVE and consists of a new software engineering method encompassing the necessary construction steps and related artifacts as well as a set of software resources (a software platform) composed largely of reusable code. CAVEMANDER's advantages are demonstrated on a command and control application in the area of military training, an application instantiated in scenarios that can test the

trainees skills for command and control, including planning and testing, information interpretation, and failure investigation. Although command and control systems are mainly associated with military applications, future research can expand the scope of CAVEMANDER to support applications in other domains, such as fighting wildfires, conducting search and rescue missions, and coordinating planet expeditions. In summary, the work described here is aimed at filling the existing research and development gap in building CAVE-based command and control applications. Future work could encompass new simulation applications, enhanced features and new software components, usability studies, and automated code generation.

Dedication

To my parents Sumran and Pornthip,

my wife Pawinee,

my daughter Chanikan, and my son Saharat

Acknowledgments

First and foremost I would like to thank my advisor Dr. Sergiu Dascalu for his excellent guidance, support, encouragement, and patience. I would also like to thank the members of my PhD dissertation committee, Dr. Wendy Calvin, Dr. Janet Usinger, Dr. Frederick C. Harris, Jr. and Dr. Eelke Folmer for kindly agreeing to serve on my committee and providing invaluable guidance and feedback on my work.

I would also like to thank the Royal Thai Navy for the financial support over the years without which I could have never been a graduate student at a U.S. university, first in a Masters and then in a PhD program. I am also very grateful to NASA and the Nevada System of Higher Education, that partially supported my work. Specifically, support was provided by NASA grant # NNX07AT65A via a sub-award and with cost share provided by the Nevada System of Higher Education: NSHE-08-51 and NSHE-08-52.

I would also like to thank Mr. William R. Sherman for providing an excellent course in virtual reality and making available the open-source VR library, FreeVR, to develop CAVE applications.

My thanks also go to all the authors who kindly gave me permissions to use figures from their published work.

I am also much indebted to my dear friends and colleagues Sohei Okomoto and Muhanna Muhanna, PhD candidates in the Computer Science and Engineering at the University of Nevada, Reno for their sustained encouragement, support, and fruitful exchange of ideas throughout my graduate years. I would also like to thank Roger Hoang, also a PhD student at UNR, for providing technical knowledge and support for running our software in CAVE.

Furthermore, I would like to express my deepest gratitude and thanks to my parents, Capt. Sumran Buntha, RTN (my father) and Mrs. Pornthip Buntha (my mother), who instilled in me the desire to explore, learn, work hard and excel, and who guided me through life with endless care and wisdom.

Last, but not least, I would like to thank my wife, Mrs. Pawinee Buntha, for her extraordinary support, understanding, care, and unconditional love.

Table of Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

API	Application Programming Interface
C&C	Command and Control
C4I	Command, Control, Communications, Computers, and Intelligence
CAVE	Cave Automatic Virtual Environment
CCRP	Command and Control Research Program
COTS	Commercial off-the-shelf
DoD	US Department of Defense
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HMD	Head Mounted Device
I/O	Input Output
ICS	Incident Command System
PC	Personal Computer
SA	Situation Awareness
SE	Software Engineering
UML	Unified Modeling Language
UNR	University of Nevada, Reno
VE	Virtual Environment
VPL	Virtual Programming Languages
VR	Virtual Reality

Chapter 1

INTRODUCTION

In this first chapter of our dissertation the motivations of our work are explained, an overview of the proposed solution is provided, and the organization and the contents of the remaining chapters of the dissertation are outlined. The main terms used in describing the research and development work done for this dissertation are also introduced.

1.1 Problem Motivation

In **Command and Control (C&C)**, which is an interactive approach for commanders to understand and make proper decisions in a timely manner, pictures of C&C situations often play a very important role during the performing of various C&C tasks. Traditionally, pictures of the situations have been displayed on large tactical glass boards or vertical paper charts while currently computers are becoming the main components in C&C systems. Although there have been already more than three decades since computers have been used for enhancing the functionality of C&C systems, the pictures of the C&C situations are still

mostly displayed on 2D media such as PC monitors or wall projectors. This raises a number of problems, stemming from the limited capabilities of 2D displays, for example difficulties in understanding and assessing geometrical parameters such as slope and elevation in the surrounding terrain, delays in interpreting various data presentations and visualizations, and reduced situation awareness when single or multiple 2D displays are used.

Due to several recognized advantages of 3D visualizations (including higher realism and richness of representations, as well as enhanced visualization capabilities), combined with the power of immersion in virtual worlds, we believe that in C&C applications 3D immersive environments such as the Cave Automatic Virtual Environment [1] (in short, **CAVE**) could significantly improve the understanding of C&C situations and the decision-making performance of the commanders. Thus, this dissertation is focused on providing solutions for improving the utilization of the CAVE in C&C applications, particularly in applications for the military domain.

There are several motivations for embarking on such topic. First, we have aimed at advancing the state-of-art in developing C&C simulations for CAVE, an area of tremendous potential and practically endless possibilities. CAVE is one of the most modern environments for experiencing and integrating new software engineering (SE) and human-computer interaction (HCI) solutions, as well as a computer-based technology still in a vigorous and rapid process of development and advancement. Second, we have focused on filling the existing gap in software development methods designed specifically for building CAVE-based applications. To the best of our knowledge, the landscape in terms of software engineering methods and techniques tailored for CAVE developments is currently quite bleak. Third, we have intended to contribute with new software resources (tools and reusable code) to the currently limited pool of such resources for supporting CAVE-based software development. Fourth, we have aimed at providing a method- and toolset-based foundation for future re-

[1] CAVE is typically considered a recursive abbreviation, as indicated above. However, a non-recursive interpretation of the abbreviation also used, albeit less frequently, is Computer Automated Virtual Environment

search and development in C&C in CAVE, which can include a greater emphasis on HCI (in particular, on usability studies) and can be extended to applications outside the military domain. Fifth, we have attempted to build on our previous experience with C&C, in particular with developing and using software tools for C&C, which includes working as an officer in the Royal Thai Navy and completing a Master thesis [18] at UNR that had proposed a new software environment with a 2D interface for C&C in naval surface operations.

1.2 Solution Overview

To accomplish the goals of this dissertation, focused on providing solutions for improved utilization of the CAVE in C&C applications, the following are proposed: (i) A new, original software construction method specifically designed for developing CAVE-based applications, a method consisting of procedural steps and software artifacts produced in these steps; (ii) A set of software resources (a "software platform") that includes several reusable server and client software components, including a graphical user interface (GUI) wizard, a scene generator, a playback module, and a CAVE interaction component, as well as a reusable library of code and application programming interface (API); and (iii) Examples of a C&C simulation application and scenarios built using our proposed method and tools, that show the utility of our set of solutions.

Encompassing all the above, we describe in this dissertation CAVEMANDER, whose name comes from the combination of the keywords "CAVE" and "Commander", and which can be seen as both an approach (an SE method) and a specialized software platform (set of software resources) aimed at supporting and expediting the software development process based on a suite of related tools.

A major expected benefit of using CAVEMANDER is to capitalize on reusable code. Built

using FreeVR [85] for its client side component, CAVEMANDER supports landscape-based C&C applications. It is centered on a set of programming libraries as well as on customizable software components such as a terrain and a set of simulation entities (e.g., ships or tanks). Functions of C&C applications include various built-in capabilities to facilitate the work and the decision-making of the commander. These capabilities allow for example watching a C&C situation in various dimensional scales and perspectives, travelling around the area of interest, estimating the potential displacement of each movable element included in the simulation, showing historical trails, determining surface among markers and space distance between points, and so forth.

At this time, CAVEMANDER is primarily intended for C&C military simulation applications, particularly for training that includes acquiring C&C skills, developing and testing strategies, and investigating failure causes of a situation. In this dissertation, the CAVE is used for the presentation of various C&C situations and for the interaction of the commander with these situations' entities. This allows decision makers to step closer into a more accurate and authentic C&C environment. By giving the decision makers the ability to immerse into a C&C situation, it is expected that they will acquire a better understanding of the situation and will achieve a better performance.

Although C&C systems are generally recognized as military systems, our research and development will allow expanding the scope of C&C systems into various other areas, such as fighting wild fires, conducting search and rescue missions, planning and executing planet expeditions, and more. Furthermore, the work presented in this dissertation, focused so far exclusively on simulation, can also constitute the basis for actual real-world real-time C&C applications with the commander following situation developments and taking control of them from inside the CAVE.

1.3 Dissertation Organization

This dissertation, in its remaining chapters, is organized as follows: Chapter 2 provides general background information on C&C systems, highlights problems with current 2D display media devices for C&C applications, and overviews several newly proposed solutions to address these problems. Chapter 3 presents an overview of Virtual Reality (VR) and then delves into CAVEs, which are the most representative devices for *immersive VR*. This chapter offers details of CAVE devices, surveys a number of current CAVE-based applications, and points to some areas of future CAVE-based software developments. Chapter 4 details the proposed SE method, emphasizing the activities included and the software artifacts produced in each activity. Chapter 5 provides a comprehensive description of the CAVEMANDER software platform, consisting primarily of server software components (including a GUI wizard that helps the user define simulation scenes and create applications scenarios) and client software components (mainly for C&C situation presentation and interaction with the simulation in CAVE). The overall architecture of the CAVEMANDER software as well as details of the available software resources are also provided here. Chapter 6 contains an example of a military C&C application and illustrates via a sequence of detailed snapshots the entire process supported by CAVEMENDER, including defining the application scene, building the simulation, creating a scenario, and running the scenario. Chapter 7 surveys related work and places the material presented in this dissertation in the context of related SE research and development efforts for building CAVE software applications. Chapter 8 points to several directions of future work and Chapter 9 finalizes the dissertation with a review of its main findings and experiences and a summary of its main contributions.

Chapter 2

BACKGROUND: DISPLAY SYSTEMS FOR C&C

Because we have chosen CAVE as a new, modern solution for displaying information in C&C applications, in this chapter we take a brief look at the defining characteristics of C&C and survey traditional methods for displaying information on C&C applications, particularly at the use of computers and their 2D interfaces as primary means for assisting the work of commanders in military applications. Based on the search of related literature, we also discuss the issues and limitations of the traditional 2D media devices for C&C and then overview a number of new solutions for displaying information and interacting with the computer systems in modern C&C centers and environments. Later, in Chapter 3 we provide details of what virtual reality means, including strengths and limitations, and discuss a number of characteristics and areas of applications that support our belief that CAVE is a suitable and powerful solution for supporting C&C applications.

2.1 Brief Overview of Command and Control

According to the DoD Dictionary of Military and Associated Terms, C&C is defined as

> The exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission. Command and control functions are performed through an arrangement of personnel, equipment, communications, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the mission [49].

Although C&C has been used since the beginning of human civilization, the efficiency of C&C has become increasingly more important in our current information age [41] [67]. In particular, three main advanced technologies (Communications, Computers, and Intelligence-gathering Systems) provide tremendous support for C&C activities. In general, the combination of C&C specific methods and their associated supporting technology is called C4I (Command, Control, Communications, Computers, and Intelligence), as depicted in Figure 2.1 [68]. The US government has designated the Command and Control Research Program (CCRP) to particularly focus on this area [93].

One of the C&C major characteristics is that it requires situation awareness, which can be achieved through quick and proper information gathering and reliable communication. Various efforts in computer science research have been directed towards exploring new ways of increasing the availability of data from the battlefield and the commander's awareness of the ongoing military situation in the area under his or her responsibility. It is worth nothing that, however, various methods view and evaluate situation awareness in different ways [81].

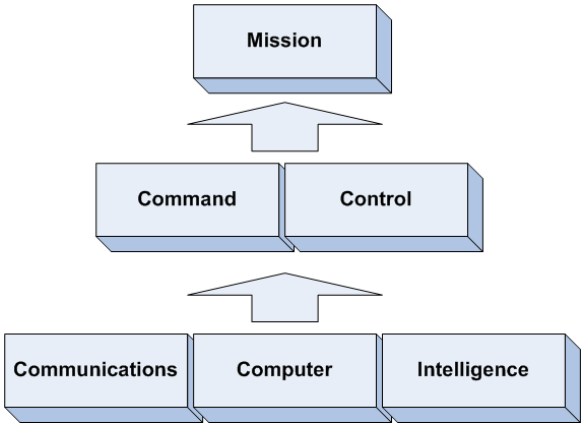


Figure 2.1: C4I = C&C + Communications + Computers + Intelligence

Although C&C was originally intended and has been widely recognized as a military activity, it is also very useful to support certain civil operations [98]. For example, to provide quick and efficient responses to emergency situations the Office of Emergency Preparedness (OEP) has worked on developing both theoretical and practical methods as well as suitable technologies for C&C activities [94]. The Incident Command System (ICS) [34], whose organization and process are depicted, respectively, in Figures 2.2 and 2.3, is also one of the C&C-oriented systems aimed at responding to emergency situations. As shown by the surveyed literature, ICS was widely used in U.S. emergency management organization [12] [77].

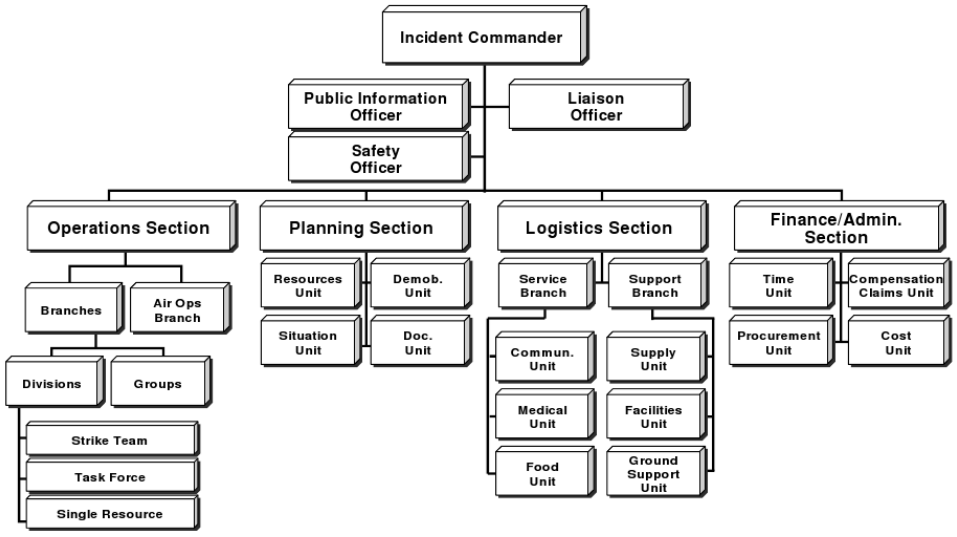


Figure 2.2: ICS Organization [34] [102]

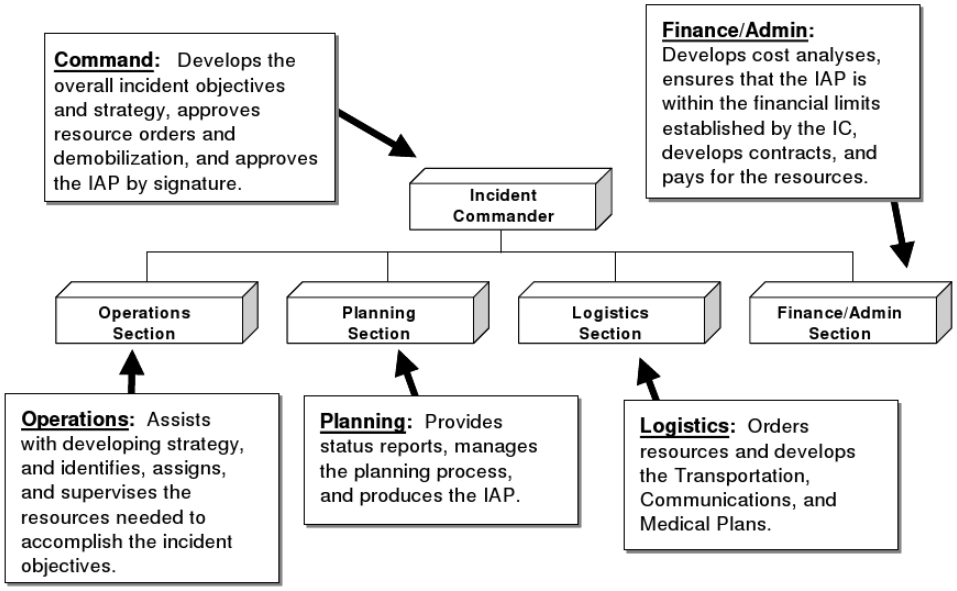


Figure 2.3: ICS Process [34] [101]

The C&C approach can also be applied to protect natural environments and their resources. For example, in this direction of using C&C in non-military domains [43], describes a method that employs the C&C approach for natural resource management. Along the same lines of work, Michael Athans proposed a C&C method for control science research [5]. C&C systems can also play an important role in space and planet expedition. Recent work has produced the C&C software called Maestro, which in essence is part of a science activity planner system developed in Java [91] for Mars exploration rover operations [65] [66] [100].

Furthermore, developed using the open source Java-oriented environment Eclipse and including Maestro among other resources, Ensemble is a software plug-in platform built for supporting various planet exploration activities [2].

2.2 Traditional 2D Displays for Command and Control

C&C systems play a vital role in displaying information about operational situations such that decision makers can thoroughly understand them and be able to take appropriate decisions in a timely manner. In essence, C&C is an interactive approach to deal with the situations that need to be kept under control. C&C systems have been used for long time, and since its early development the computer has become the main component of such situations.

However, despite many decades of using computers in C&C, the pictures of the operational situations have been usually displayed only on 2D screens, such as computer monitors and wall-mounted screens, as shown in Figure 2.4, or large tactical boards and charts. In general, the basic set of requirements for C&C display systems consists of the following:

Figure 2.4: ISS Flight Control Room [63] [103]

- Display the situation on real-time
- Present an all-encompassing overview picture

- Make sure that all critical pieces are shown
- Provide various viewpoints to the commander

Interestingly, older methods, based on using large paper navigation maps as well as acetate overlays and tactical boards for pencil annotations are still being used in various parts of the world to display the summary image of an operational situation. Unfortunately, this traditional approach is often inconvenient and can lead to late results, which are no longer useful for commanders. Our prior work, presented in [18] [19] [29] and [28], was intended to contribute to addressing this problem by providing an effective software solution for C&C in naval task forces for surface warfare training. Using a systematic software engineering process [88] and the Unified Modeling Language (UML) [4] as major development devices, we have designed and implemented a prototype of C&C software tool that runs on Linux and can be used for training and simulation. As pointed out in [29], we enhanced this system with Human-Computer Interaction (HCI) features that facilitated software usage and allowed rapid access to information, thus efficiently supporting commanders in their decision making process. This software tool was designed to be cost-effective by using Commercial Off-the-Shelf (COTS) technology such as regular personal computers. The intended users of the proposed software were fleet commanders, commanding officers (ship captains), and navy leaders in charge of commanders and commanding officers.

The tool consisted of a main 2D interface (Figures 2.5) through which all the functions and features of the software were accessible. More details are provided in [18] [19] [29] and [28], but it is worth noting that the software allowed effective simulation for maneuvering of our own ship (Figure 2.6), planning and executing interception of an enemy ship (Figure 2.7) and calculating and following an escape route out of territory dominated by enemy vessels (Figure 2.8 – 2.10). Among smaller but useful features and facilities included, we can mention the option for the user to select the screen background (black or white, with the opposite color used for the foreground text) or the possibility to turn on or off various screen

markers and indicators such as the compass. These functions and features allowed the use of the tool in various environments and under various lighting conditions. This prior work of ours is relevant not only in that it allowed a useful investigation of requirements and challenges associated to the development of 2D C&C software, but also because it triggered the work described in the present dissertation. In essence, we have been motivated by this prior work of ours to go beyond the scope of 2D-interface C&C software and explore and take advantage of the possibilities made available by the modern 3D technology. In our more recent work, we aimed at investigating modalities of using the 3D technology, in particular the immersive VR paradigm, in modern C&C applications.

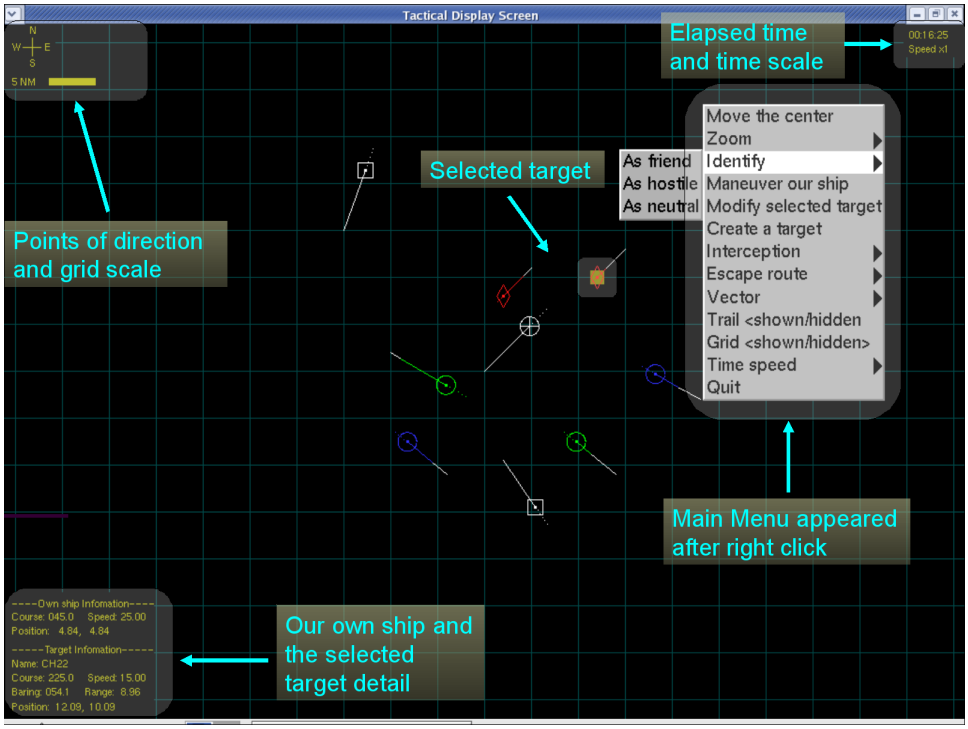


Figure 2.5: Our Prior 2D Software Tool for Naval Surface Warfare Simulation and Training [18]

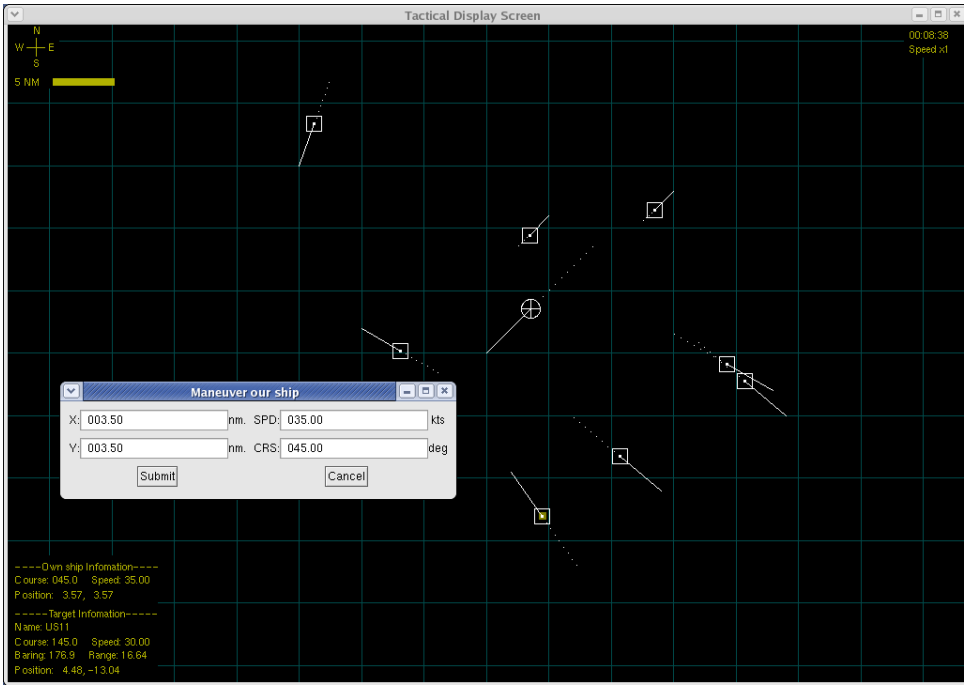


Figure 2.6: Prior 2D C&C Software Tool — Panel for the Maneuver Our Ship Function [18]

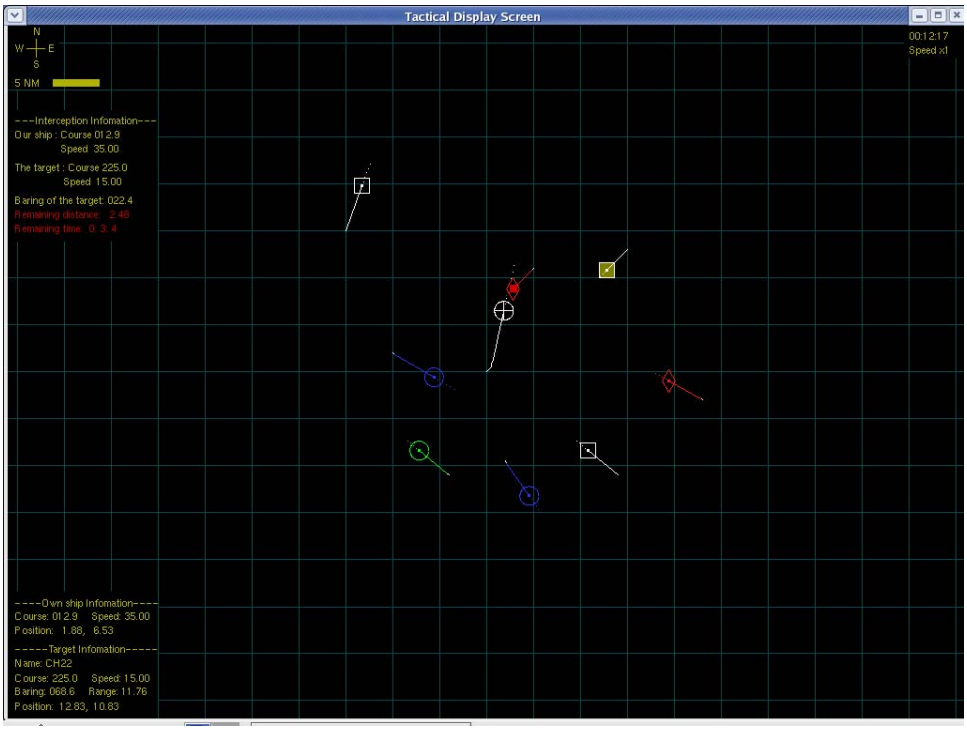


Figure 2.7: Prior 2D C&C Software Tool — Example of Target Interception [18]

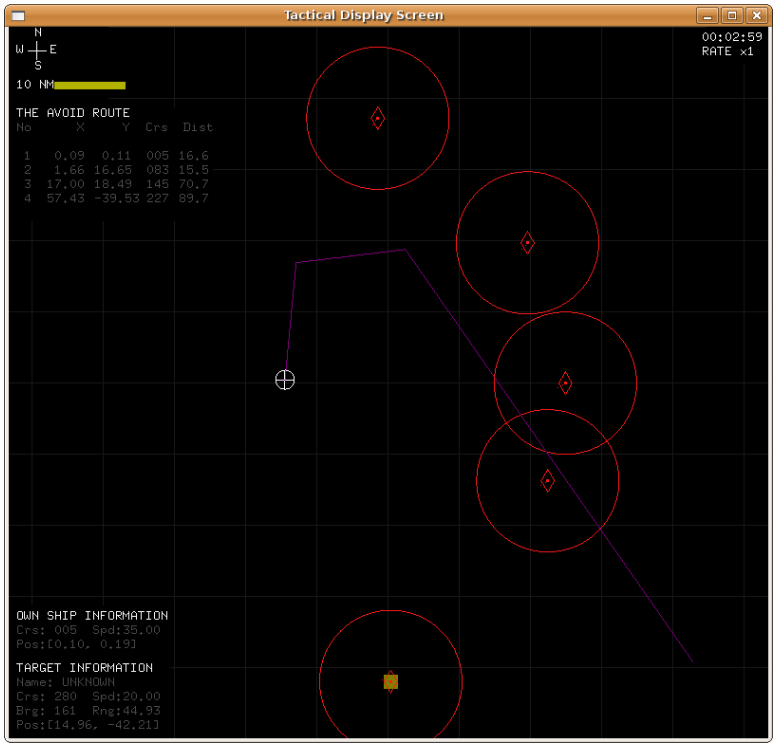


Figure 2.8: Prior 2D C&C Software Tool: Example of Escape Route Planning (Part I) [19]

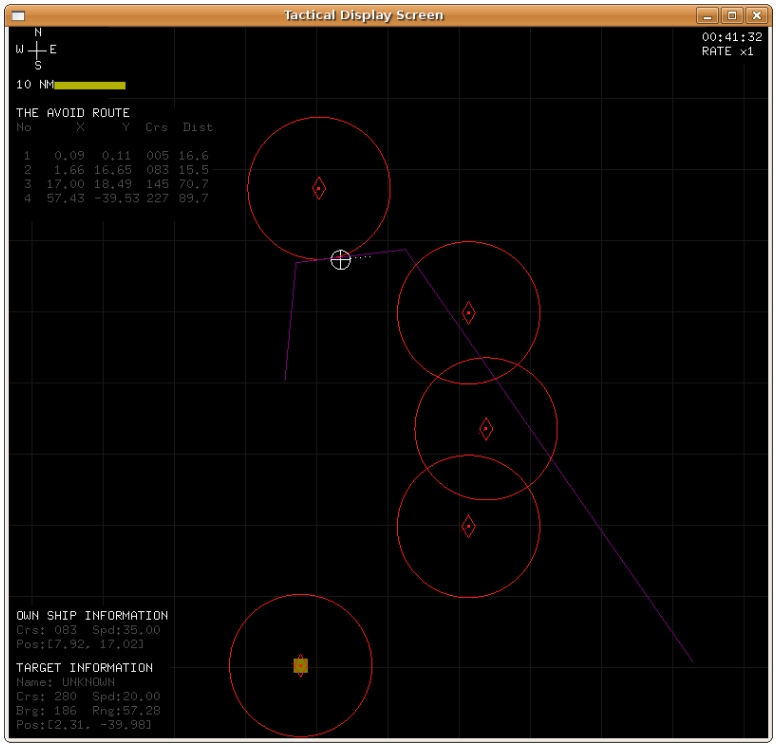


Figure 2.9: Prior 2D C&C Software Tool: Example of Escape Route Planning (Part II) [19]

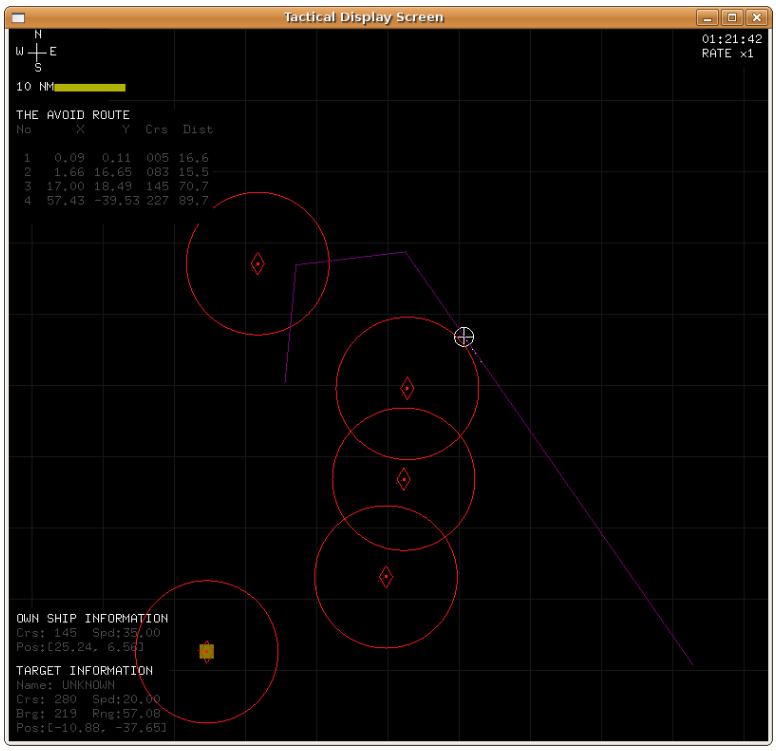


Figure 2.10: Prior 2D C&C Software Tool: Example of Escape Route Planning (Part III) [19]

2.3 Limitations with 2D Displays for Command and Control

While 2D computer-based display systems for C&C have evident advantages over traditional paper- and glass-based solutions, and obviously they can be very effective in numerous applications, there are still several limitations with employing only the 2D technology in C&C.

In particular, the capabilities of current communications technology give decision makers the ability to access large streams of information instantly and globally. However, while a great amount of information is available, identifying an essential piece of data becomes increasingly more difficult. Some data may be critical in a situation but it can be useless in another event. Due to the vast quantity of available information it is often impossible to display it in its entirety, and hence a critical piece of information might be left among

the omitted items. As pointed out by Aoki, the complexity and multiplicity of interactive regions in Combat Information Centers can create problematic situations for the decision makers [3].

In essence, displaying too detailed, excessive information pertaining to a C&C situation can decrease the awareness and performance of the commanders [58]. On the other hand, reducing the amount of information displayed to the commander can result in the loss of the essential ones. The selection process (of what data should be displayed at a given moment of time) is therefore a critical aspect of 2D C&C. Unfortunately, depending on the particular selection technique used and the particular context of the application, this process can also be unreliable. Thus, the C&C software developer's dilemma can be described as finding the best solution for displaying the right data at the right time, without overwhelming the commander with an excessive amount of information. As discussed in the next section, there are no easy answers to this dilemma.

2.4 New Display Solutions for Command and Control

Although tackling the problem of optimal information presentation in 2D C&C application is not an easy task, there have been several proposed solutions for this problem. According to our investigations, they can be grouped in the following categories:

(a) *Data Categorizing and Filtering.*

Although an ontology of information can be a way for operating large-scale control systems [8], the complexity and density of data involved often makes it very difficult or even impossible to come with a good design of appropriate data structures. Moreover, personal preferences regarding data layouts can vary from one commander to another. Information filtering is one of the methods to handle large information flows but several

issues still exist [40]. A good context model can possibly estimate a set of relevant pieces of information [17] [44]. However, the estimated result cannot be relied on by a critical system, such as a C&C system. Large information flows are still required to be presented to the decision makers. In addition, a general problem that needs to be considered when designing information display systems is to reduce the effect of change blindness, which is described as the failure of the human user to detect changes in representations of information that take place in his or her sight area [58].

(b) *Multiple Displays.*

A possible way of displaying a large amount of information is to divide the incoming data into several partitions and then display these data partitions on multiple display screens. A method using a perspective-aware interface can be used to enhance multi-display environments [61] (Figure 2.11). However, while users move their focuses from one screen to another, their train of thought might be lost [39]. In Combat Information Centers of US navy ships each tactical screen is designated for each dimension of the warfare, such as anti-air, anti-surface, and anti-submarine [14]. Each battle dimension is monitored and evaluated by a coordinator officer such that the related information can be evaluated and summarized before appearing to the chief commander. However, overwhelming complexity and data multiplicity can occur if the boundaries of data become overlapped and coordinators have different evaluation strategies [3].

(c) *A Single Large Display Screen.*

Instead of using multiple screens, there have been several attempts to fit the information into a single large screen. It has been shown that using large display screens can provide better situation awareness [37]. Some studies have pointed out that large screen displays improve the situation awareness and the assessment of the situation during C&C activities [32] [35]. For example, Australia's DSTO has proposed the Virtual Planning Room (ViPR) [16]. ViPR displays information on a large spherical screen. The information in a 3D metaphor, coupled with VR technologies, aims to allow the users to focus on a set of information while other sets are displayed in the background. However,

spatial focusing of people is limited; users cannot pay attention to all information that is displayed on the large screen at the same time. Also, interacting with large display screens is more difficult than with ordinary PC monitors [7].

Figure 2.11: Perspective-Aware Interface for Multi-Display Environments[2] [61]

(d) *3D Environments.*

The benefits of 3D images have been indicated by numerous research studies. For example, [89] shows that immersive environments can significantly improve the trainees' performance in complex procedures (Figure 2.12). Also, 3D visualizations for air traffic control seemed to improve awareness of the relative position between aircraft-aircraft or aircraft-airspaces landmarks [79]. Another experiment has led to an interesting result indicating that 3D spatial organizations of information (Figure 2.13) enabled users to access data items surprisingly quickly [24]. In addition, multiple layers of information that need to be presented by several 2D images can be combined into a 3D image (Figure 2.14) that allows the users to gaze at it together [15].

(e) *Virtual Reality.*

Combining 3D visualizations with the power of immersion in virtual worlds (Figure 2.15) can improve the learning process of complex spatial structures [82] and sophisticated procedures [89]. Two major virtual reality systems, head-mounted displays (HMD)

[2]Figures 2.11, 2.12, 2.13, 2.14, 2.15, 3.5, 3.6, and 3.7 are reproduced with permission from their respective authors.

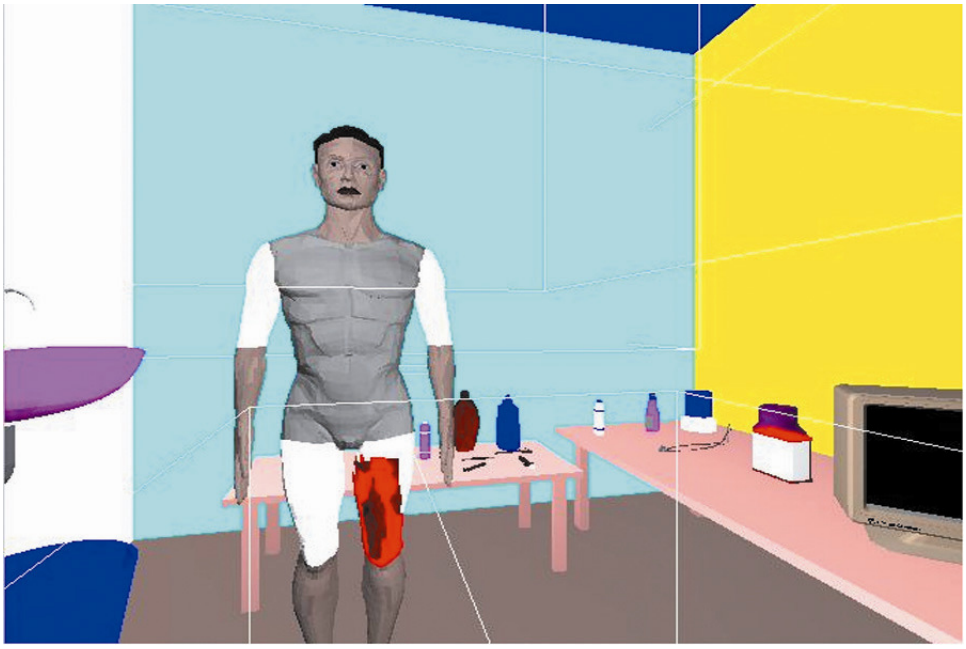


Figure 2.12: Immersive Environment Improves Training Performance for Complex Procedures [89]

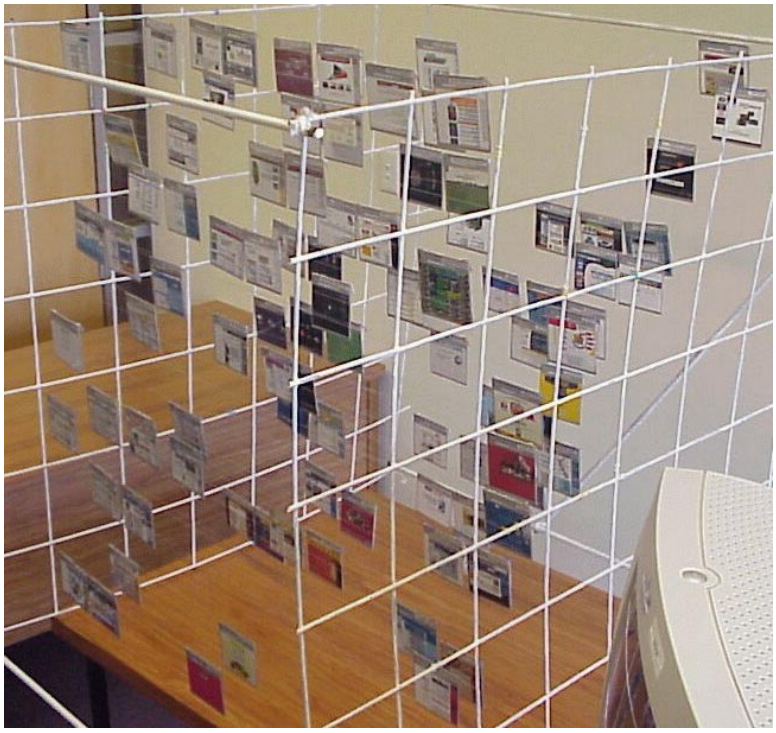


Figure 2.13: Spatial Organization of Information [24]

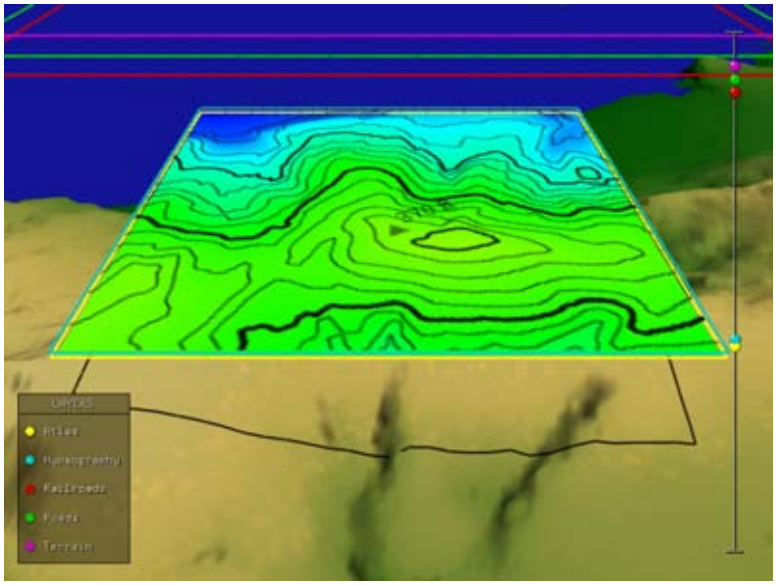


Figure 2.14: Hybrid 2D/3D GIS [15]

and the CAVE, are good virtual displays to create immersive environments [86]. For C&C tasks, in which commanders often require also interacting and communicating to people in the physical world, CAVE is likely to be the best candidate. A usability study compared three display interfaces (a desktop, the Fishtank, and the CAVE) for biological data exploration tasks. The results uniformly showed that users are more satisfied by using CAVE [74]. Our prior work also has shown that VR can help accelerate brain modeling and simulation research using virtual neuro-robotics [38]. In general, integrating VR technologies in complex applications brings added value, as highlighted in [71].

Based on these considerations, as well as on our determination to explore the challenging and captivating new VR technology, we have chosen CAVE as the medium for supporting C&C applications. The next chapter provides more details on VR and CAVE.

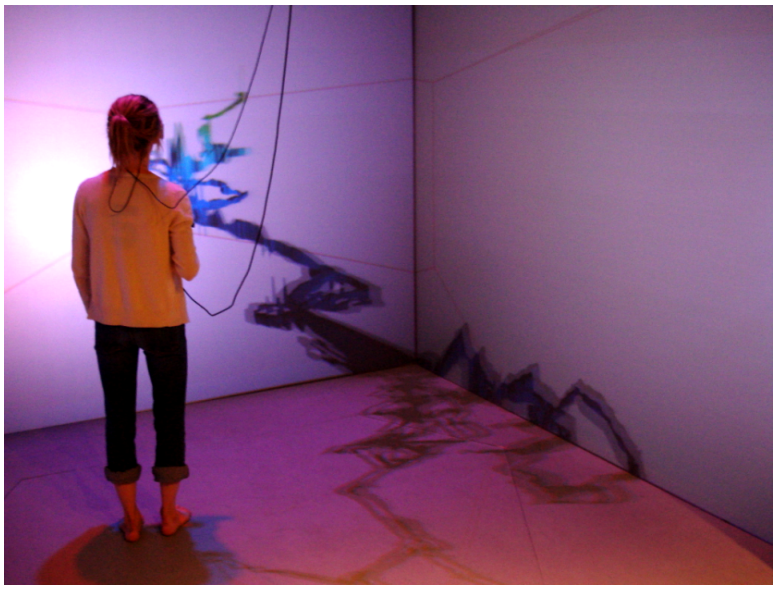


Figure 2.15: CAVE Improves Accuracy and Efficiency in Spatial Understanding of Complex 3D Structures [82]

Chapter 3

BACKGROUND: VIRTUAL REALITY AND CAVE

Because they provide the context for the work presented in this dissertation, in this chapter we take a closer look at the new, powerful technology and HCI paradigm provided by VR, and explore in some depth the main facets of CAVE, an environment that best illustrates the category of immersive VR. The main characteristics, specific devices or systems, and several defining applications are presented for both VR and CAVE. Specific interaction models within CAVE as well as HCI challenges for building CAVE-based simulations are also presented in this chapter.

3.1 Virtual Reality: An Outlook

Defining what is the exact meaning of virtual reality as a term used in several disciplines and by people coming from different backgrounds was not, and is still not, a straightforward

task. The concept of virtual reality was introduced by Ivan Sutherland in 1965, whose dissertation, SketchPad, a Man-Machine Graphical Communication System, brought several new ideas to the study of computer graphics, virtual reality, and computer interaction [92]. Jaron Lanier, the founder of the Virtual Programming Languages (VPL) research company, however, is considered to be the one who coined the term virtual reality in 1989 [54]. He defined virtual reality to be a way, supported by computers, to create a world, other than the real world, with people, other than the real people, living in that world [54]. A newer and more descriptive definition of virtual reality was presented by Sherman and Craig [86], who defined a virtual reality experience to be a medium (a virtual world) that is composed of several interactive simulations. These simulations have the ability to track the positions, movements, or actions of the person using the virtual reality simulation and give feedback to that user accordingly. This, therefore, gives the user the feel of being physically or mentally immersed in that experience or simulation.

3.1.1 Elements of a Virtual Reality Environment

In their book [86], Sherman and Craig indicated four core elements for an experience or a system to be considered virtual. These are: virtual world, immersion, feedback, and interactivity.

(a) *A Virtual World (Medium)*

Several researchers have set different criteria for a system to be considered a virtual reality one. First of all, the virtual reality experience has to take place in a virtual world. Biocca and Levy [13] defined the virtual world to be a space generated by a computer. In this space one or more users interact with two-dimensional, three-dimensional or other representations of objects or people, called avatars.

(b) *Immersion (Mental or Physical)*

The second key element of a virtual reality experience is immersion. Immersion could be of two types, mental immersion or physical immersion. Many people have experienced the mental immersion while reading a novel, listening to music, daydreaming, or watching a movie for example, where they become involved in the novel or the movie to the extension that they feel they live with the characters in their times, while in fact the readers or audience are just sitting on chairs or sofas. In physical immersion, people are physically involved in an experience; in a virtual reality experience, for example, the representation of the virtual world is modified and updated according to the locations, orientations, sights or movements of participants who are physically immersed in that experience. Between mental immersion and physical immersion, the latter is considered to be the key type of immersion as an element of a virtual reality experience. In other words, a virtual reality experience cannot be achieved without physical immersion. Mental immersion, on the other hand, can have different levels of immersion in a virtual reality experience. Nakatsu and Tosa [62] suggested two other classifications of immersion; they classify immersive situations into passive immersion and active immersion. They explained that when we forget about ourselves or when we lose our consciousness, we are most likely in a passive immersion. Active immersion, on the other hand, guarantees that our consciousness is not lost but we become more absorbed to a specific action. Nakatsu and Tosa claimed that if an interaction is involved in an action, it leads to an active immersion; this is the case in a virtual reality experience. If an action does not include interaction, such as when we read a novel or watch a movie, it is a passive kind of immersion.

(c) *Feedback*

Another key element of a virtual reality experience is feedback, also called sensory feedback [86]. This is also a key element of the interaction between the participant and the virtual reality system. In other words, the virtual reality system should reflect on and respond to the different actions done by the participant user, such as moving the head,

rotating the torso, walking, waving, picking up something, holding, kicking, looking, or maybe speaking. In order to obtain such a feedback in a virtual reality system, this system should track some or several parts of the participant's body depending on how specific and accurate the feedback needs to be. The tracked parts of the body may include the head, the hands, the orientation of the chest, or more.

(d) *Interactive Environment*

Interactivity is the last but not least element of a virtual reality world. The virtual reality system should provide an interactive interaction between the user and the virtual world. If a simulation of a virtual world is dynamic (i.e. involves interaction and feedback according to the user's positions, movements, or actions), it is considered to be interactive. Immersion, on the other hand, may not involve dynamic feedback to the user, it may, however, be just a response to a static graphical representation [80].

3.1.2 Classification of Virtual Reality Systems

Several forms of VR systems have been invented and experienced by researchers and participants. These systems could be classified into two main categories: immersive and non-immersive virtual reality as shown in Figure 3.1. Non-immersive VR could be divided into subcategories such as monitor-based VR and hand-based VR. Immersive VR, on the other hand, includes vehicle simulators, head-based VR, and projection-based VR. Examples of such categories include the FishTank display as a monitor-based VR, the handheld devices as a hand-based VR, the flight simulator as a vehicle simulator, the head mounted display (HMD) as a head-based VR, and the ImmersaDesk as a projection-based virtual reality.

In projection-based VR, large flat screens are used to display the virtual world to the participant. It can use either a single projector, such as the ImmersaDesk, or multiple projectors, such as the Cave Automatic Virtual Environment (CAVE), which is discussed

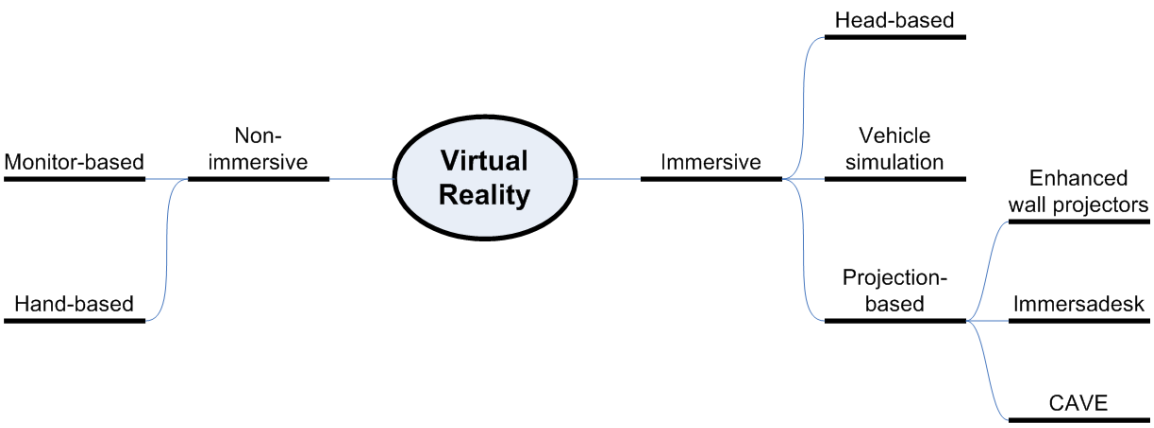


Figure 3.1: Proposed Taxonomy of VR Systems

in the next section.

Non-immersive VR is widely used in desktop computer games. Adding a three-dimensional element to a desktop computer game or any other visualization tool on a desktop computer, makes that game or tool a non-immersive VR one. In his book [42], Steven Heim mentioned three main characteristics of non-immersive VR screen-based, pointer-driven, and three-dimensional graphical presentations. Furthermore, the authors of [10] claimed that non-immersive virtual reality systems do not need special input or output devices to display a VR environment. A good specialized graphics card, however, is usually needed to run such applications.

3.2 CAVE: Characteristics, Devices, and Applications

First created by researchers at the University of Illinois's Electronic Visualization Lab in 1992, the CAVE is a room-sized cube where stereoscopic pictures or animations, which can be seen through special 3D glasses, are displayed on the walls using multiple projectors (Figure 3.2). A head tracking device, typically connected to the 3D glasses, contributes to

rendering in real-time stereoscopic images for the users' eyes. Through such images, the effect of physical immersion in the virtual world is realized. The CAVE was first built in response to a challenge for scientists and researchers to develop and demonstrate a one-to-many visualization tool that utilizes large projection screens [26] [27].

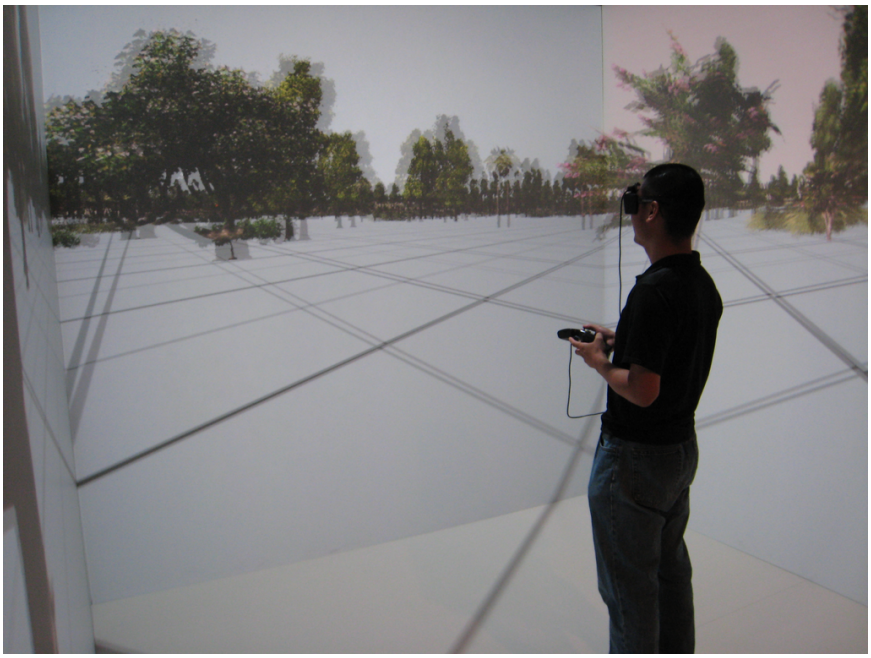


Figure 3.2: A 4-Wall CAVE at the Dessert Research Institute in Reno, Nevada

Different CAVEs around the world use projectors that are directed to three, four, five or six of the walls of the CAVE. Today, the CAVE is considered one of the key interactive VR environments. That is, an environment that includes elements of virtual world, immersion, sensory feedback, and interactivity [86]. The most typical devices that can be found in a CAVE are the wand (Figure 3.3), which allows the user to point and enter commands (including for navigation), the goggles, which are essential to create the illusion of 3D immersion to the user, and the head tracker (Figure 3.4), which is necessary to position the human user inside the virtual world displayed in CAVE.

Day by day, more and more applications are being designed and run inside the CAVE. CAVE has been used for many purposes, including education, entertainment, and exploration, such as for interactive story telling [22] or for spaceflights over the surface of the planet

Figure 3.3: A Wand for CAVE

Figure 3.4: Goggles and Head Tracker for CAVE

Mars [69]. CAVE offers one of the best and most complex data visualization tools that exist at this time. After a decade of using the CAVE as a display of geology information for oil exploration planning, it has been shown that the CAVE helps to increase oil recovery (Figure 3.5) [55]. In geosciences, the CAVE has been used as an interactive visualization tool for measurement, analysis, and interpretation of geological data [51].

Since the CAVE provides an immersive environment without any harm to users, it is possible to simulate hazard situations to observe the behavior and reaction of people with a high

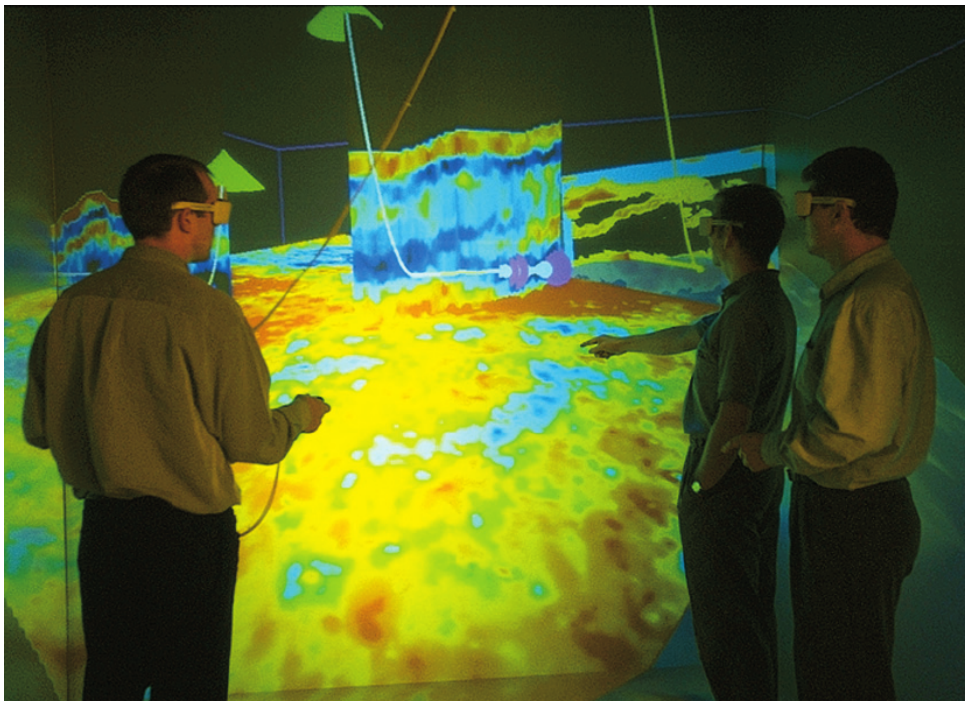


Figure 3.5: CAVE Used for Oil Exploration Planning (Courtesy of StatoilHydro ASA) [55]

degree of fidelity. The CAVE, for example, has been used in the collection of data to study human behaviors while evacuating from an area under a terrorist bomb attack [84].

Furthermore, the CAVE can be used to enhance the educational experience such as in NICE [48]. NICE is a VR system that allows children to build, cultivate, and interact with virtual ecosystems to create stories from their interactions. The CAVE is also widely used in scientific and data visualizations, such as in [90] [9] [11] [50]. The CAVE is also explored in medicine; for example, in [105] an immersive virtual environment was introduced for tensor magnetic resonance imaging (DT-MRI), which can be used in the studying of the changes in white matter structures after gamma-knife capsulotomy and pre-operative planning for brain tumor surgery. Geographic Visualization inside the CAVE cannot be missed here, an example of that having been presented in [56].

The CAVE is also used in entertainment. The CaveUT system, for example, is an immersive entertainment system that supports immersive virtual reality installations based on

the Unreal Tournament game engine [46] [47]. Learning through entertainment using the CAVE was also explored in [78], in which the author explored interactivity in virtual reality environments, including the CAVE, through a system specifically designed for children that allows them to learn by playing. Another example of such applications is the widely played game, Second Life, which was also explored inside the CAVE to study different interaction methods [53].

In applications that share certain characteristics of C&C in CAVE, researchers have focused on surveillance and security systems, such in [71], where several surveillance cameras are carried by a mini Blimp which in turn is tele-operated using a VR interface with haptic display (Figure 3.6 [71]). The CAVE serves as an interactive control room for the entire surveillance system, receiving multiple video streams from its set of airborne and fixed cameras.

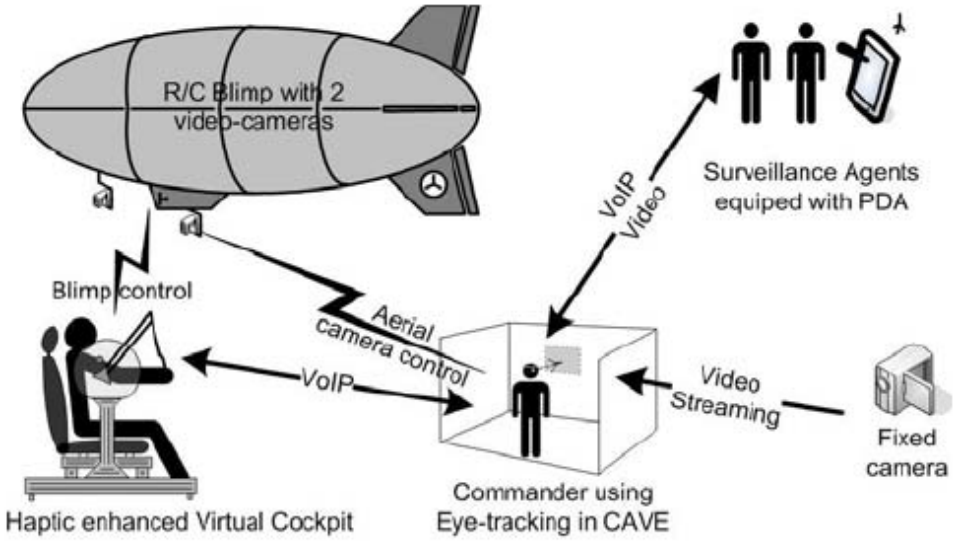


Figure 3.6: Example of Advanced CAVE-based Surveillance and Security Application [71]

In this application, where advanced technologies are emphasized, the visual contents received from the blimp is rendered to the pilot via a head mounted device. Gesture interface included in the system allows the controlling of the blimp by the pilot from inside this virtual cockpit (Figure 3.7) [71].

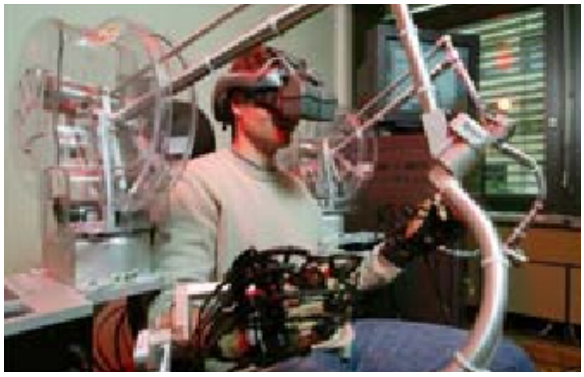


Figure 3.7: Virtual Cockpit with HMD and Gesture Interface [71]

Another application that relies on VR and focuses on exploring new paradigms for situation awareness (SA) and C&C in military centers is described in [16]. The paper describes the Future Operations Centre Analysis Laboratory (FOCAL) at Australia's Defence Science and Technology Organisation (DSTO) and presents the conceptual design and prototype development of the Virtual Planning Rooms (ViPR) environment, which is conceived as a novel immersive visualization solution that displays multi-media information on large displays placed on the walls of its rooms.

Yet some other studies have been focused on various user requirements and usability aspects of CAVE. For example, the work presented in [92] describes an experimental comparison of interaction in CAVE versus the real world that included two manipulation tasks, one single handed, the other double handed. The results show, among other things, that users made more errors in CAVE without the virtual hand than with it, and committed fewer errors in the real world. The usability issues related to the visual feedback in both tasks are also pointed out in this study, and several implications for usability and prototyping of virtual environments (VE) are discussed. Along similar lines of investigation, [97] examines the effects of real-world distraction on user performance in VEs.

Using three tasks designed to cause different levels of awareness of the real-world distraction, the authors explore the effect of reduced visual stimulus in the peripheral area on

user performance and usability in immersive VEs. Their study indicates that real-world distraction can have direct effects on user performance (positive or negative), depending on the particular tasks performed and the specific environments in which they are performed. In terms of performance requirements for CAVE, [75] Preddy and Nance indicate that a general requirement for CAVE applications is that either real-time or quasi real-time performance is required on the simulation model. In addition, based on their experience with the Virginia Tech CAVE and a survey of related literature, the authors propose three key requirements for successful simulations in CAVE [75]: Portability among CAVE-specific input output (I/O) devices; Effective and efficient inter-process communication; and Overcoming the limitations related to I/O device interaction.

Although the CAVE offers several benefits, its cost has been prohibitive for many research institutions. Currently, however, the cost is decreasing and low-cost CAVE-like systems are becoming available [23]. We expect that the CAVE will continue to gain popularity in the near future. Due to its obvious advantages and its promising future, we have focused on using CAVE for running C&C simulation applications, and have embarked on contributing to SE methodologies and tools for facilitating the design and development of such applications, As further discussed in Chapter 7, Related Work, although there are several programming libraries to support the implementation of VR and the CAVE applications, methodologies or frameworks to build such systems can hardly be found.

3.3 Interaction Inside the CAVE

Interaction inside the CAVE is mainly done by either direct manipulation or navigation. Other styles of interaction include menu-selection, communication, command-line and other, less used, styles of interaction.

3.3.1 Direct Manipulation

In this technical report [60], the author lists three styles of interaction inside a VR environment through manipulation, which can also apply to interaction through direct-manipulation inside the CAVE. The first style is the direct user control. This style of interaction is done through interface gestures. The second style of direct manipulation is done through physical devices, such as the wand. The user holds the wand inside the CAVE and uses it to move objects, point to things, push buttons, etc. It has the same use as the mouse but in a three-dimensional environment. A virtual control is another style of manipulation inside the CAVE, the physical device in this scenario is replaced by a virtual one. This virtual device, though, can have more possible functionalities than those in a physical one. Sherman and Craig [86] introduced a new style of manipulation, which is the agent control. Here, the user is in direct communication with an intelligent agent, who, in turn, performs a requested action or command.

3.3.2 Navigation

Map navigation is a well-known and widely used style of interaction in desktop applications. It is also considered a key interaction style inside the CAVE. Navigation includes two main processes: way-finding and travel [86]. In order to navigate through a map, for example, the user has to know the destination, go through a path, and travel in the path to reach that destination. Travel can be done through different styles, such as physical locomotion, ride along, fly-through, towrope, move the world, put me there, etc. [86].

3.4 Human-Computer Interaction Challenges for Building CAVE-based Simulations

As explained in the previous sections, the CAVE is an interactive environment that involves a significant amount of interaction between the user and the system itself. Several human-computer interaction challenges arise to deal with the interaction design of applications for the CAVE, especially when it comes to simulations. Going through the process of interaction design, the first challenge that an interaction designer will face is the question of which data-gathering technique should be used to identify the user needs and establish requirements. Few research investigations have been done to study and identify which data-gathering technique is most suited for a CAVE-based simulation. Those techniques, including questionnaires, interviews, focus groups, observations, studying documentation, and researching similar products [83], were well explored in desktop applications in terms of advantages and disadvantages. The same comparison is needed, however, for CAVE-based simulations.

Following the interaction design lifecycle, another key challenge that concerns interaction designers for CAVE-based simulations is the fact that there are no clearly identified efficient and effective conceptual models or interface metaphors in this case. As discussed earlier, the two main styles of interaction inside the CAVE are manipulation and navigation; other styles of interaction, however, need to be more investigated and researched. Examples of such styles include the menu-selection, exploring, browsing, conversing, and searching. Data interpretation and analysis could also be considered a challenge here. CAVE-based simulations are relatively new to the software engineering industry; thus, the specification and design aspects of such applications are not thoroughly or comprehensively explored yet. This also applies to task analysis, diagrams, and architectural models for this kind of simulations.

Building a low-fidelity prototype for a CAVE-based simulation is another challenge not to be missed. Low-fidelity prototypes are considered by many researchers and developers to be useful in terms of simplicity and affordability. It is also known that low-fidelity prototypes are useful when it comes to quickly explore designs and ideas in the early stages of the development cycle [83]. This, however, may not be easily transferred to CAVE-based simulations. In the latter, 3D rather than 2D presentation of the data has to take place in terms of sketching and storyboarding. This would also consume more time since the designer has to sketch a whole environment and be able to spread it into three, four, five, or six sides of the CAVE.

A CAVE is not a system that can be owned by an end-user yet. It is much more expensive than a PC and, as mentioned earlier, takes a large physical space. As a result, evaluating a prototype or a final product to be used inside the CAVE raises a challenge of getting intended users to travel in order to participate in the evaluation of that prototype or product. This consumes more money, more arrangements, and less flexibility compared with the evaluation of desktop applications. Although some emulators and simulators were built to test CAVE applications on desktop computers (e.g. VR Juggler and FreeVR), they do not give the full specifications, features, or interaction styles that can be found in a real CAVE. The difference between a real CAVE and an emulator of it becomes wider and more obvious when it comes to the evaluation of a real-time critical system that runs inside the CAVE.

There are also the challenges that are encountered when dealing with the IO devices inside the CAVE. Several devices, from different brands, can be used as IO devices, including sensors, projectors, goggles, wands, gloves, speakers, etc. This brings up the challenge of understanding the underlying structure of the IO device and the real-time synchronization of these devices together. The refresh rate of the graphical component inside the CAVE, for example, is usually faster than the refresh rate of a CAVE-specific motion device [75]. Such a challenge should be taken into serious account when designing CAVE-based simulations. Using shared memory either synchronously or asynchronously for CAVE-based simulations

is another challenge that should be considered by interaction designers and software engineers as well. This was presented in more detail in [75]. Macedonia recommended the use of asynchronous data sharing if the simulation is real-time [57]. This, however, needs to be more investigated in a CAVE-based simulation.

Figure 3.8 summarizes the challenges that were explained in this section by dividing them into two main categories, hardware and software challenges, and then into sub-categories.

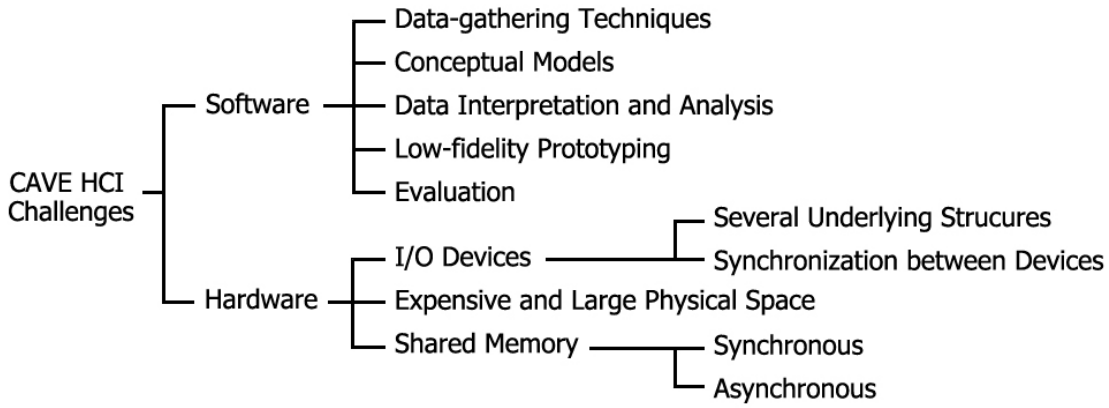


Figure 3.8: Proposed Classification of CAVE HCI Design Challenges

Using the CAVE as a shared collaborative virtual environment raises several challenges in its own. Synchronizing data between two or more CAVEs, a desktop computer locally-connected to a CAVE, or a desktop computer remotely-connected to a CAVE or several CAVEs introduces several requirements that should be carefully considered in terms of network connections, security, bandwidth, real-time rendering, and maybe more. Also, using the CAVE and a different platform, such as a laptop, a mobile device, a desktop computer, etc., raises the challenge for software designers and programmers to design their applications to accommodate several platforms.

Chapter 4

CAVEMANDER: PROPOSED METHOD

This chapter focuses on the methodological component of CAVEMANDER. Specifically, we describe the proposed approach (method) for designing, building and executing simulation scenarios in CAVE. The method consists of a number of activities (steps), each of which aimed at producing a specific artifact, which is a design and/or an implementation artifact. The activities covered include scene definition, simulation build up, scenario creation, and scenario execution. These are described using software engineering notations (in particular, UML diagrams) and cover two key phases of the software engineering process: design and implementation (we refer together to them as software construction phases). As discussed in the Future Work of the thesis the method can be extended in the future with activities pertaining to other phases of the software process, in particular requirements specification and evolution (maintenance). The proposed approach's focus on design and implementation has been driven by practical reasons, as by completing software construction activities it is possible to define, build, and run simulation applications in CAVE. The other major component of the CAVEMANDER project, its set of software resources that support the

proposed construction method, is described in Chapter 5.

4.1 Method Activities and Artifacts

An overview of the proposed method, focused on the construction of CAVE-based C&C simulations is presented in Figure 4.1 in the form of a UML activity diagram that highlights both the activity flow (consisting of method steps) and the object flow (consisting of artifacts, or design and implementation components). From a procedural point of view, the main four activities are, in order, scene definition, simulation build up, scenario creation, and scenario execution. All these are described in more detail in the following sections, but from a terminology point of view, it is important to note that by **scene** we mean a collection of C&C unit types (movable or fixed, e.g., tanks or supply bases), together with their defining parameters and commands, as well as associated environment factors (e.g., visibility or weather conditions). In essence a scene provides a higher level context on which a specific **simulation** (and, consequently, a simulation scenario) can be built and is created or modified/updated during the scene definition activity. Next, by simulation (or simulation application, to differentiate it from a possible real-world application in which CAVE could be involved in the future) we mean the specific software implementation that supports the scene, in particular (but not only) the classes associated with the unit types included in a scene (e.g., the ClassTank or ClassSupplyBase). While scene definition is a design activity, simulation build up (in which the simulation is created) is an implementation (coding) activity in which existing reusable code is integrated with new, customization code, and a runnable software package (SimScene) is produced. Finally, a **scenario** is a concrete embodiment of a simulation, in which a given scene is particularized through the assignment of specific values to unit types, parameters, and environment variables (e.g., number of tanks = 10, number of supply bases = 2, and max visibility = 10 miles). A scenario is built from scratch or updated from an existing one in the scenario creation activity. After a

scenario is created it can be run in the scenario execution activity of the CAVEMANDER software construction method. From the above, it can be seen that a scenario is both a design artifact (a description of the C&C situation) and an implementation artifact (it can run in CAVE). From Figure 4.1 it can also be seen that the above activities need to be performed in sequence only when a new scene is created (to define the new scene, then build its associated SimScene code, and then create its first associated scenario), but in the case previous artifacts are already available, it is possible to skip some activities and perform only those needed at some point in time (for example, only to update an existing scenario). All four method activities are described next in their dedicated sections using UML activity or state-chart diagrams that show specific steps performed and samples of artifacts produced.

4.2 Scene Definition

Scene definition is the first activity of the proposed CAVEMANDER software construction procedure. For each new simulation application this activity must be performed in order to create the simulation's associated scene. If a suitable scene already exists, then this activity is optional and can be performed only if updates are needed to be applied to the scene.

As shown in Figure 4.2, there are in essence three parts to this activity. First, an application scene concept needs to be defined for identification purposes (e.g., is this scene for a land military C&C simulation or is it for a naval surface operation?). The specification of the application scene concept consists simply in creating a text describing the purpose of the simulation. Second, the scene's unit types, the unit types' models, their properties, and their available commands need to be specified. Third, the scene's environment factors taken into consideration need to be defined and their properties detailed. As the second and the

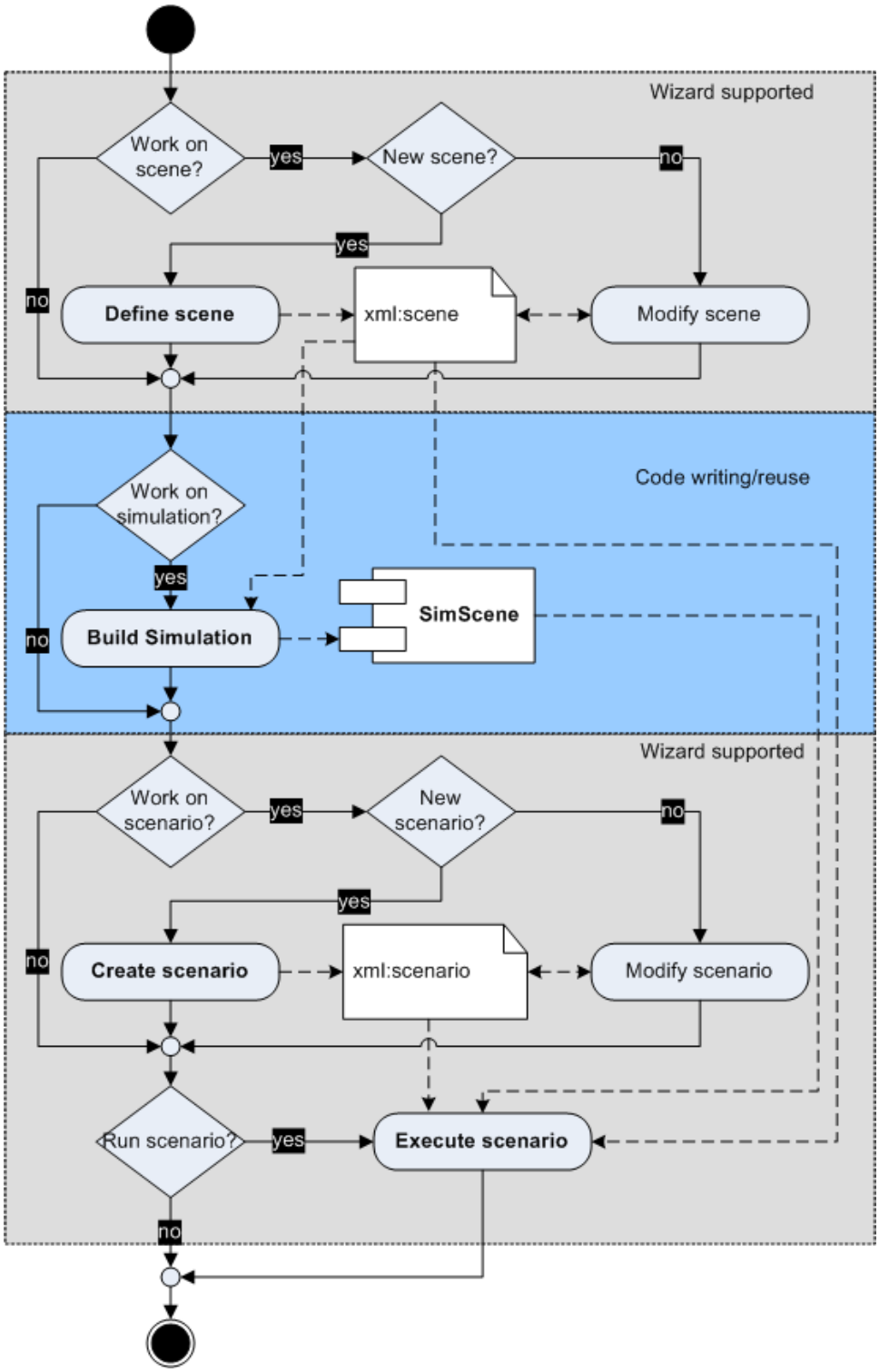


Figure 4.1: Overview of the CAVEMANDER Method

third parts of the activity can be conceptually performed in parallel, they are represented on two parallel, separate branches of the activity diagram shown in Figure 4.2. The result of this activity is the creation of a scene XML file, an artifact that contains the description of all elements that make up the scene, including unit types and environment factors. Note that the scene contains unit types, and not units per se (the latter are specified later in the scenario creation activity). An example of a scene XML file artifact is shown in Figure 4.3. Note that for the sake of simplicity the example is rather simple, with only two unit types (a hummer and a hovercraft) and only three environment factors (visibility, wind direction, and wind speed) included. This is shown in Figure 4.4 with a clearer, more detailed conceptual representation of the same artifact described in Figure 4.3 using XML. In larger scale simulations the scene description file can grow very large, particularly as a given simulation and scenario may not use all the units and environment factors contained in a scene, which thus can gather all kinds of descriptors, some of them not necessarily used through the scene's lifetime. The scene XML file artifact created in this activity is used in the other three activities of the CAVEMANDER method.

4.3 Simulation Build Up

Simulation build up is the second activity of the proposed CAVEMANDER software construction procedure. For each new simulation application this activity must be performed in order to create the simulation's associated code, which is included in the package called SimScene. If a suitable simulation code package already exists, then this activity is optional and can be performed only if updates are needed. Notably, this is the most code development-intensive activity of the proposed software construction method. In is also worth noting that this activity can consist of two parts: creating code from scratch (possibly, partially using existing API methods already included in CAVEMANDER) or reusing existing code available in the CAVEMANDER platform. In contrast with the scene definition and sce-

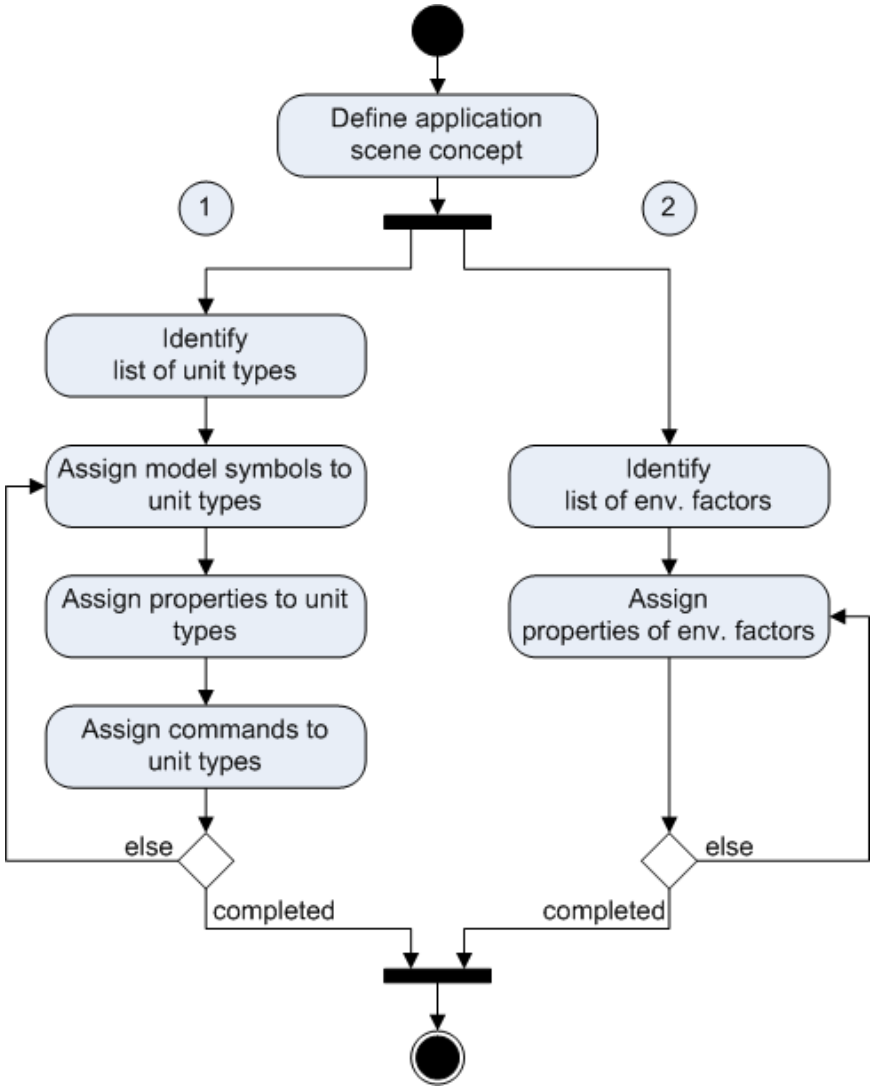


Figure 4.2: CAVEMANDER Method — Scene Definition Activity

Figure 4.3: CAVEMANDER Method — Example of Scene Artifact Produced in the Scene Definition Activity (XML File)

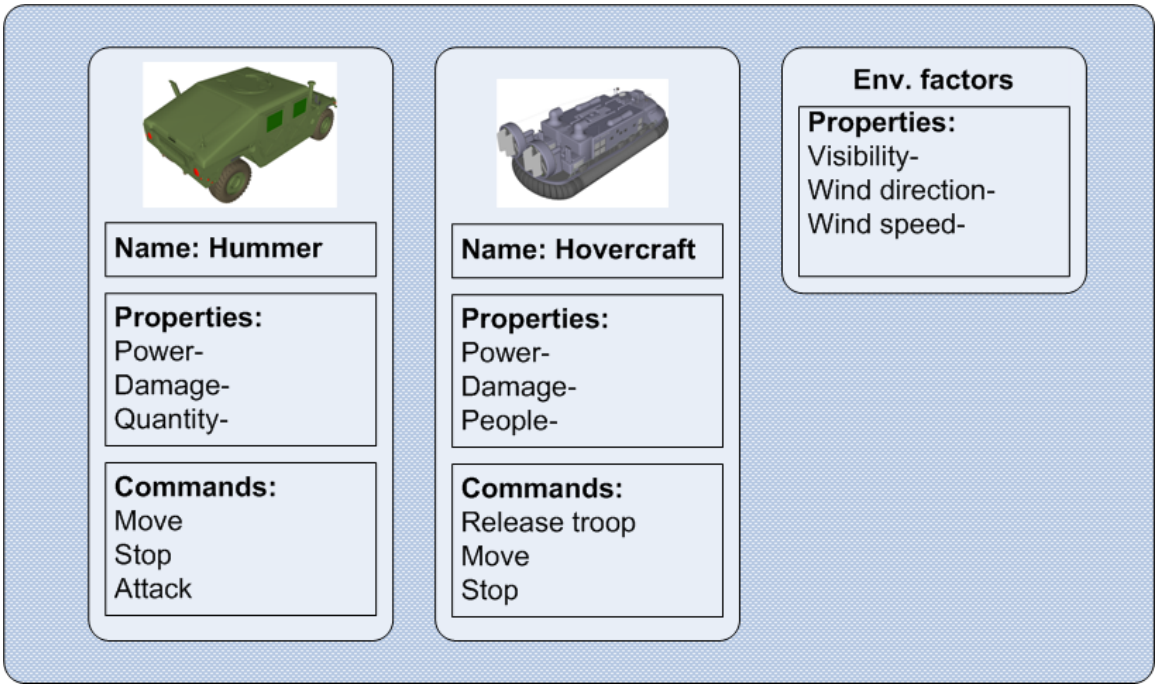


Figure 4.4: CAVEMANDER Method — Example of Scene Artifact Produced in the Scene Definition Activity (Conceptual Representation)

nario creation activities which can be easily performed by end users without knowledge of computer programming, the simulation build up activity involves code writing, meaning that knowledge of a programming language is needed. However, as the software resources included in CAVEMANDER will grow in time through added development and possibly open source community contributions, the need for writing customized code in this activity will decrease. Nevertheless, as new applications will always be likely to be needed, the need for specialized code customization by programmers cannot be completely eliminated.

As shown in Figure 4.5, there are six parts to this activity. First, an associate scene file needs to be selected to drive the simulation's code build up activity. Second, the classes needed to for the scene's unit types are written or retrieved from the exiting CAVEMANDER library. The writing of a new class for a new scene unit type involves writing code to describe the unit type, its properties, and its associated commands. Sample commands that can be written or re-used from the existing CAVEMANDER API are shown in Table 4.1. Third,

Table 4.1: CAVEMANDER Method — Examples of Command Functions Produced or Re-used in the Simulation Build Up Activity

Command Function	**Description**
MOVE (Position)	MOVE (Position) & Move unit from current location to another.
STOP (void)	Stop and station unit while moving.
SAFE_DESTRUCT (void)	Permanently remove unit from simulation.
ATTACK (Target)	Move unit close to the target and attack when the target is within encounter distance.
GROUP (Units)	Add units as subordinate units of a selected unit (leader unit). After adding, the subordinate units will perform all the commands that are assigned to the leader unit.
UNGROUP (Units)	Remove units from following the command of a unit leader.

the classes for the scene's environment factors are written or retrieved from the existing CAVEMANDER library. Fourth, the programmer needs to include all these new classes in the main simulation package, called SimScene. Fifth, an update mechanism for each time step needs to be provided by the programmer. Lastly, the code created in this activity needs to be compiled and linked to the CAVEMANDER library.

The result of this activity is the creation of a compiled and linked code file, an artifact that contains the executable description of all elements that make up the scene, including unit types and environment factors. A conceptual representation example (not the actual code produced) of an artifact created in this activity is shown in Figure 4.6. Note that some of the code needed for describing related unit type commands is reused from the existing CAVEMANDER library (the code for the Move and Stop commands) while other has to be written from scratch by the developer (the code for the Release-troop command of the Hovercraft unit type). The code created in this activity is used in the scenario execution activity of the method.

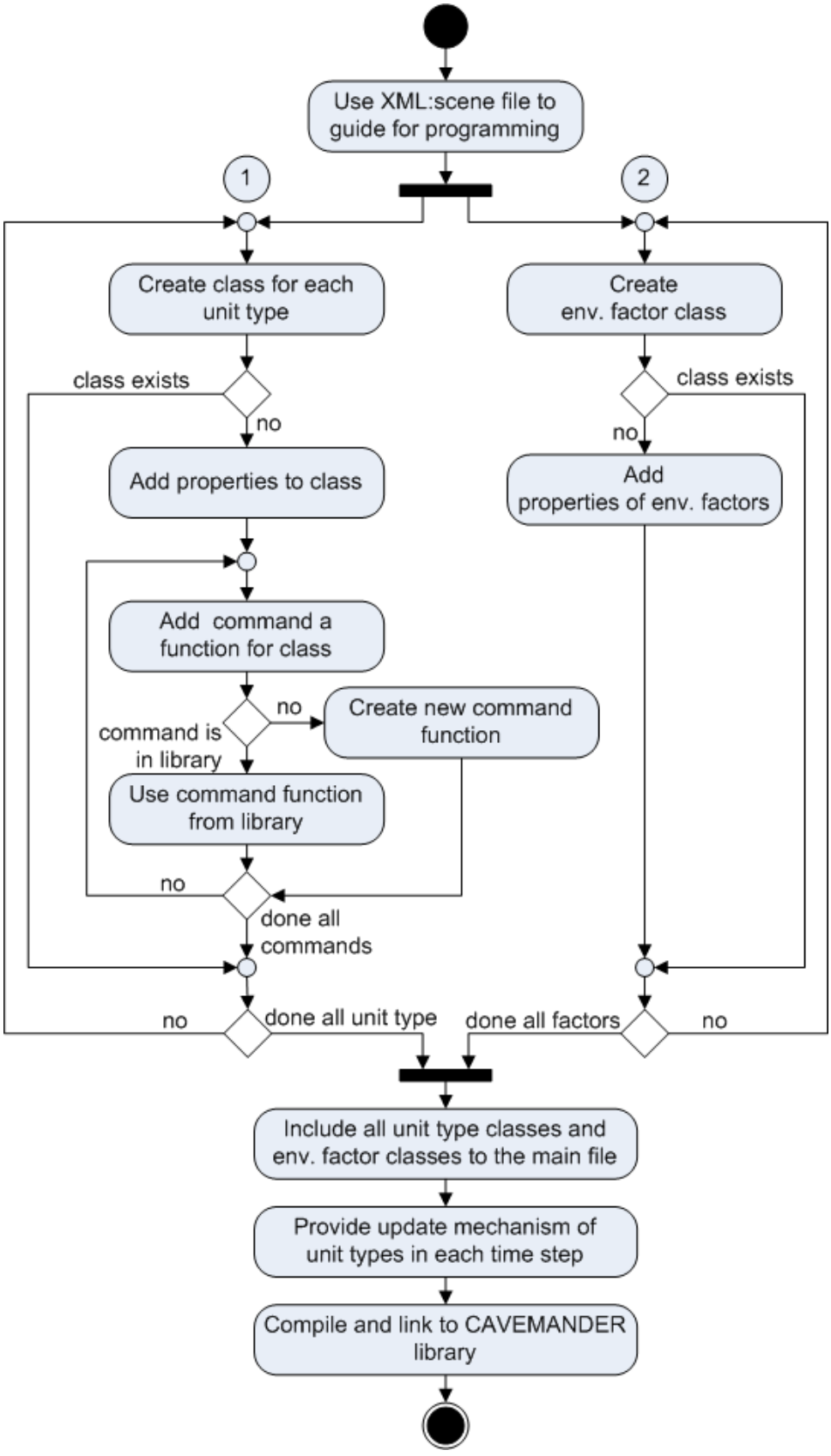


Figure 4.5: CAVEMANDER Method — Simulation Build Up Activity

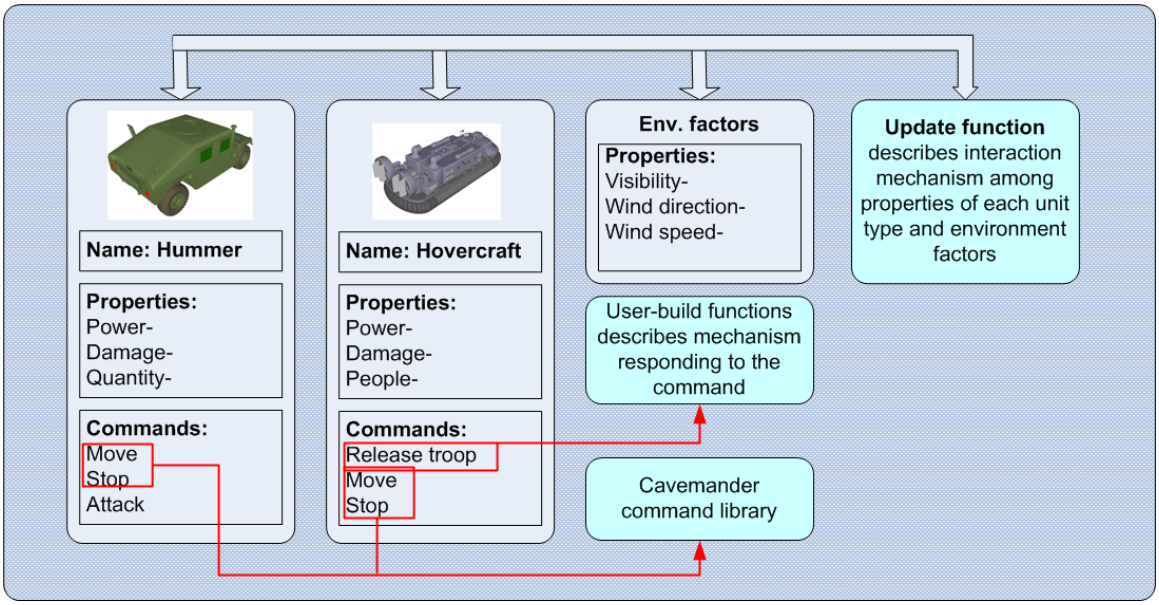


Figure 4.6: CAVEMANDER Method — Example of Artifact Produced in the Simulation Build Up Activity (Conceptual Representation)

4.4 Scenario Creation

Scenario creation is the third activity of the proposed CAVEMANDER software construction procedure. For each new simulation application this activity must be performed in order to create the simulation's first associated scenario. If a suitable scenario already exists, then this activity is optional and can be performed only if updates are needed.

As shown in Figure 4.7 there are in essence four parts to this activity. First, a configuration of the scenario is needed. This includes defining the scenario concept (i.e., providing specific details on the scenario purpose, for example a scenario associated to a scene can deal with providing supplies to the front line units, while another, associated with the same scene, can be about limiting the advancement in a given area of enemy troops). The specification of the scenario concept consists simply in creating a text describing the purpose of the scenario. Other configuration items that need to be defined are the log file name (for play back purposes), the communication port number for running in CAVE, the name of the

scene with which the scenario is associated, and the terrain on which the scenario will run (a terrain representing a given area, such as the San Francisco Bay area, or a region on Mars). Second, the scenario's units, their values (IDs) and their initial locations need to be specified. Third, the scenario's environment factors need to be initialized. As the second and the third parts of the activity can be conceptually performed in parallel, they are represented on two parallel, separate branches of the activity diagram shown in Figure 4.7. Lastly, the scenario needs to be loaded on the client in order to be ready to be invoked and run. The result of this activity is the creation of a scenario XML file, an artifact that contains the description of all elements that make up the scenario, including units and environment factors. Note that the scenario contains individual units, rather than unit types (the latter being specified in the scene creation activity). An example of a scenario XML file artifact is shown in Figure 4.8. As with the scene XML file artifact, on larger scale simulations the scenario description file can grow rather large. Conceptual representations for the scenario artifacts, one for a military simulation, and the other for a planet expedition simulation are shown, respectively, in Figures 4.9 and 4.10. The scenario XML file artifact produced in the scenario creation activity, together with the SceneSim code created in the simulation build up activity, are used in the scenario execution activity of the CAVEMANDER method.

4.5 Scenario Execution

Scenario execution is the last activity of the proposed CAVEMANDER approach. It simply means that the resources created for simulation are loaded on the server as well as on the client (CAVE) and then the simulation begins. During the simulation, the user has the possibility to slow down, speed up, pause, resume, and stop the simulation (Figure 4.11. In other words, the execution time can be adjusted to a fraction or a multiple of the real-time. In addition, a playback feature that allows reversing the course of a scenario execution is also included in the design of CAVEMANDER (but not yet implemented, and left for

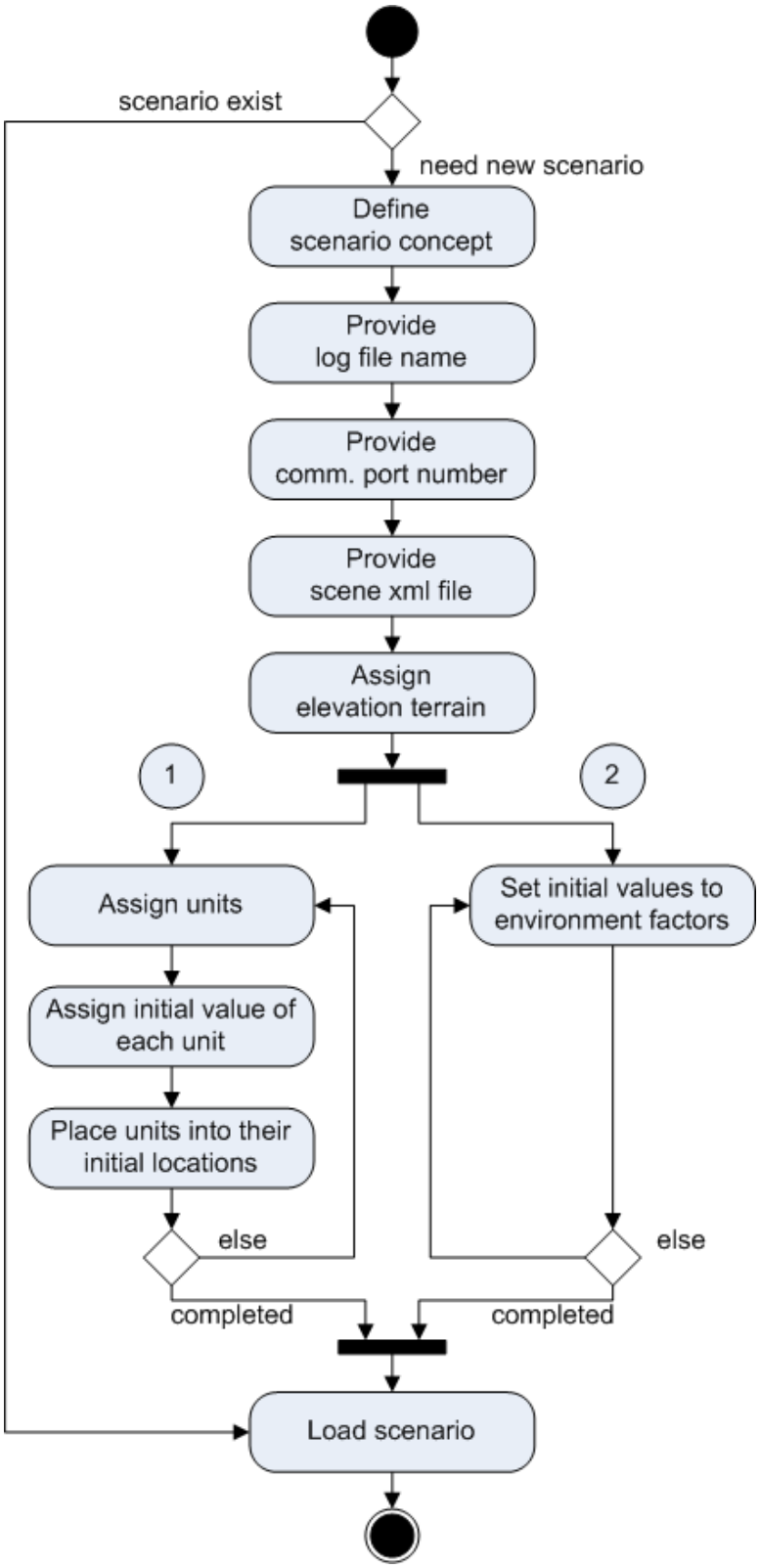
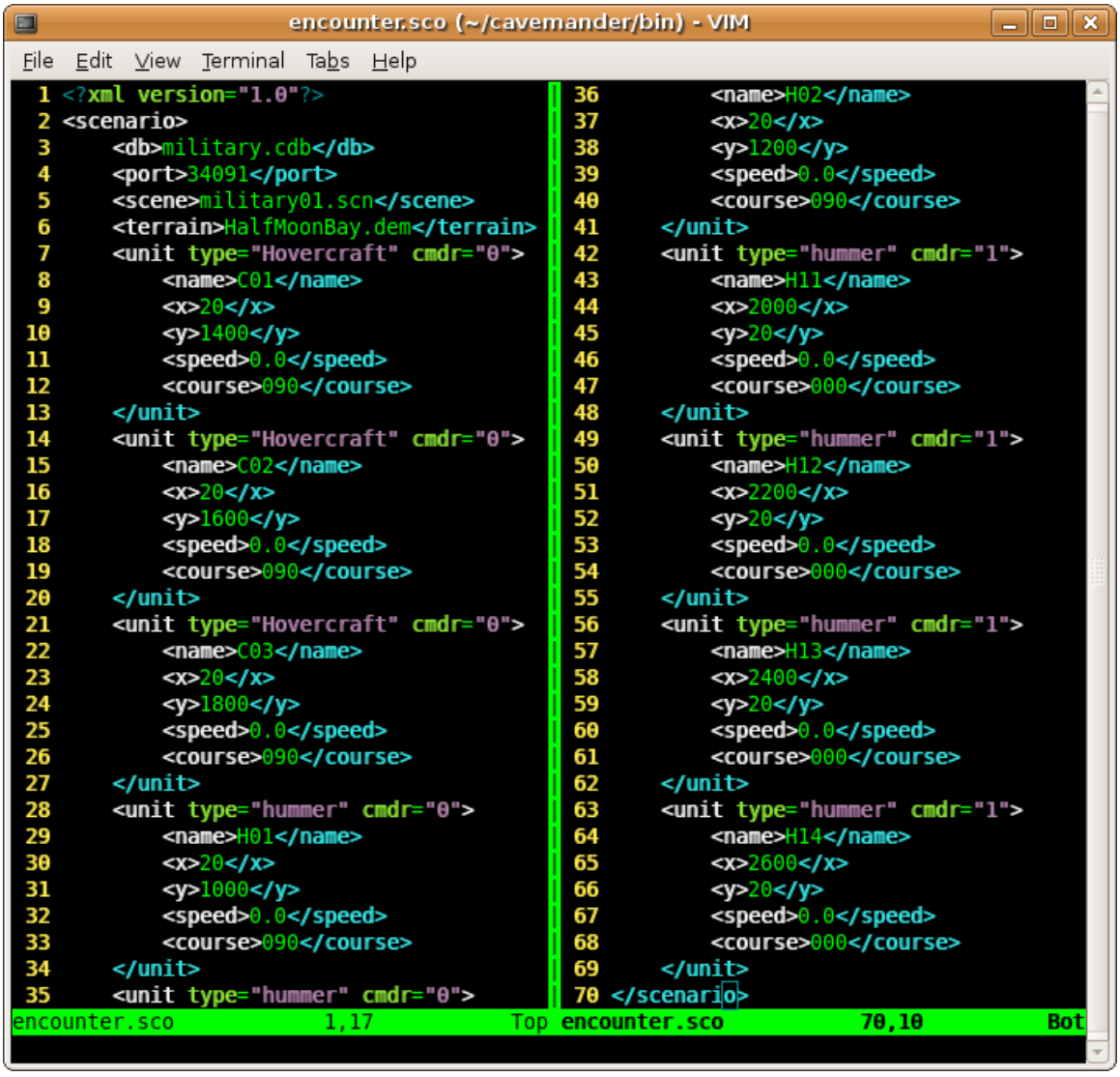


Figure 4.7: CAVEMANDER Method — Scenario Creation Activity




```
<?xml version="1.0"?>
<scenario>
    <db>military.cdb</db>
    <port>34091</port>
    <scene>military01.scn</scene>
    <terrain>HalfMoonBay.dem</terrain>
    <unit type="Hovercraft" cmdr="0">
        <name>C01</name>
        <x>20</x>
        <y>1400</y>
        <speed>0.0</speed>
        <course>090</course>
    </unit>
    <unit type="Hovercraft" cmdr="0">
        <name>C02</name>
        <x>20</x>
        <y>1600</y>
        <speed>0.0</speed>
        <course>090</course>
    </unit>
    <unit type="Hovercraft" cmdr="0">
        <name>C03</name>
        <x>20</x>
        <y>1800</y>
        <speed>0.0</speed>
        <course>090</course>
    </unit>
    <unit type="hummer" cmdr="0">
        <name>H01</name>
        <x>20</x>
        <y>1000</y>
        <speed>0.0</speed>
        <course>090</course>
    </unit>
    <unit type="hummer" cmdr="0">
        <name>H02</name>
        <x>20</x>
        <y>1200</y>
        <speed>0.0</speed>
        <course>090</course>
    </unit>
    <unit type="hummer" cmdr="1">
        <name>H11</name>
        <x>2000</x>
        <y>20</y>
        <speed>0.0</speed>
        <course>000</course>
    </unit>
    <unit type="hummer" cmdr="1">
        <name>H12</name>
        <x>2200</x>
        <y>20</y>
        <speed>0.0</speed>
        <course>000</course>
    </unit>
    <unit type="hummer" cmdr="1">
        <name>H13</name>
        <x>2400</x>
        <y>20</y>
        <speed>0.0</speed>
        <course>000</course>
    </unit>
    <unit type="hummer" cmdr="1">
        <name>H14</name>
        <x>2600</x>
        <y>20</y>
        <speed>0.0</speed>
        <course>000</course>
    </unit>
</scenario>
```

Figure 4.8: CAVEMANDER Method — Example of Scenario Artifact Produced in the Scenario Creation Activity (XML File)

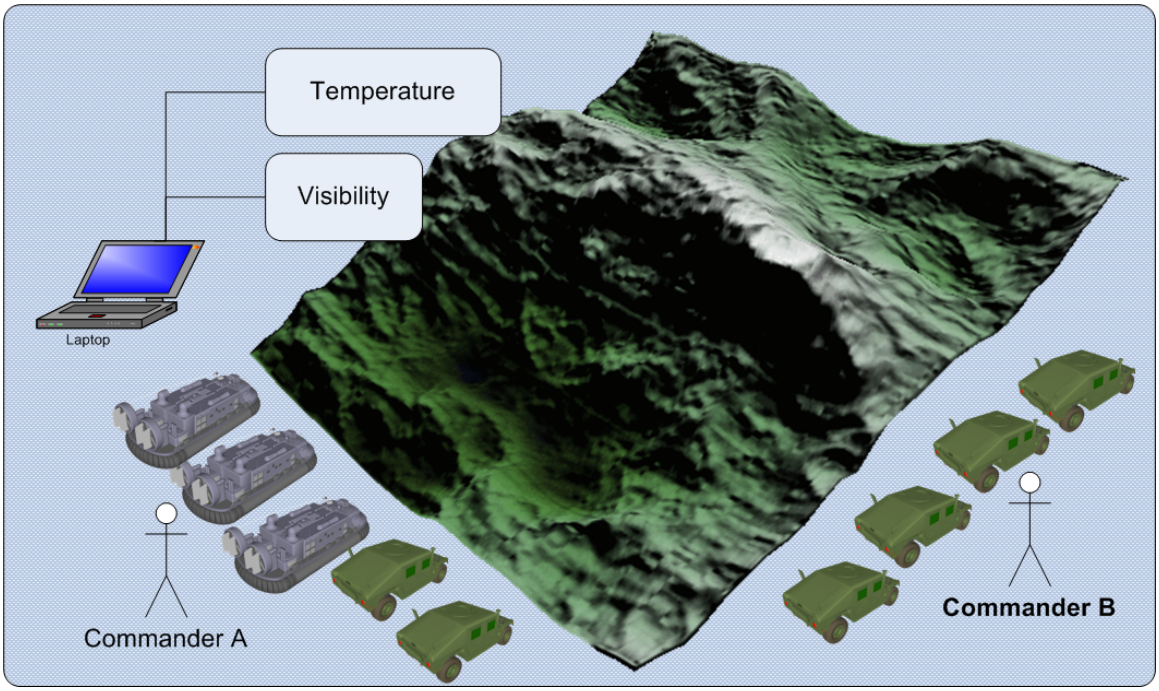


Figure 4.9: CAVEMANDER Method — Example of Scenario Artifact Produced in the Scene Definition Activity (Conceptual Representation for a Military Simulation)

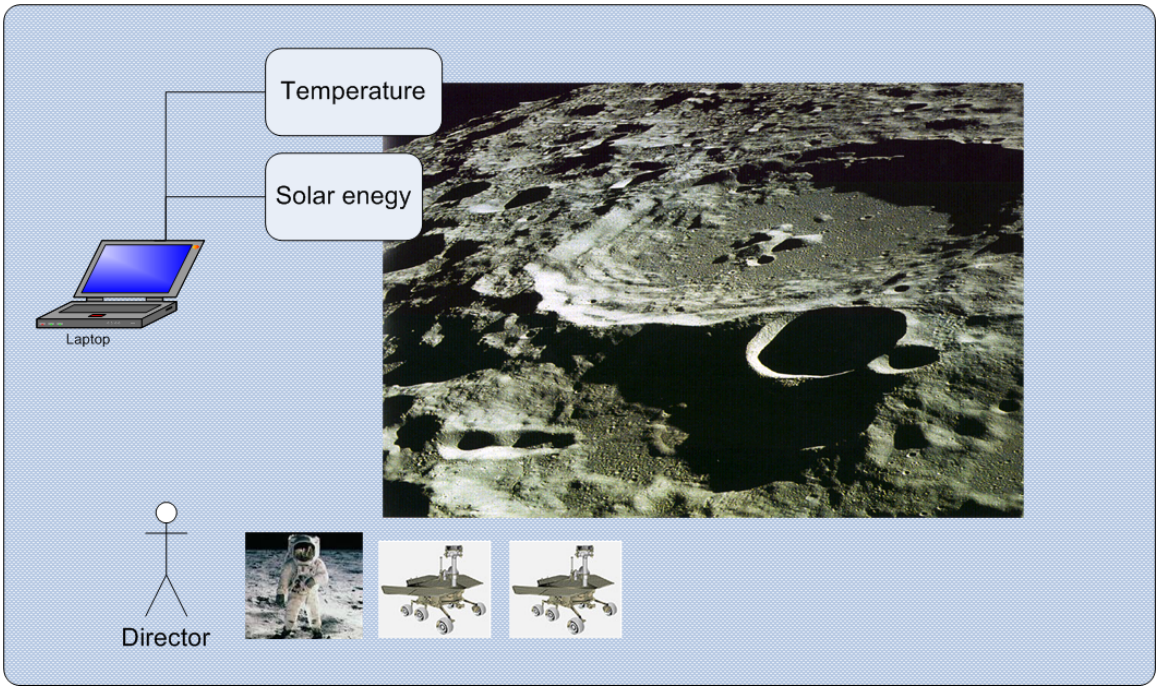


Figure 4.10: CAVEMANDER Method — Example of Scenario Artifact Produced in the Scene Definition Activity (Conceptual Representation for a Planet Expedition Simulation)

future work).

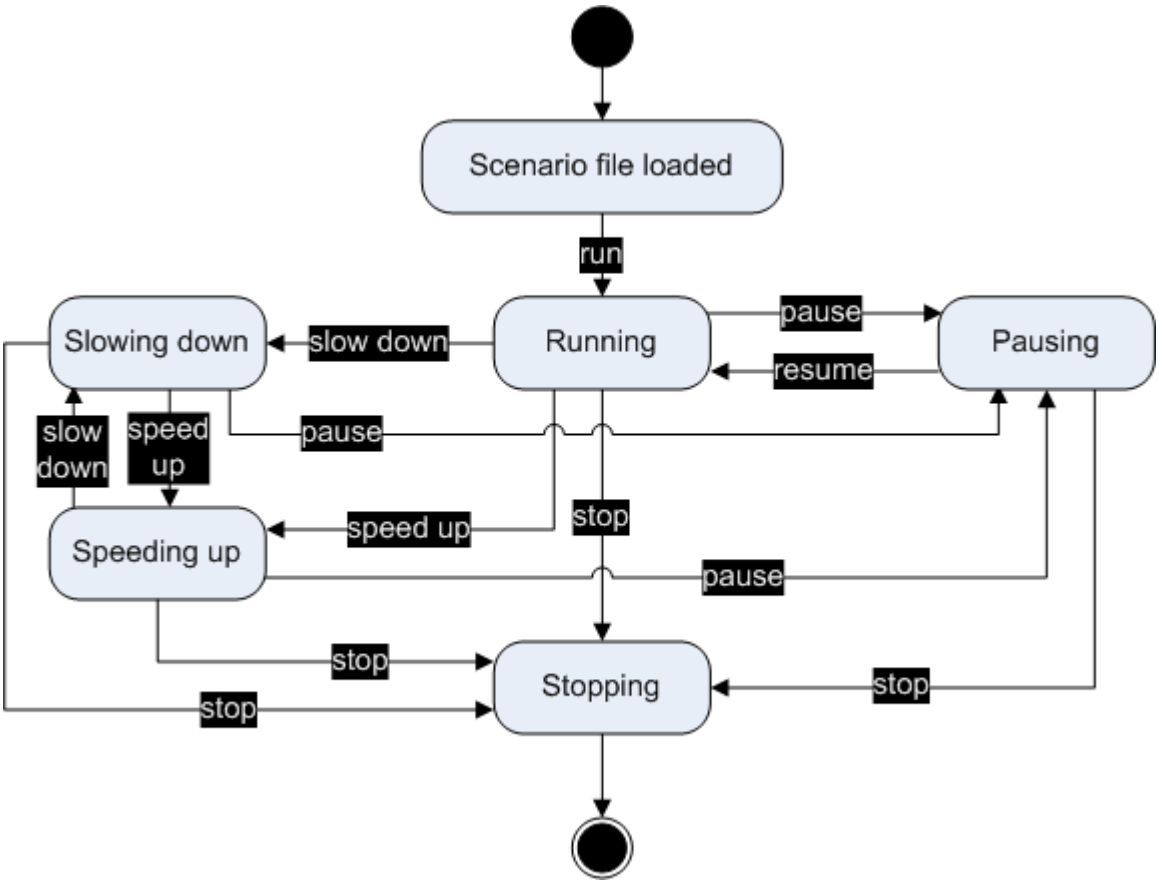


Figure 4.11: CAVEMANDER Method — Scenario Execution Activity

As shown in Figure 4.12, during the simulation the CAVE user can be in one three possible modes: command mode, in which commands can be issued by the user (a commander) to the scenario units, view mode in which the user can zoom in and out of the terrain and can select scenario units, and a travel mode, in which the user can move through the VR world by flying, walking, or riding along a selected unit, can move either forward or backward, and can rotate the view available to him or her. Sample illustrations of these three user modes are shown, respectively, in Figures 4.13, 4.14, and 4.15.

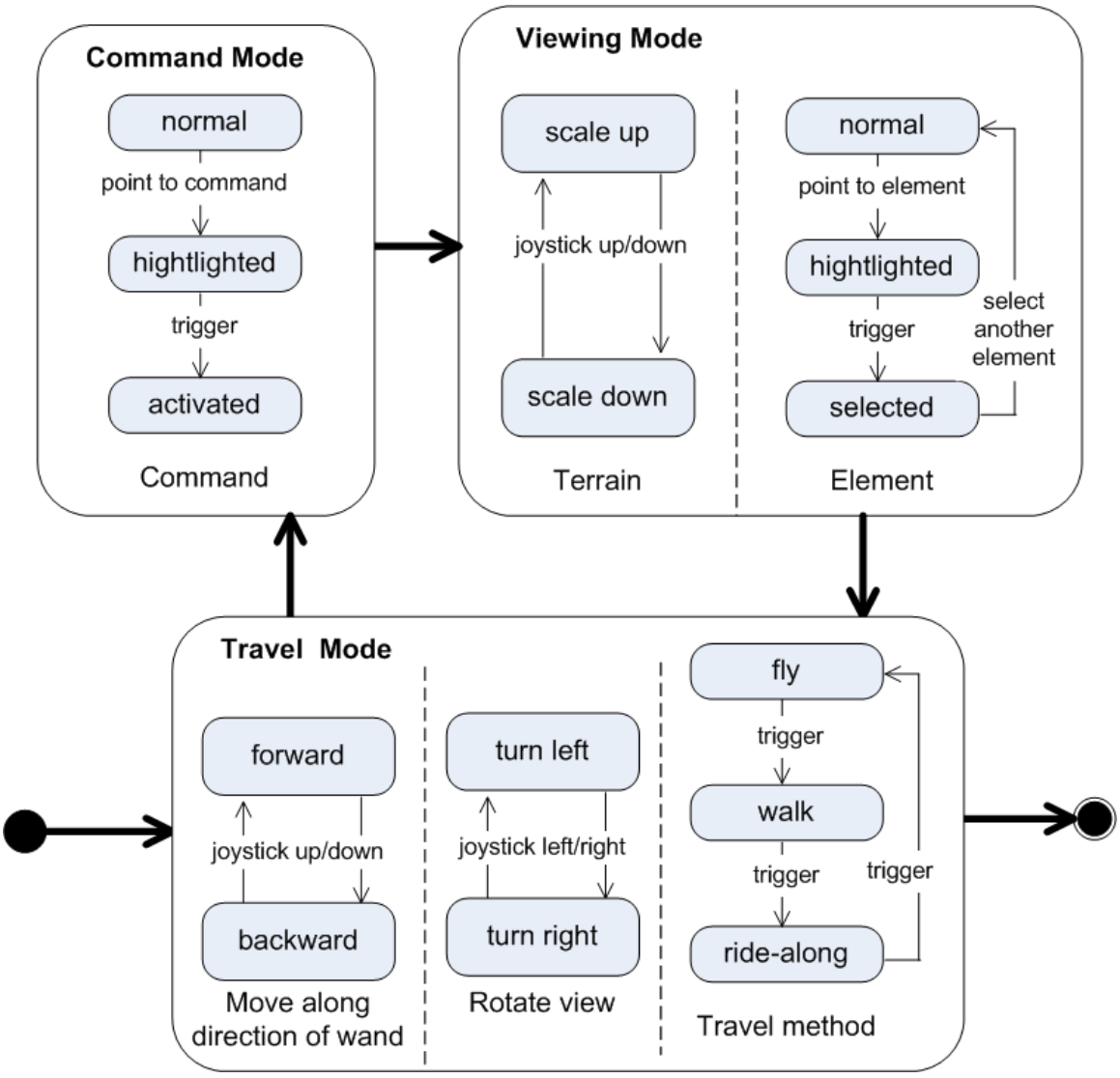


Figure 4.12: CAVEMANDER Method — User Modes in Scenario Execution Activity

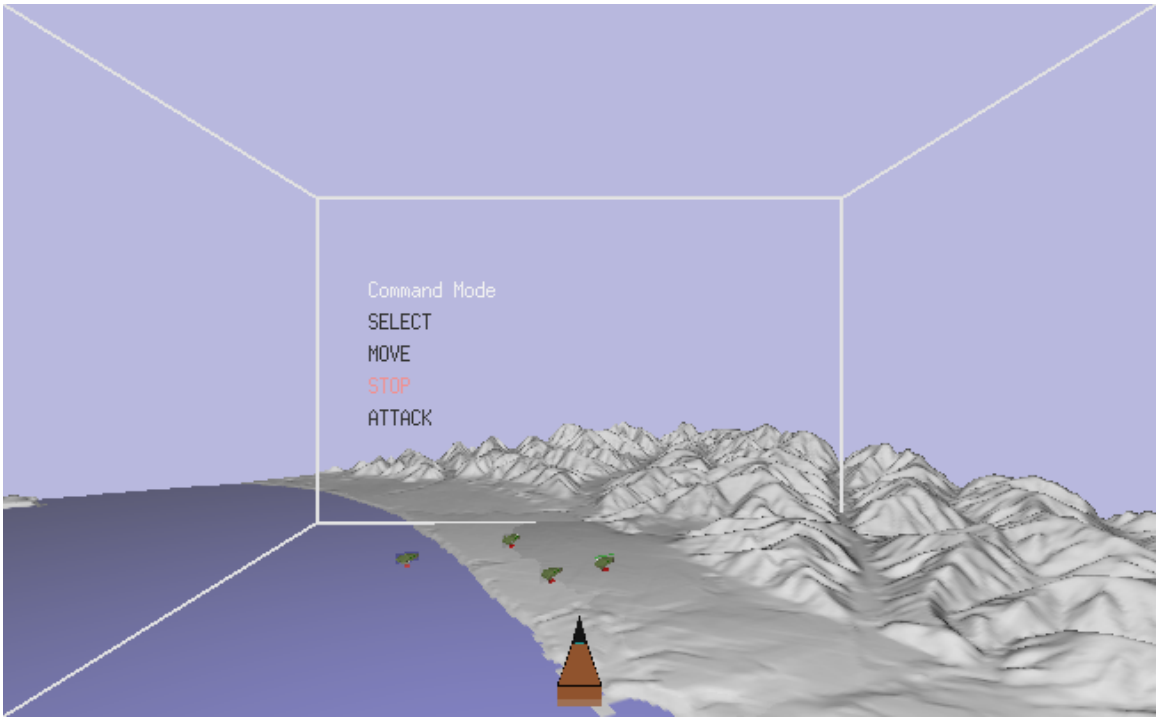


Figure 4.13: CAVEMANDER Method — Example of Command User Mode in Scenario Execution Activity

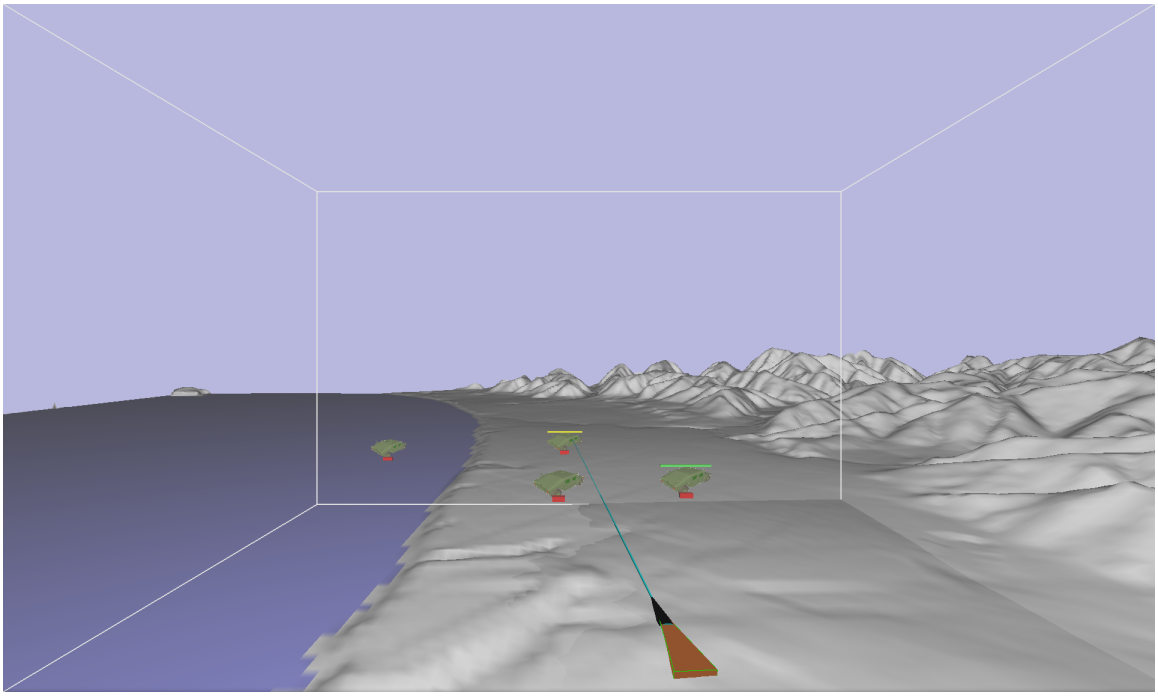


Figure 4.14: CAVEMANDER Method — Example of View User Mode in Scenario Execution Activity

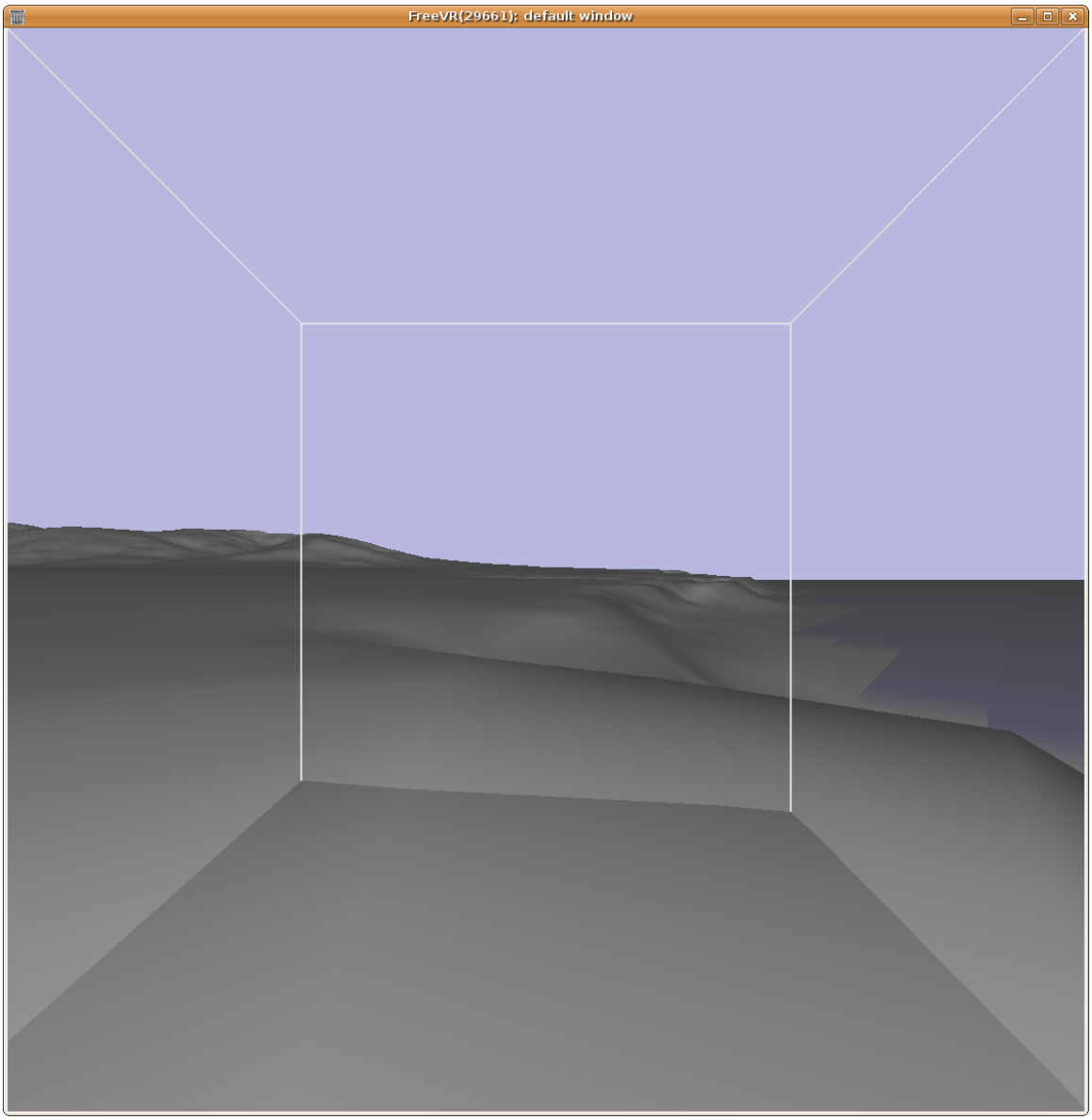


Figure 4.15: CAVEMANDER Method — Example of Travel User Mode in Scenario Execution Activity

Chapter 5

CAVEMANDER: SOFTWARE PLATFORM RESOURCES

This chapter describes the software platform resources of CAVEMANDER. The definition we used for a *software platform* includes a software application that acts as a base for other programs and accepts third-party plug-ins [73]. At this time, CAVEMANDER does not accept third-party plug-ins (this can be part of its future developments), but its present software resources have been specifically designed to provide a basis for building other programs, specifically C&C simulations for CAVE. In fact, the set of CAVEMANDER software resources, presented in this chapter, have been designed to support the proposed simulation software construction method described previously in Chapter 4. While the method represents the first of the CAVEMANDER project's defining components, the second such component consists of the set of supporting software resources described here. In summary, CAVEMANDER provides both a software construction method and an architecture to create C&C simulation scenes and scenarios. The CAVEMANDER architecture consists of two main parts, a server and a client part. Within the server part, of particular significance is the CAVEMANDER Wizard (also called ServGUI), a software program that, as shown

earlier in Figure 4.1, enables end users who are not computer experts to perform three of the four major activities included in our proposed method: scene definition, scenario creation, and scenario execution. The description of CAVEMANDER as a software platform is illustrated in this chapter by using various UML diagrams (specifically, use case diagrams, class diagrams, and component diagrams) as well as GUI snapshots.

5.1 CAVEMANDER Architecture

The CAVEMANDER software platform is based on a client-server software architecture, where the server provides information and services to multiple clients [88] [76]. The server acts as an information and service provider that communicates with the clients, sending them required pieces of information. The clients request and use services and information from the server. This architecture has many advantages, including specialization of functions (with separation of requirements, which facilitates better use of resources and optimization), extensibility, and replaceability. In our system the server contains the simulation information and code and the client is responsible for displaying scenes and scenarios as well as providing an interactive system for the user to interact with the scenarios.

The main components of the CAVEMANDER architecture are the **Server** and the **Client**, as shown in Figure 5.1. Note that in this Figure the components currently developed are shown with a bold contour line, while the other (secondary components) are included in the architecture to provide guidance for future developments of the platform.

The Server has two core components, **ServGUI** and **SimScene**, along with three other, secondary components: the **Log**, the **Playback**, and the **CommHub**. The first core component of the Server is the ServGUI, or the CAVEMANDER Wizard. This is a full-fledged software tool in its own, intended to be used by users that are not necessarily computer

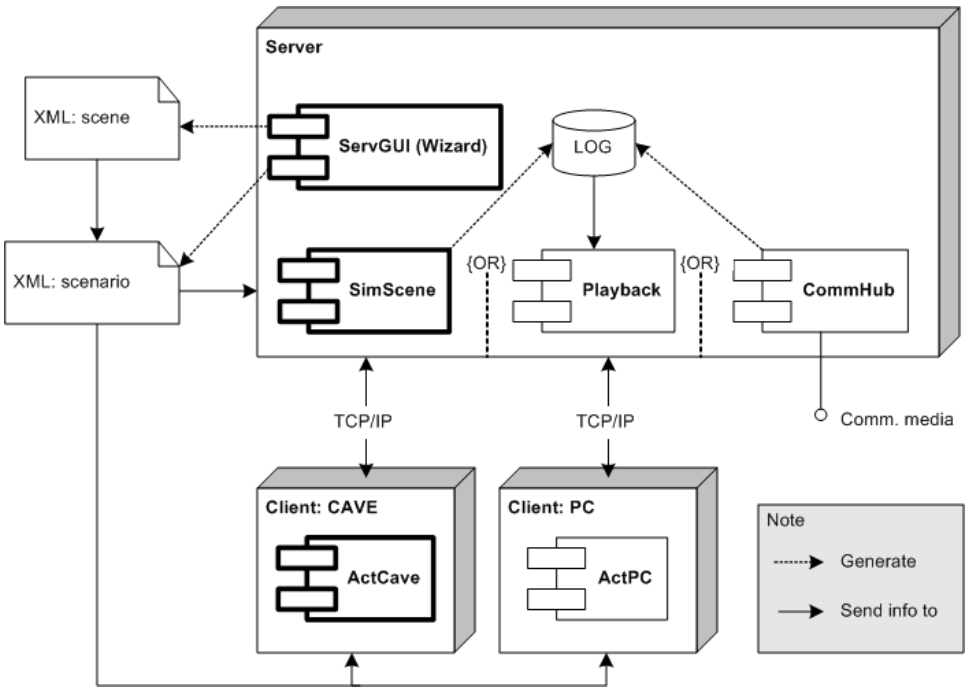


Figure 5.1: CAVEMANDER Software Platform Architecture

experts. The Wizard's main role is to enable the creation of simulation scene and scenario descriptors in XML format that can be used by the other software components. ServGUI provides a graphical user interface wizard, which makes the process of creating scenes and scenarios efficient and effective, and supports three of the four CAVEMANDER method activities: scene definition, scenario creation, and scenario execution. SimScene is the second core component of the Server and consists of a set of reusable software resources, including C&C unit descriptor classes, templates, and API command functions. The secondary components of the Server are designed as follows: the Log is simply a collection of log files that can be used for recording scenario executions for the purpose of later playback (re-running). The software for re-running recorded simulation scenarios is included in the Playback package, which is intended to allow the instructors and the trainees to review and analyze C&C scenarios performed in CAVE. The CommHub is another software component planned as future work, which will allow connecting the CAVE with real-world C&C applications via telephone lines, wireless connections, and other communication media.

The Client has two software components, **ActCAVE** (developed and available for use as part of CAVEMANDER platform) and **ActPC**, both designed to enable the execution of simulation scenarios on CAVE and, respectively, regular personal computers. Both ActCAVE and ActPC are built on top of FreeVR [85] Details of the available core software components of the platform are provided next.

5.2 Server Resources

5.2.1 User Interface Wizard (ServGUI)

Figure 5.2 presents the use case diagram of the CAVEMANDER Wizard, or ServGUI component. The Instructor (or, in UML terminology, *the actor*) can interact with this wizard by creating a scene, creating a scenario, running a simulation, running a playback, or running a communication hub. All these use cases are further discussed with the support of several GUI snapshots in the following subsections. Note that, as indicated earlier, although the Wizard has the graphical interface for these functions, running a playback and running a communication hub (for connecting with the real-world that exists outside the CAVE) are not implemented at this time, they being beyond the scope of the work described here.

Figure 5.3 depicts two user interface tab widgets for selecting a task: the *Construction* tab and the *Execution* tab. The *Construction* tab provides the user with button widgets to create new scenes, create new scenarios, modify an existing scene, and modify an existing scenario. The *Execution* tab provides three button widgets to run the server program. These buttons include the *Simulation*, the *Playback*, and the *Communication hub*. Clicking on any button on either the Construction tab or the Execution tab will lead to a specific wizard window, which is responsible of supporting the task related with the clicked button.

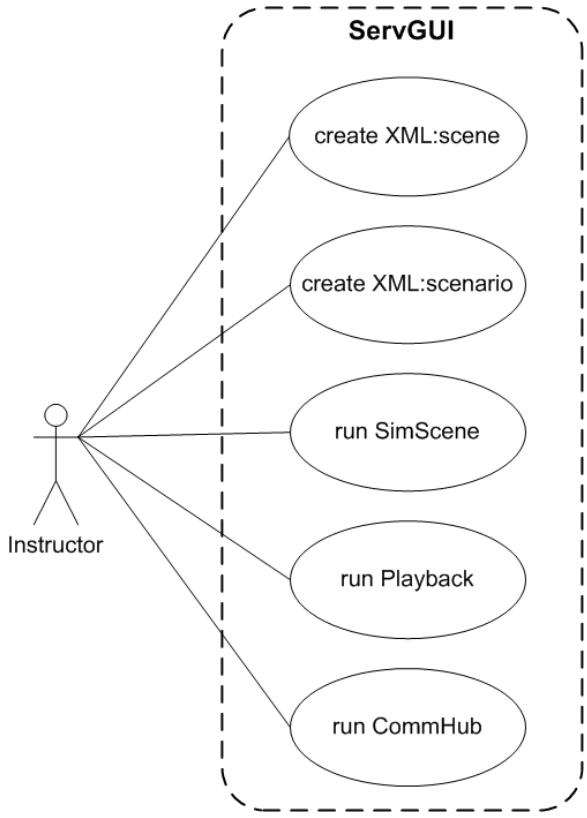
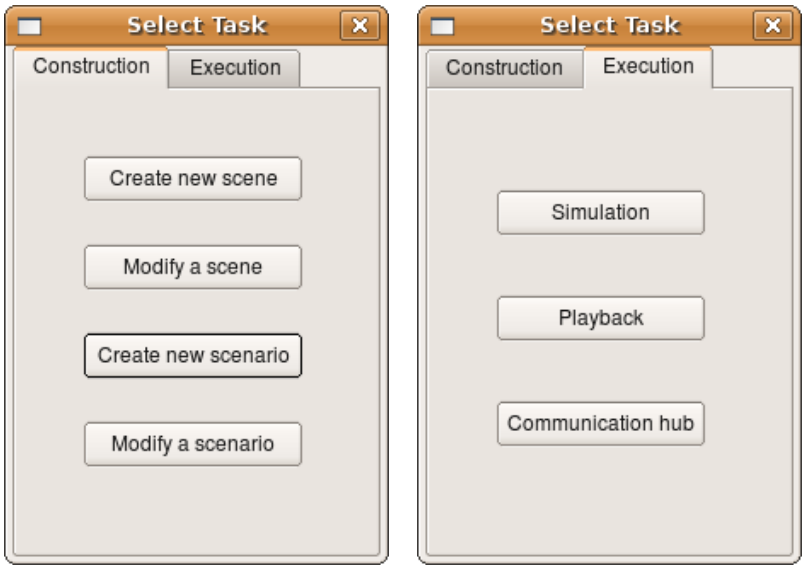


Figure 5.2: Use Case Diagram of the CAVEMANDER Wizard (ServGUI)



(a) Construction Tab (b) Execution Tab

Figure 5.3: CAVEMANDER Wizard — Widgets for Task Selection

Wizard functions to define a scene

For example, when the user clicks on the *Create new scene* button, a window will be displayed, as shown in Figure 5.4. Here, the user starts by providing a name and a description of the scene. The user can then click on either the right arrow button at the bottom-right side of the window or on the *type* tab to go to the next tab, as shown in Figure 5.5.

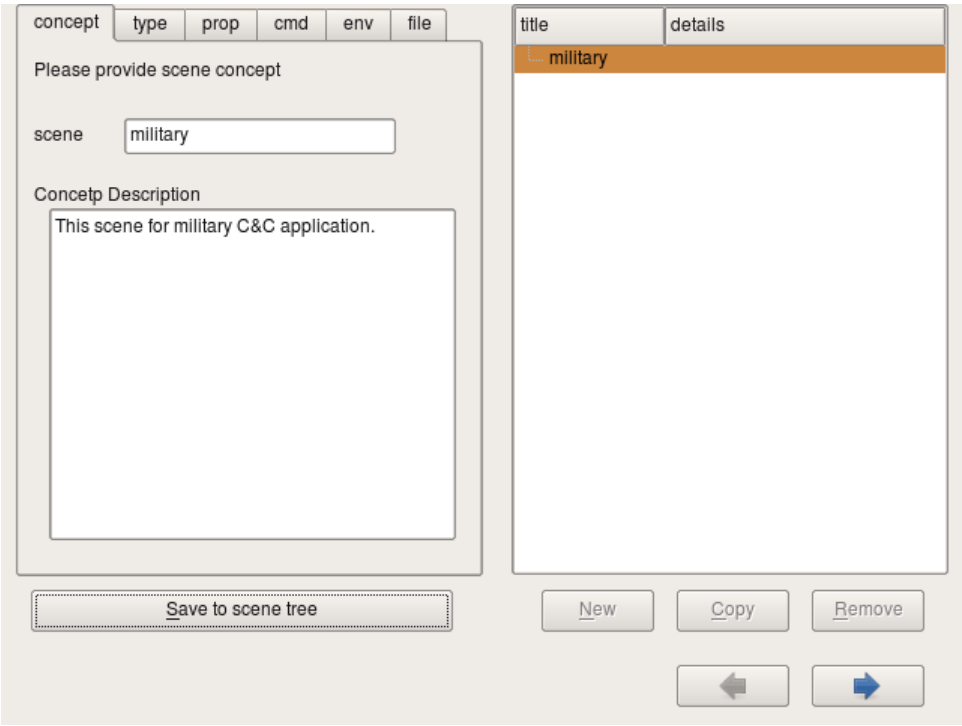


Figure 5.4: CAVEMANDER Wizard — The Scene Concept Tab

The *type* tab gives the user the ability to create new unit types to be included in the scene. For each unit type, the user has to provide a unique name, select a 3D model file from an existing list of 3D models, and include the unit type in the scene tree by clicking on the *Save to scene tree* button. The user is provided with the ability to add, copy, or remove as many unit types as needed by the use of *New*, *Copy*, and *Remove* buttons located underneath the scene tree widget.

Once done with creating the tree of unit types, the user can provide properties for each unit type. This can be done on the *prop* (properties) tab, as shown in Figure 5.6. Each

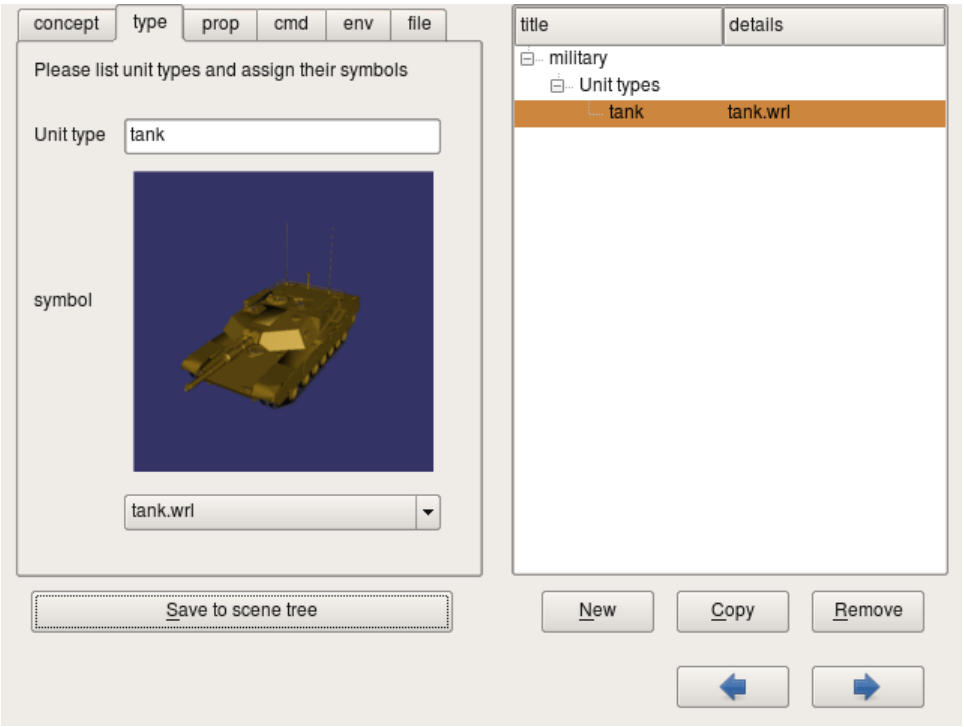


Figure 5.5: CAVEMANDER Wizard — The Scene Type Tab

property must be given a unique name, a data type (integer, float, or string), an initial value, a maximum value, and a minimum value.

The next step for the Wizard user is to provide each unit type with the necessary commands related to it. This can be done on the *cmd* (commands) tab, as shown in Figure 5.7. The user has to give each new command a unique name and a message type. Commands could have one of the following three message types:

- Cmd only — no messages after the command, e.g. *stop*.
- Cmd + position(s) — this command is followed by one or more position(s), e.g. *moveTo 340, 330*
- Cmd + target(s) — This command is followed by one or more unit(s), e.g. *attack tank02*

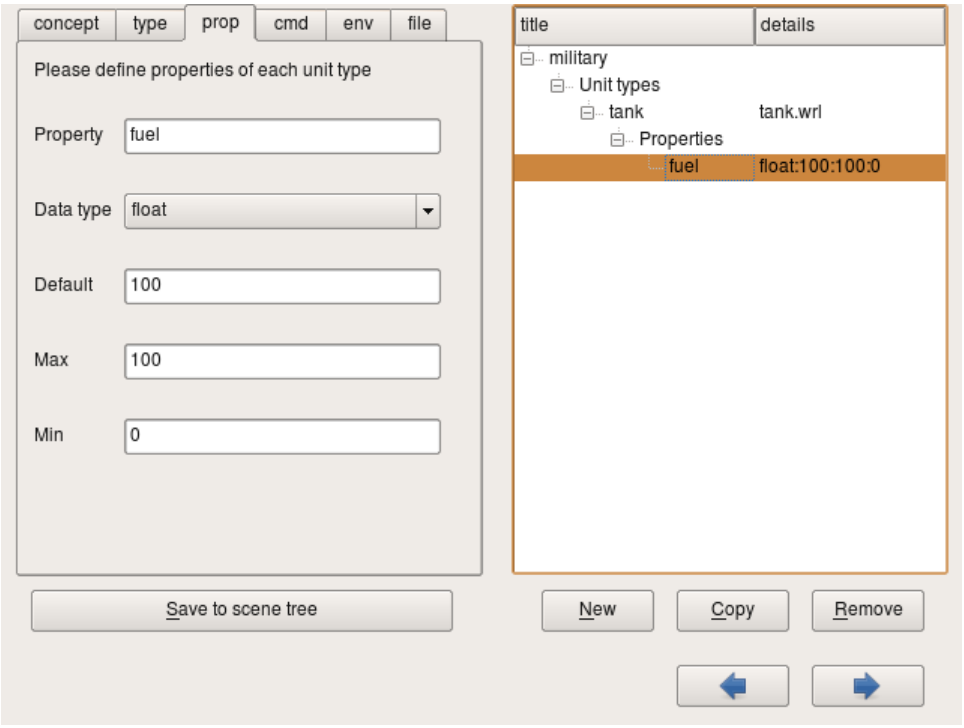


Figure 5.6: CAVEMANDER Wizard — The Scene Properties Tab

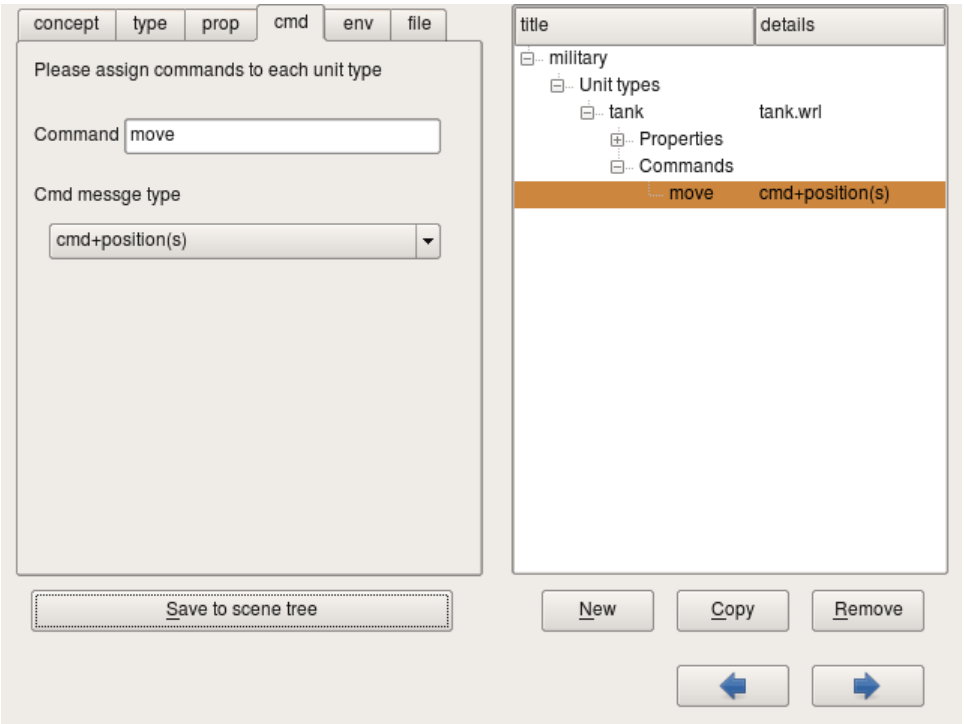


Figure 5.7: CAVEMANDER Wizard — The Scene Commands Tab

After creating the unit types and providing each unit type with its properties and commands, the user can define the environment factors of the scene on the *env* (environment factors) tab, as shown in Figure 5.8. Each environment factor requires a unique name, a data type, a default initial value, a maximum value, and a minimum value. The user can add as many environment factors as needed to be included under the EnvFactors component of the scene tree displayed on the right side of the *Creating a new scene* window.

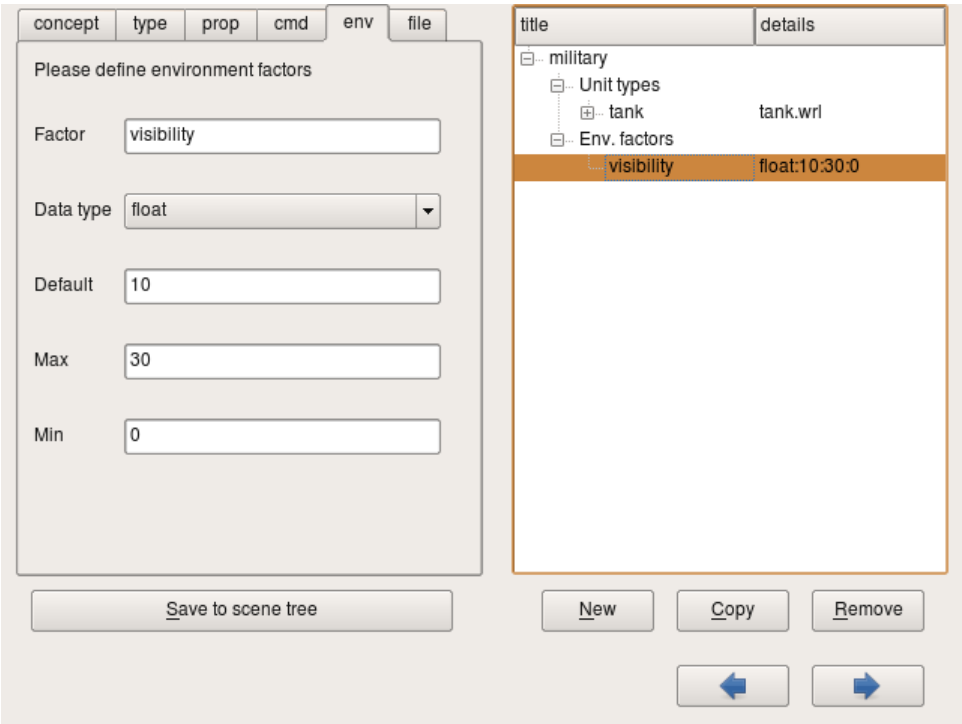


Figure 5.8: CAVEMANDER Wizard — The Scene Environment Factors Tab

Finally, the user can save the scene tree to a scene file that is generated as an XML scene file, as shown in Figure 5.9. The extension for such files is "scn".

Wizard functions to create a scenario

Creating a new scenario involves several steps as well. The first step is to give the scenario a unique name and a description, as shown in Figure 5.10.

The next step in creating a scenario is to provide the configuration data of the scenario.

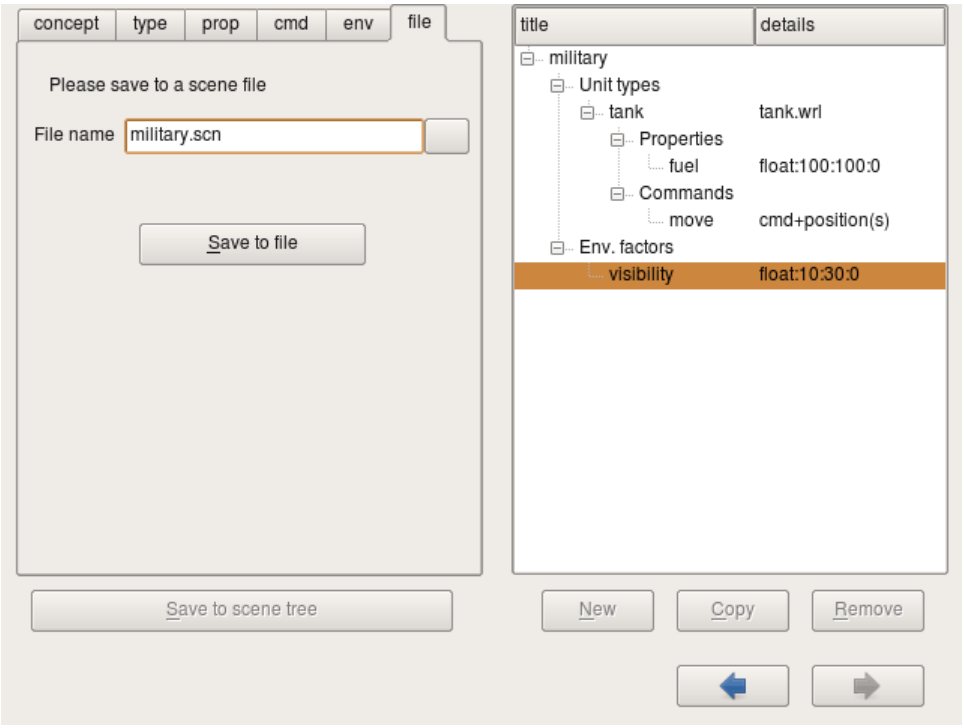


Figure 5.9: CAVEMANDER Wizard — The Scene File Tab

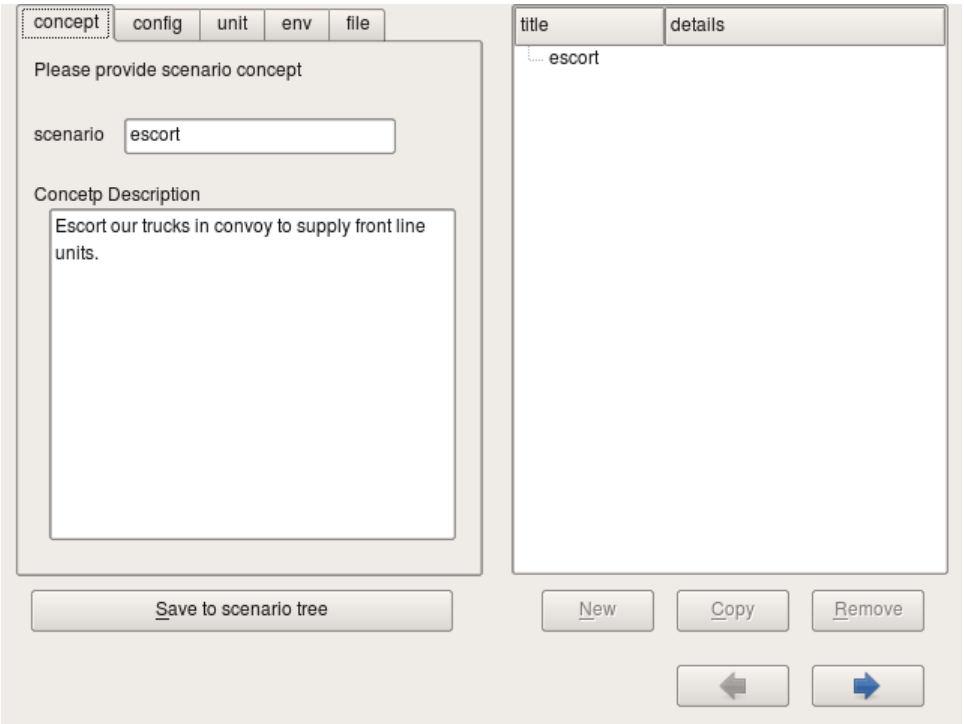


Figure 5.10: CAVEMANDER Wizard — The Scenario Concept Tab

Figure 5.11 shows the *config* (configuration) tab, which gives the user the ability to select a scene file from a list of existing files, select a terrain file, provide a log file name to store communication messages for playback purposes, and provide a socket communication port number.

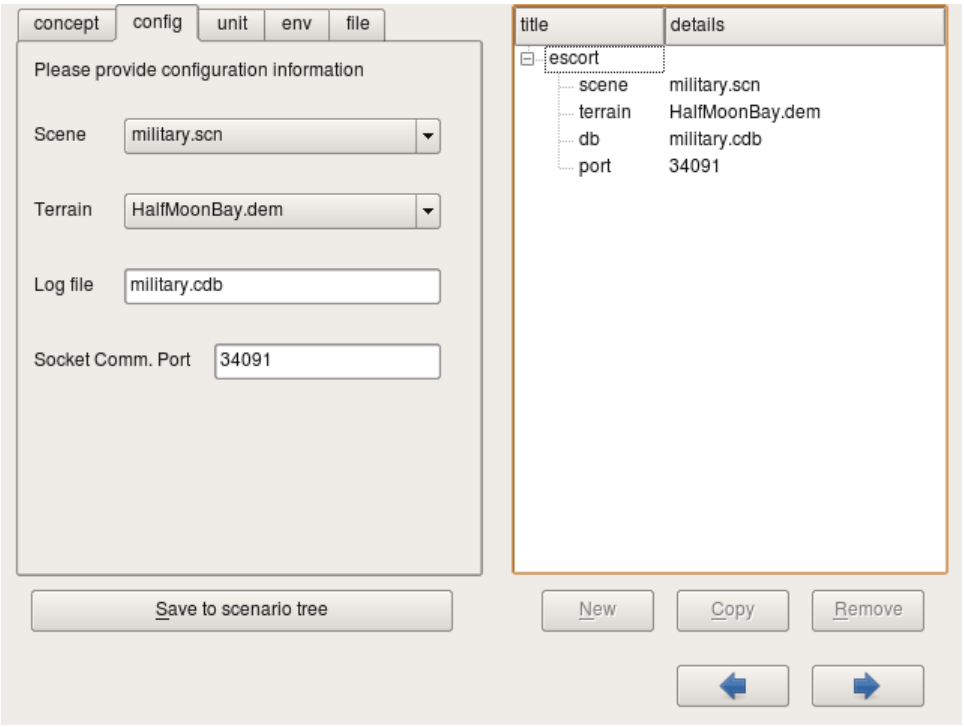


Figure 5.11: CAVEMANDER Wizard — The Scenario Configuration Tab

Figure 5.12 shows the *unit* tab. Here, the user can add all the units involved in the scenario. For each unit, the user has to provide a unique name, a unit type, and an ID number of the commander in charge of that unit (this allows CAVE simulation with more than one commander). Also, if needed, the user has the ability to change the initial value of each property.

To customize each scenario, the user can also change the initial value of each environment factor if needed, as shown in Figure 5.13.

As with the scene file, the user finishes creating a new scenario by saving the scenario tree

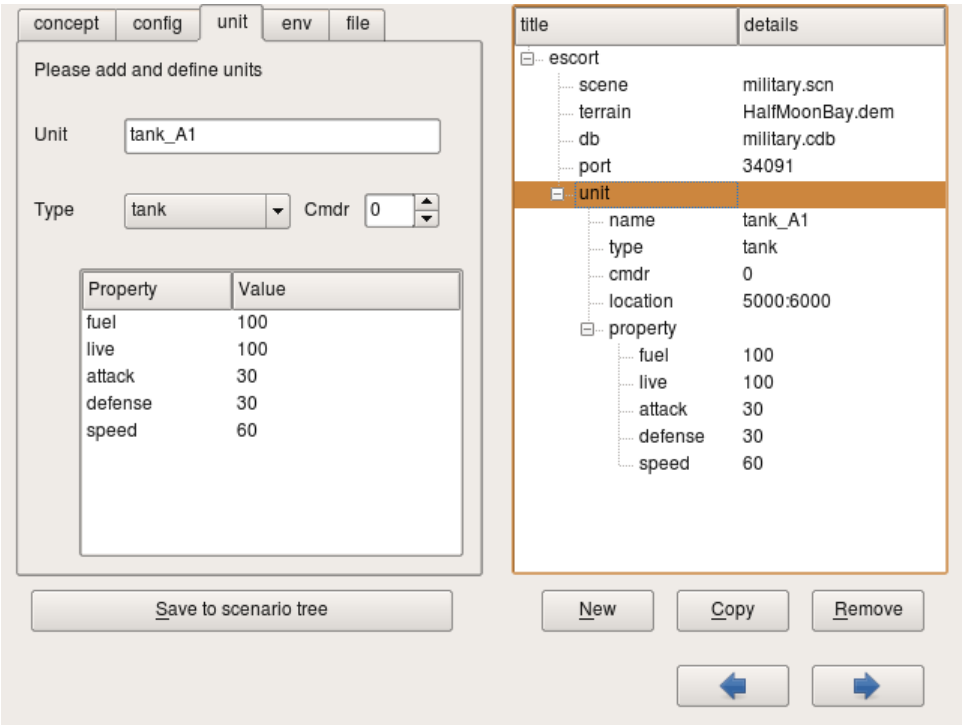


Figure 5.12: CAVEMANDER Wizard — The Scenario Units Tab

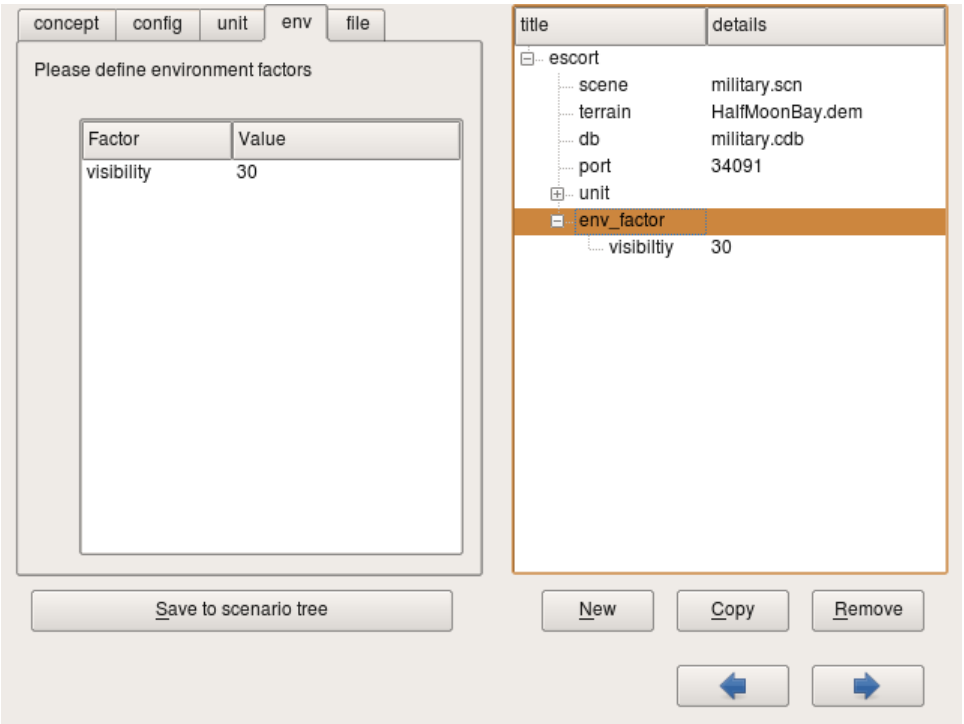


Figure 5.13: CAVEMANDER Wizard — The Scenario Environment Factors Tab

to a scenario file that is generated as an XML file, as shown in Figure 5.14. The extension of such files is “sco”.

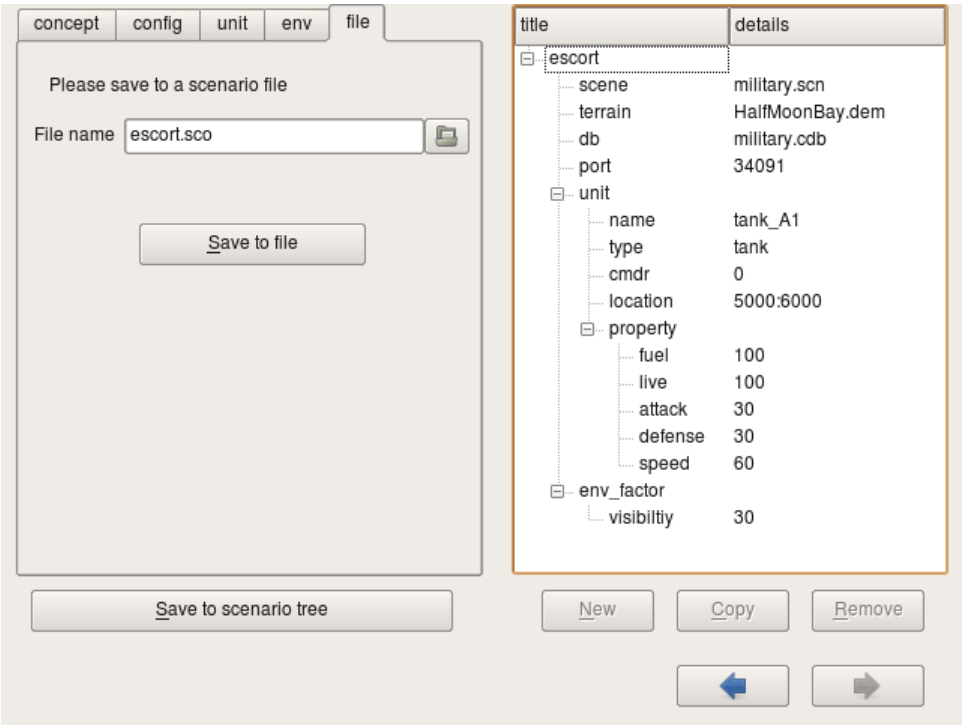


Figure 5.14: CAVEMANDER Wizard — The Scenario File Tab

5.2.2 Scene Simulator (SimScene)

Figure 5.15 presents the use case diagram of the SimScene component of the CAVEMANDER architecture. Two actors can interact with the use cases of the SimScene: the (CAVE) System and the Instructor. The System actor interacts with SimScene by reading a scenario, waiting for a connection, accepting a connection, performing a simulation, responding to a command, distributing information, or recording a communication message. The Instructor, on the other hand, interacts with SimScene by selecting a scenario, running a scenario, changing the time speed, pausing/resuming a scenario, or stopping a scenario.

Figure 5.16 shows an excerpt of the classes and templates needed for building a simulation.

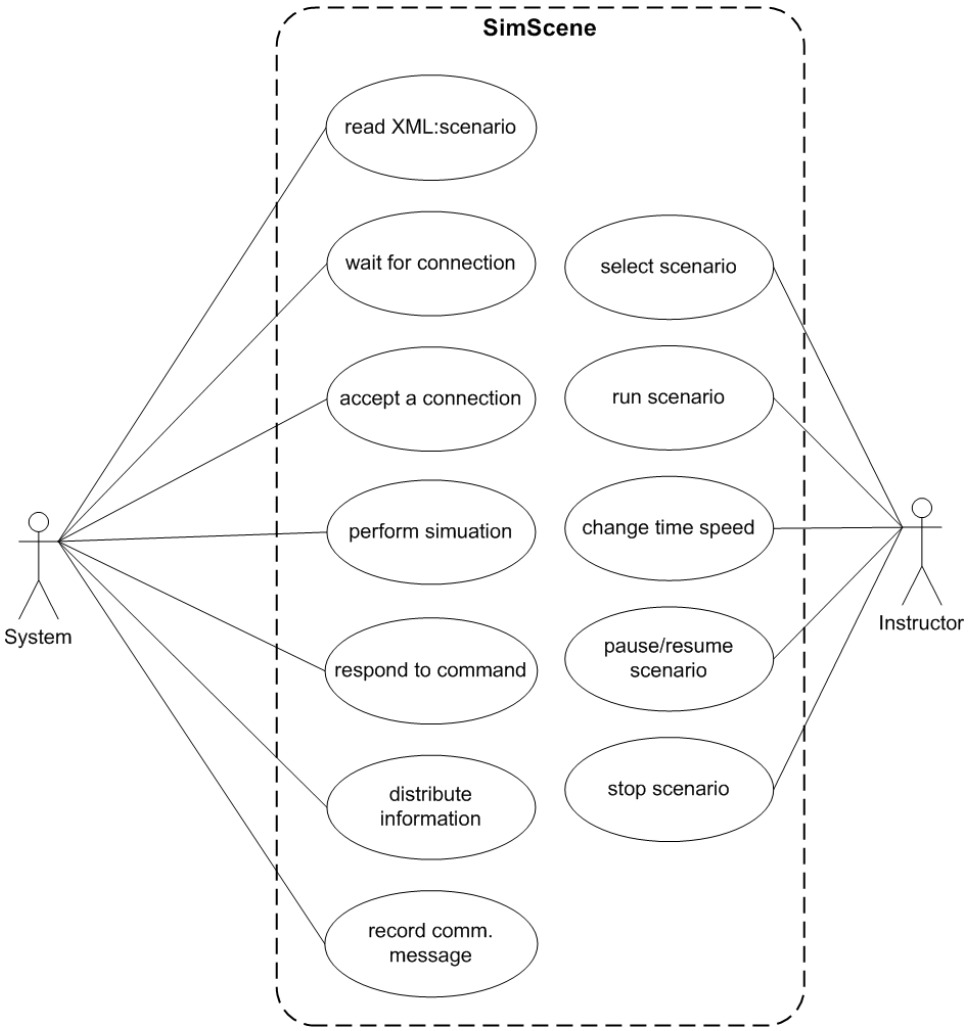


Figure 5.15: Use Case Diagram of SimScene Component

It consists of several classes including: UnitBase, UnitType1, UnitType2, up to UnitTypeN, EnvFact, XmlParser, SimTime, SimWorld, msgHandle, ServerSock, and Terrain. Some of these classes are already included in the library, whereas others are either modified from a template or need to be written by a software developer. As shown in the figure, UnitType classes are derived classes that inherit their attributes and behavior from the UnitBase class.

When a scenario is created and ready to be run, it can be loaded from the Wizard's scenario window, as shown in Figure 5.17. Once a scenario is loaded, the data representing that scenario will be displayed as a tree on the right side of the window. The user can then manage the execution of the simulation scenario by playing, pausing, resuming, fast forwarding, rewinding, or stopping the simulation. The simulation time is shown in seconds elapsed from the start of the simulation. The user has the ability to change the time scale by using either the fast forward button or the spin box, which also indicates the current time scale. Furthermore, sent and received communication messages are displayed in the textbox at the bottom-left side of the window and written into a log file.

5.2.3 Simulation Replay (PlayBack)

Figure 5.18 shows the use case diagram of the Playback server component. As with the SimScene component, two actors can interact with the use cases of the Playback software: the (CAVE) System and the Instructor. The System actor can interact with the PlayBack by reading a scene, reading a scenario, waiting for a connection, accepting a connection, reading a log file, scheduling a message, or sending a message. The Instructor can interact with the Playback software in the same way he or she can interact with SimScene, which includes selecting a scenario, running a scenario, changing the time speed, or stopping a scenario.

The graphical user interface for running a playback is the same interface used for SimScene

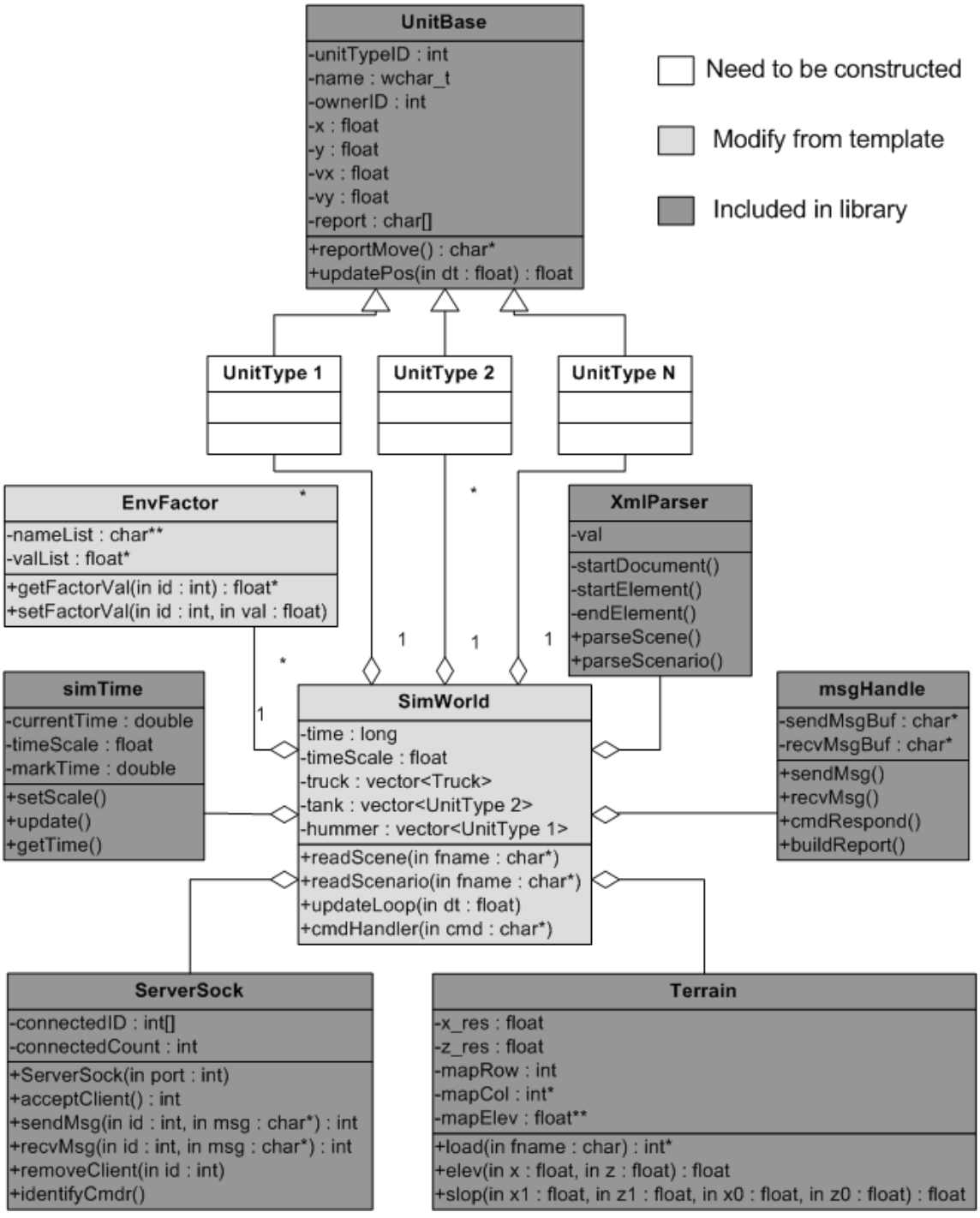


Figure 5.16: SimScene Classes and Templates for Building Simulations (Excerpt)

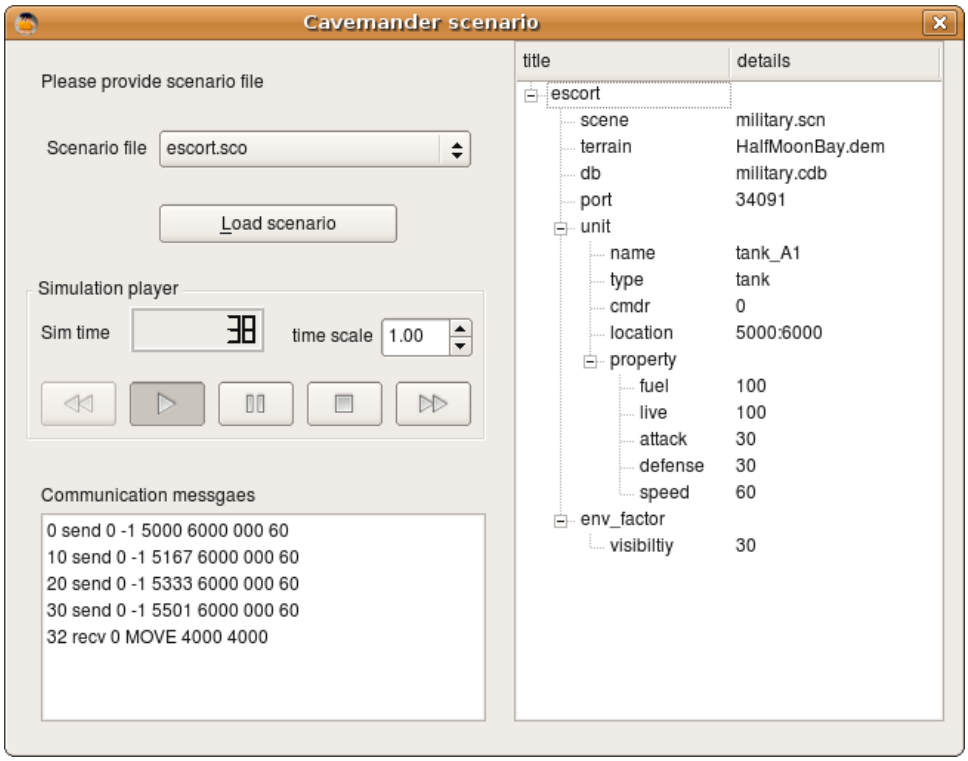


Figure 5.17: CAVEMANDER GUI for Running Scenarios

(Figure 5.17) but with the backward button being enabled.

5.2.4 Communication Hub (CommHub)

Figure 5.19 shows the use case diagram of the CommunicationHub component of the CAVEMANDER architecture. Several use cases have been already described in previous subsections. New use cases include defining a communication media, changing a communication media, running a communication hub, and stopping a communication hub. The instructor is the user who has the ability to interact with these new use cases.

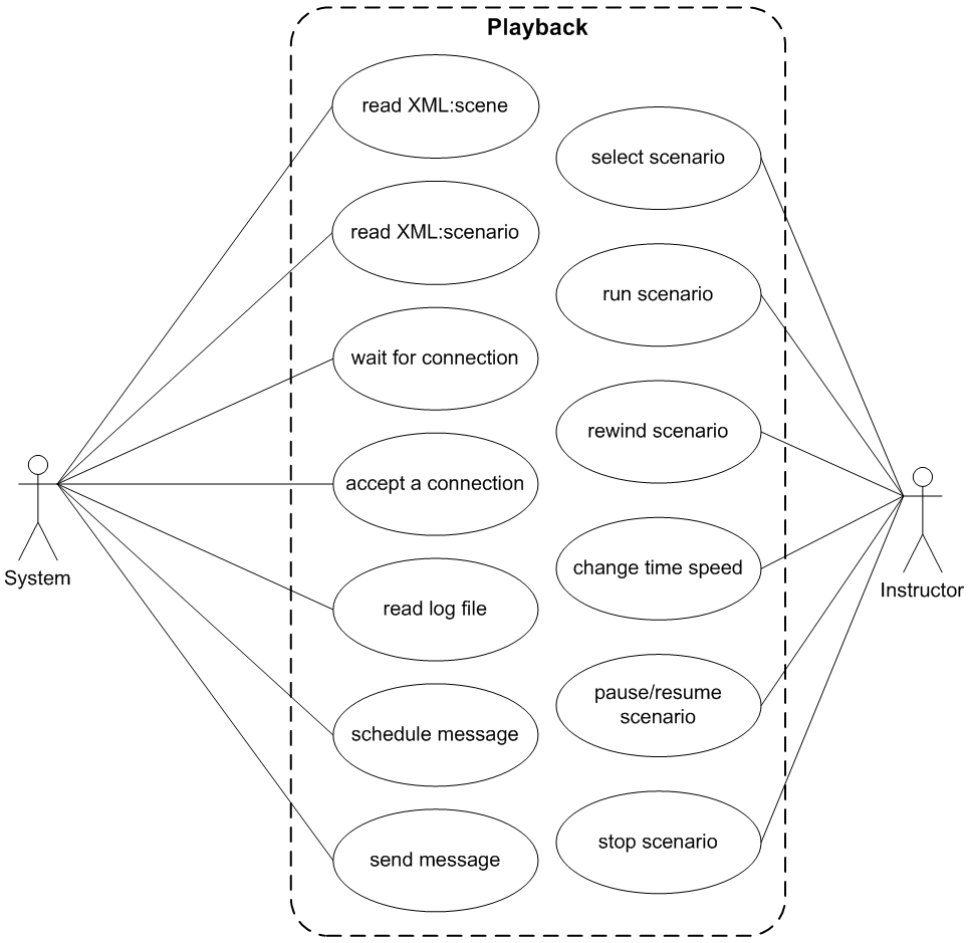


Figure 5.18: Use Case Diagram of Playback Component

5.3 Client Resources

5.3.1 Interacting with the Simulation Scenario in CAVE (ActCAVE)

As discussed earlier, the **Client** is a core component of the CAVEMANDER software platform. The ActCAVE software is in charge of managing the interaction with a simulation inside the CAVE. Figure 5.20 shows the use case diagram of ActCAVE, which can be also used for ActPC, the only difference being that in the latter case the System is a personal computer (PC) rather than a CAVE. The actors interacting with ActCAVE are the Commander and the (CAVE) System. The commander can travel on terrain, change terrain

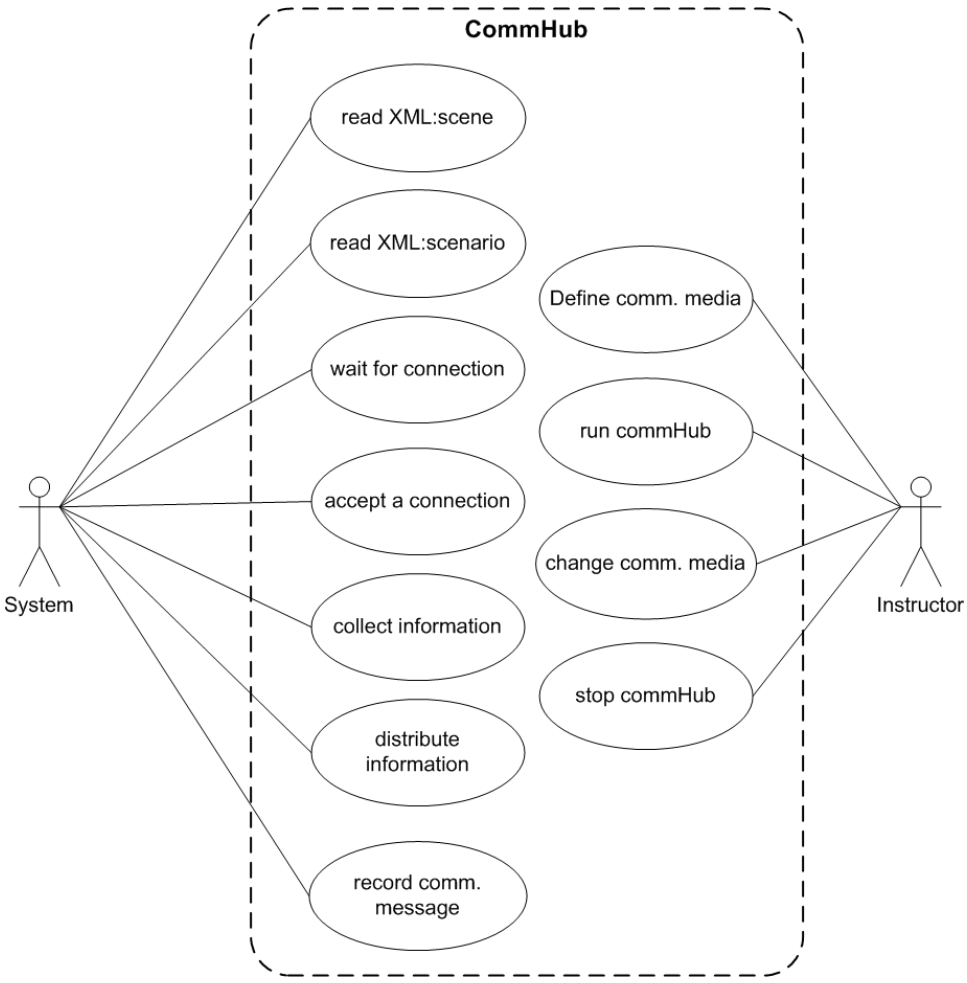


Figure 5.19: Use Case Diagram of CommHub Component

scale, monitor units, select the main property to be shown on the top of units, select a unit, select a location, or issue a command. The System is given the ability to read a scene file, read a scenario file, generate a terrain, connect to the server, or receive information.

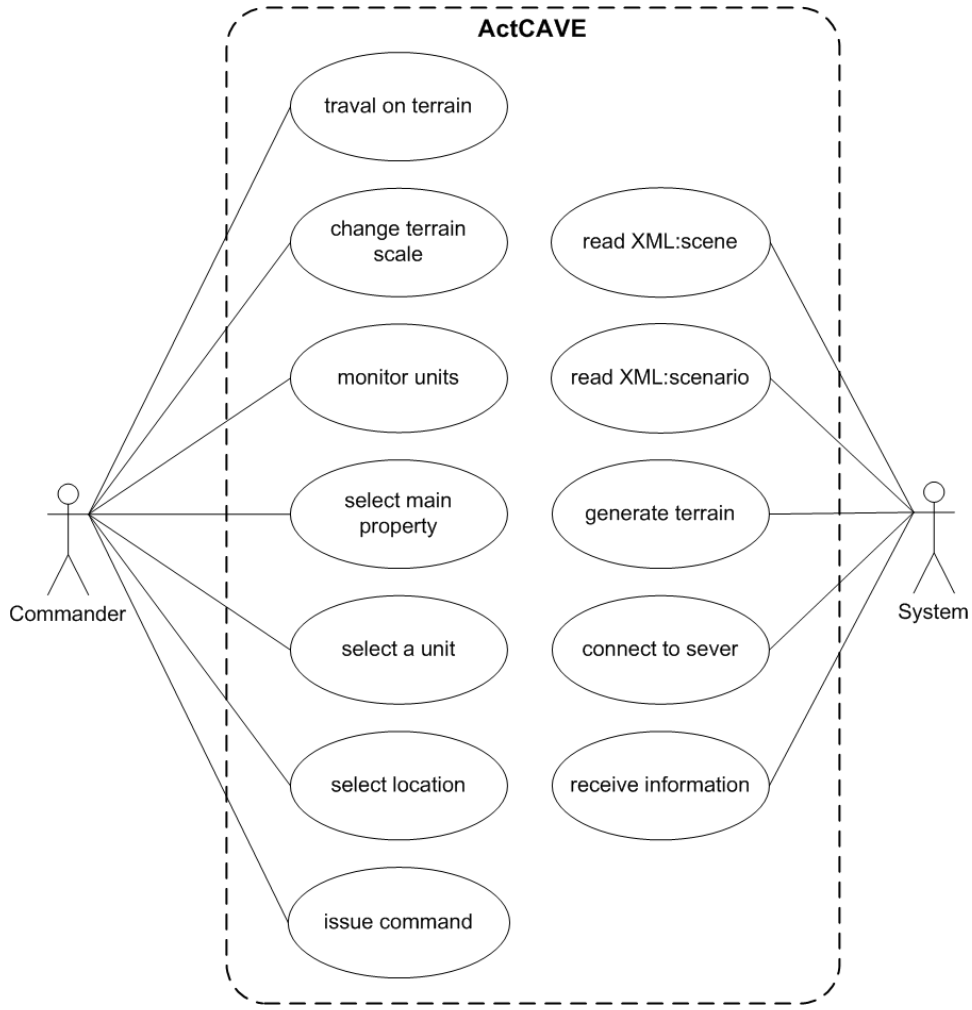


Figure 5.20: Use Case Diagram of ActCAVE and ActPC Components

The class diagram of ActCAVE is shown in Figure 5.21. It consists of twelve classes that collaborate to perform simulation tasks. The classes include *ModelType*, *ImageType*, *EnvFactor*, *ServerSock*, *Terrain*, *ElementType*, *CAVEWorld*, *Cmd*, *Element*, *Trail*, *MsgHandl*, *and XmlParser*.

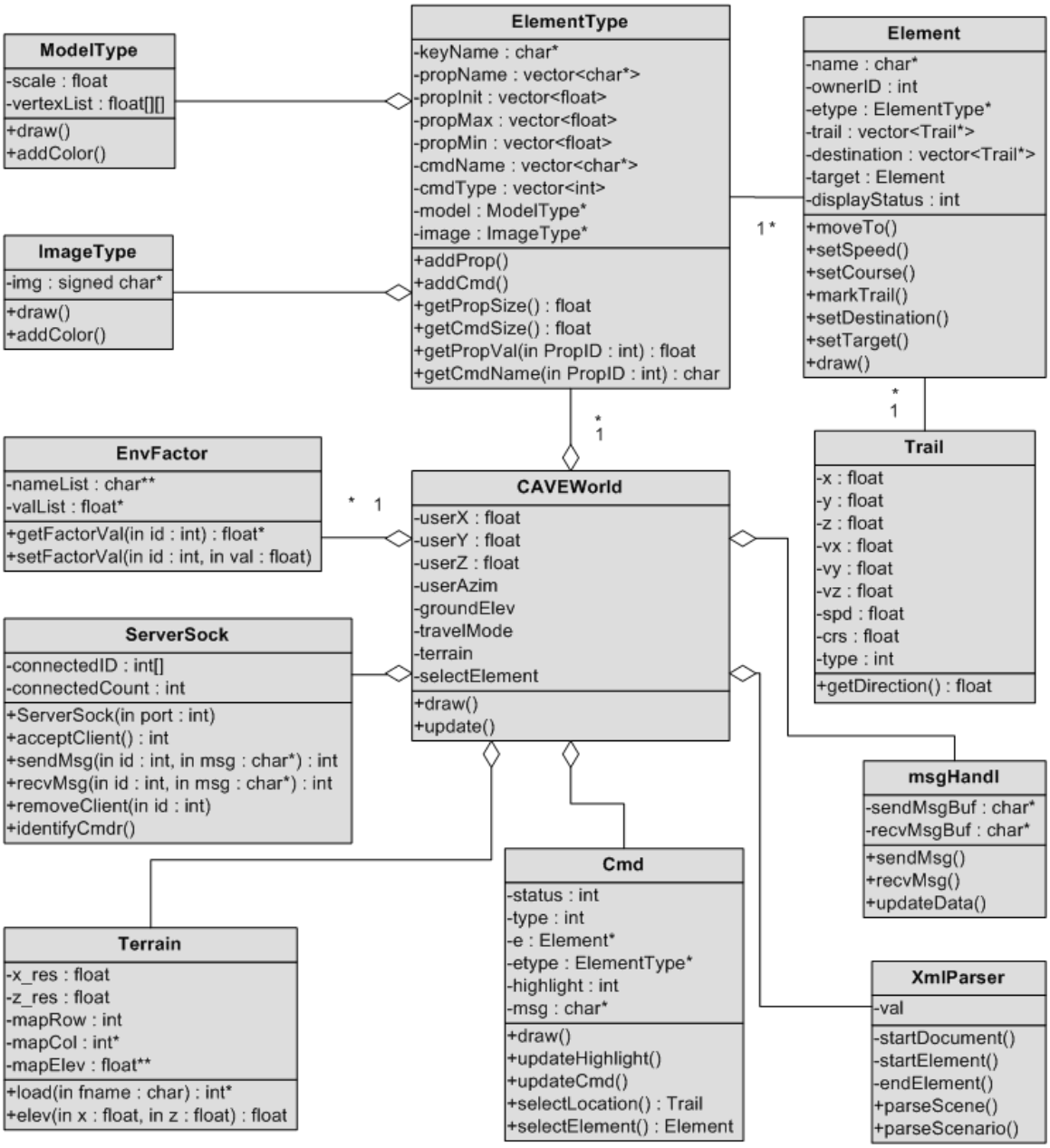


Figure 5.21: Class Diagram of ActCAVE (Partial)

5.3.2 Interacting Simulation on PC (ActPC)

ActPC provides the same attributes and behavior of ActCAVE except that it displays the simulation scenario on a PC screen instead of inside the CAVE.

5.3.3 User Modes in CAVE

While running a simulation scenario inside the CAVE, the user can be in one of the three modes shown earlier in Figure 4.12. The user starts by entering the travel mode. He or she can move forward, move backward, turn left, or turn right. The user is also given the choice to fly, walk, or ride-along. The user can change his or her state to the command mode. Here he or she can point to a unit to highlight it and press a button to activate a command. The user can also be in the viewing mode, where he or she can either scale up and down, or select an element. Scaling up and down is done through the use of a joystick, whereas selecting an element is done by pointing to that element and then pressing a key on the wand.

Chapter 6

EXAMPLE SIMULATION SCENARIO

This chapter demonstrates an example of creating and running a simulation scenario using the proposed method described in Chapter 4, and the software platform presented in Chapter 5. A scene definition and a scenario is built using the provided CAVEMANDER GUI Wizard, and the simulation is executed in the CAVE immersive environment, where a commander interacts directly with the objects of the virtual world. The chapter presents in some detail the description of the representative scenario example, covering all the steps involved in achieving C&C operation in the simulated environment using CAVEMANDER. However, for the sake of simplicity only few snapshots of the Wizard or details of the artifacts are presented. Furthermore, the scenario is somewhat simplified and only several relevant states in its execution are presented.

6.1 Simulation Application Description

We developed a military C&C application simulation that allows a commander to perform C&C tasks inside the CAVE system. In this environment, such simulation can be used as a training or experimentation of specific military operations. During the execution of the simulation, the commander has to observe, evaluate, plan, and issue commands to his or her task force. We created a representative scenario to demonstrate the utility of our proposed system, describing its states and steps involved in executing a training simulation, which is to escort and distribute supplies to frontline units. The performance of a commander is measured by the number of frontline units that receive the supplies, and the time it takes to achieve this goal.

Since this is an initial approach of building C&C applications for CAVE using CAVEMANDER, we have kept the scenario rather simple, in order to clarify the concept and the operations. This scenario includes only three types of military units: Hummer, Tank, and Truck. Each unit type has different abilities and properties, where the Hummer has the highest speed and a reasonable attack factor, Tank is the slowest vehicle but has the greatest attack factor, and Truck has a moderate speed and very low defense power but it is the only vehicle that can carry supply. The following section items describe the overall situation of the scenario, indicate the mission and strategy that are given to the commander, and outlines the status of the enemy force.

Situation

Our two different platoons are stationing half mile away from each other in frontline areas, and they are both located about five miles from our base. Both of them are short of supplies (e.g. food, water, ammunition, etc.), where we estimate that the supplies will last only for five days maximum. Thus, we need to deliver new supplies to both platoons as soon as

possible. However, our intelligent units have identified the presence of enemy forces in the surrounding area.

Mission

We need to send two trucks with supply loads to each platoon successfully, avoiding the enemy force and reaching the frontline units before they run out of supplies. If any of the trucks is destroyed or does not deliver the supply load within the given duration, the mission is considered to be failed.

Enemy forces

According to the information provided by our intelligence units, there are several enemy tanks patrolling the area between our platoons and the base.

Strategy

We plan to form two convoys consisting of two tanks and two trucks full of supply loads for each of the two platoons. Each convoy travels independently to their designated platoon. We also utilize three Hummer high mobility vehicles to search the safety of the areas, securing the paths for the convoys.

6.2 Scene Definition Using the GUI Wizard

For simplicity, only few on the many possible details involved in scene definition are presented. However, for illustration purposes, the main units involved in the scenario are described using Tables 6.1, 6.2, and 6.3. Also, in Table 6.4 details of environment param-

Table 6.1: Characteristics of the Hummer Unit Type

Hummer			
Model: hummer.3ds			
Properties:	**Initial**	**Max**	**Min**
live	100	100	0
fuel	100	100	0
defense	50	100	0
attack	50	100	0
speed	0	100	0
Command:		**Supplement**	
Move		+ position	
Attack		+ target	
Patrol		+ positions	
Follow		+ target	

eters used are presented. A sample snapshot of the CAVEMANDER Wizard during scene definition is shown in Figure 6.1.

6.3 Simulation Build Up Using Templates and Library Resources

To illustrate the simulation build up activity an excerpt of the simulation class diagram is presented in Figure 6.2. Notice that some of the code has been reused (in dark background), some was created using templates (lighter background), and some was written from scratch (white background). Again, to keep things simple only some of the details of the code included in the simulation SimScene package are presented in Figure 6.2.

Table 6.2: Characteristics of the Tank Unit Type

Truck			
Model: truck.3ds			
Properties:	**Initial**	**Max**	**Min**
live	100	100	0
fuel	100	100	0
defense	20	100	0
attack	20	100	0
speed	0	100	0
Command:		**Supplement**	
Move		+ position	
Approach		+ target	
Follow		+ target	
Load		+ none	
Supply		+ target	
Self destruction		+ none	

Table 6.3: Characteristics of the Truck Unit Type

Truck			
Model: truck.3ds			
Properties:	**Initial**	**Max**	**Min**
live	100	100	0
fuel	100	100	0
defense	20	100	0
attack	20	100	0
speed	0	100	0
Command:		**Supplement**	
Move		+ position	
Approach		+ target	
Follow		+ target	
Load		+ none	
Supply		+ target	
Self destruction		+ none	

Table 6.4: Environment Factor: Visibility

Environment factors			
Factors:	**Initial**	**Max**	**Min**
visibility	10	30	0

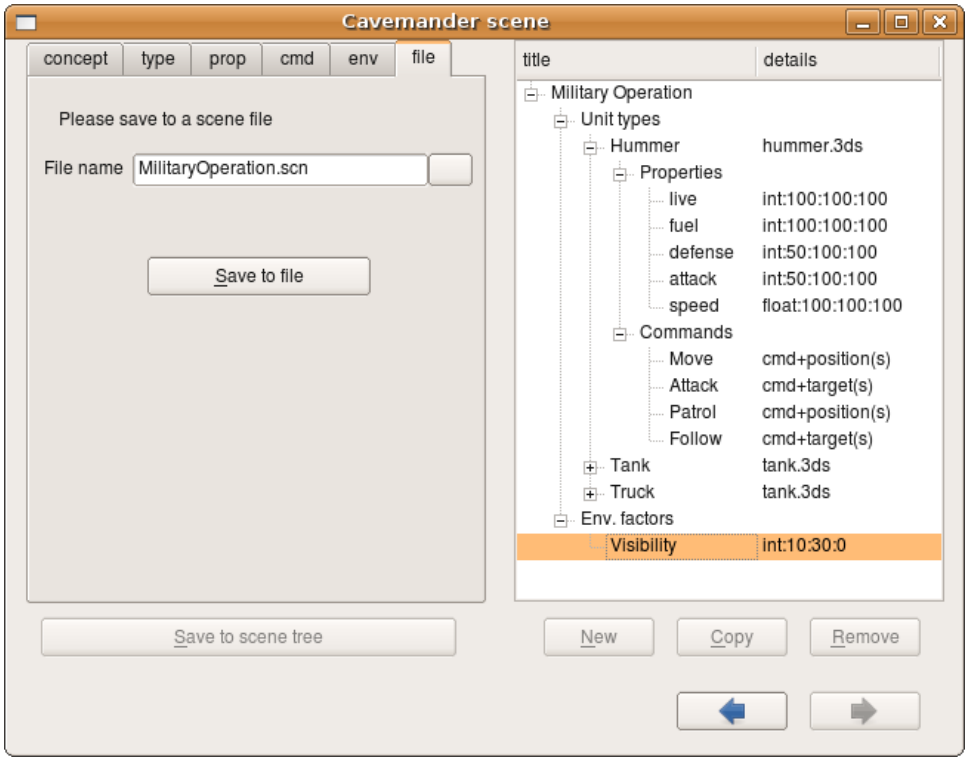


Figure 6.1: Sample GUI Wizard Snapshot from Scene Definition

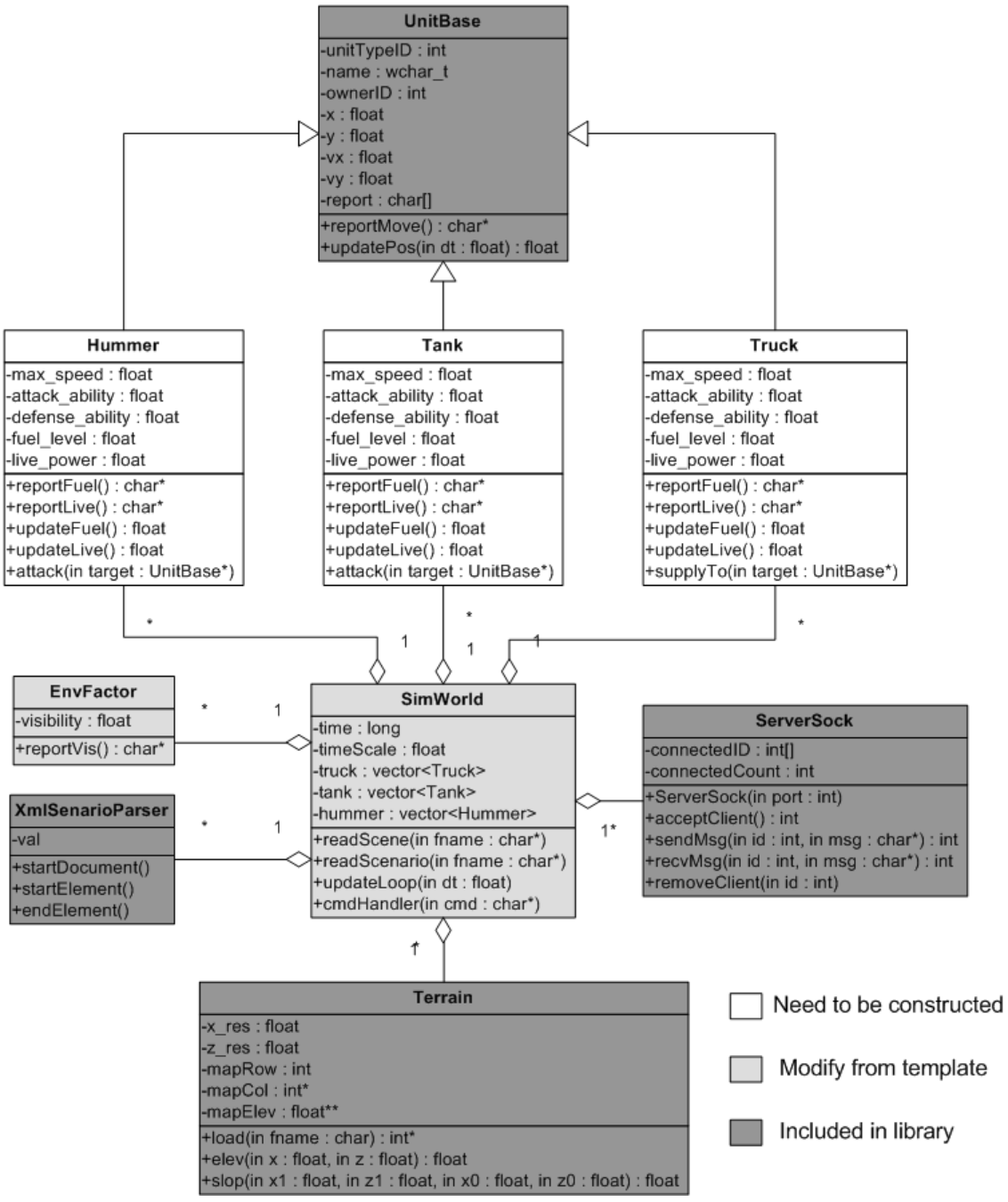


Figure 6.2: Excerpt form SimScene Class Diagram of Simulation Code

6.4 Scenario Creation Using the GUI Wizard

During the next activity, scenario creation, specific values are given to the unit types and their parameters, as well as to the environment parameters involved in the scenario. Figure 6.3 provides a sample snapshot from the many possible during the scenario creation using the CAVEMANDER Wizard.

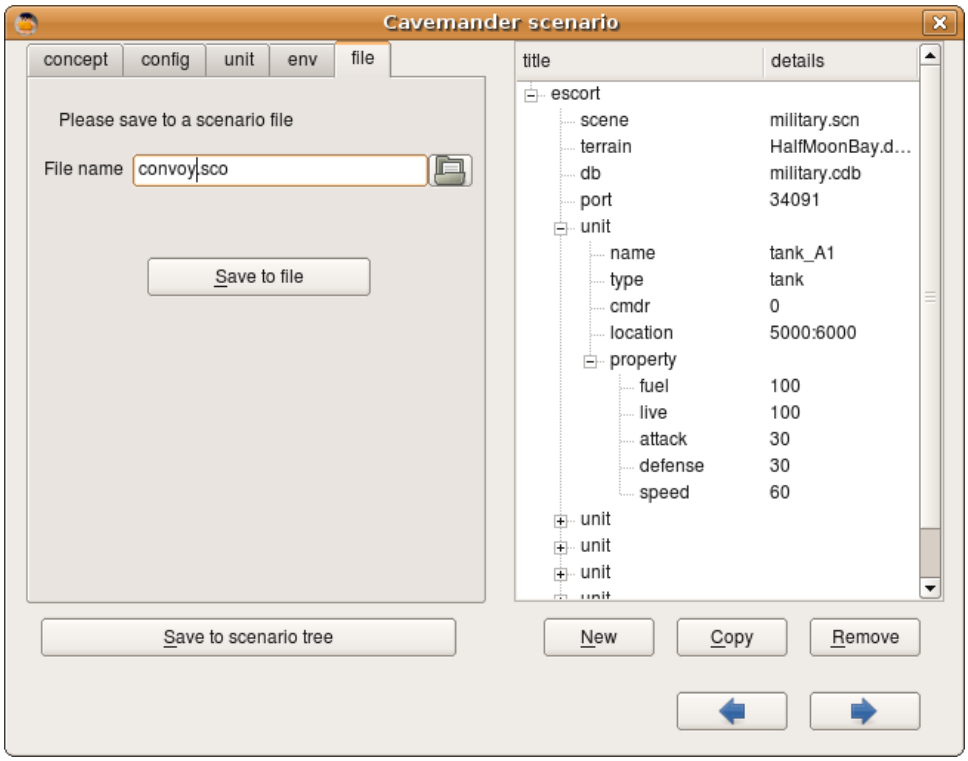


Figure 6.3: Sample GUI Wizard Snapshot from Scenario Creation

6.5 Running Scenario in CAVE Using ActCAVE

This section explains the actual execution of the scenario, describing interactions with the commander. The actions taken and the notable resultant states of the simulation scenario are listed in Table 6.5. Note that the sequence of events is only particular to this execution, which may not be the same for a different execution.

Table 6.5: Notable States throughout the Scenario

State	Description
1	Initial state
2	Load supplies to each truck
3	Form two convoys
4	Direct each hummer to search for the safety in the paths to the platoons
5	Direct each convoy to follow the safe route to the designated platoons
6	One of the hummers encounters an enemy tank
7	The enemy tank destroys the hummer
8	Assign our closest tank to approach and attack the enemy tank
9	Direct remaining hummers to cover the entire search area
10	Reroute the convoy to avoid an engagement area
11	Our tank attacks the enemy tank
12	Our tank destroys the enemy tank
13	Both convoys reach the designated platoons successfully

In the following section, snapshots of selected states are shown to illustrate various states of the situation and results of the actions taken by the commander.

Figure 6.4 presents the view at the initial state when the simulation begins. The viewpoint is at the base, showing all the tanks, hummers, and trucks at their initial positions.

Figure 6.5 displays the view when the scenario is in State 5, where two convoys are formed with tanks and trucks, and start moving along the safe route surveyed by the hummers ahead.

Figure 6.6 depicts the view in State 6 of the scenario, when one of our hummers encounters an enemy tank while searching the area. Not that the enemy units are not visible until they fall inside the visible range of our unit.

Finally, Figure 6.7 shows the commander's view in State 10 where one of the convoys reroute the path to avoid an engagement with the enemy tank. Note that one of our tanks

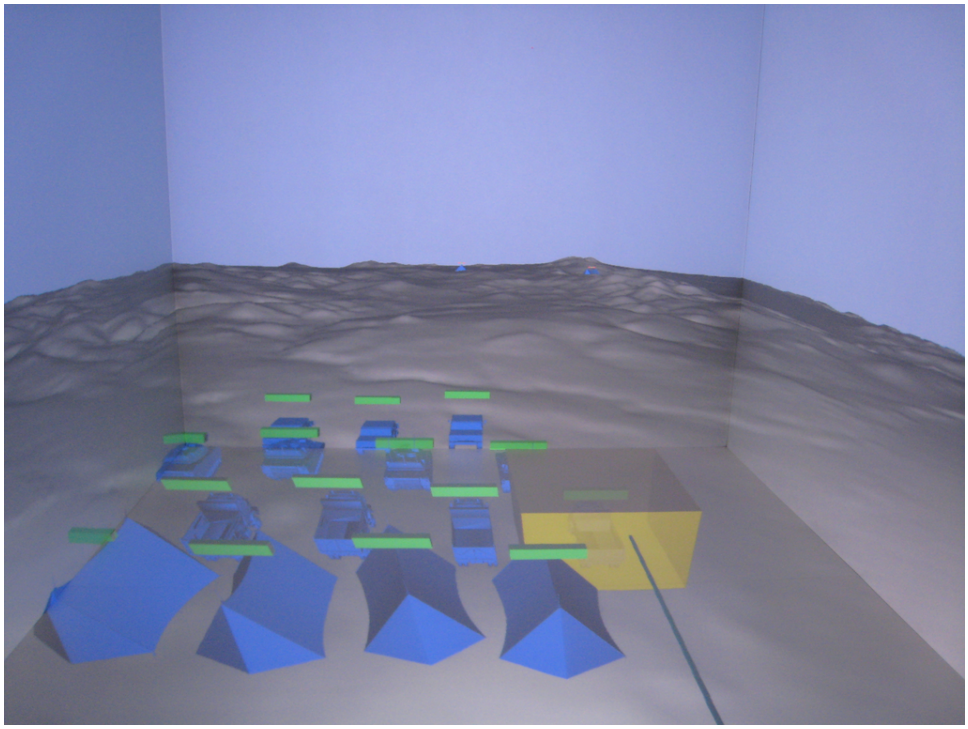


Figure 6.4: Sample Scenario Execution State — State 1 (Initial State)

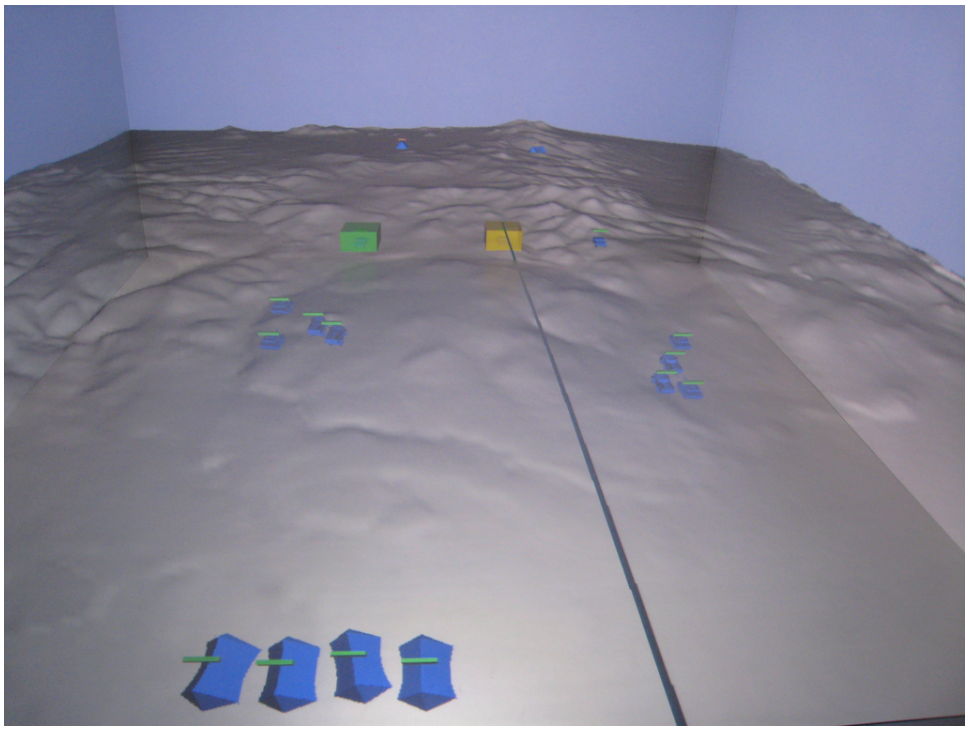


Figure 6.5: Sample Scenario Execution State — State 5 (Convoys Moving Along Secure Paths)

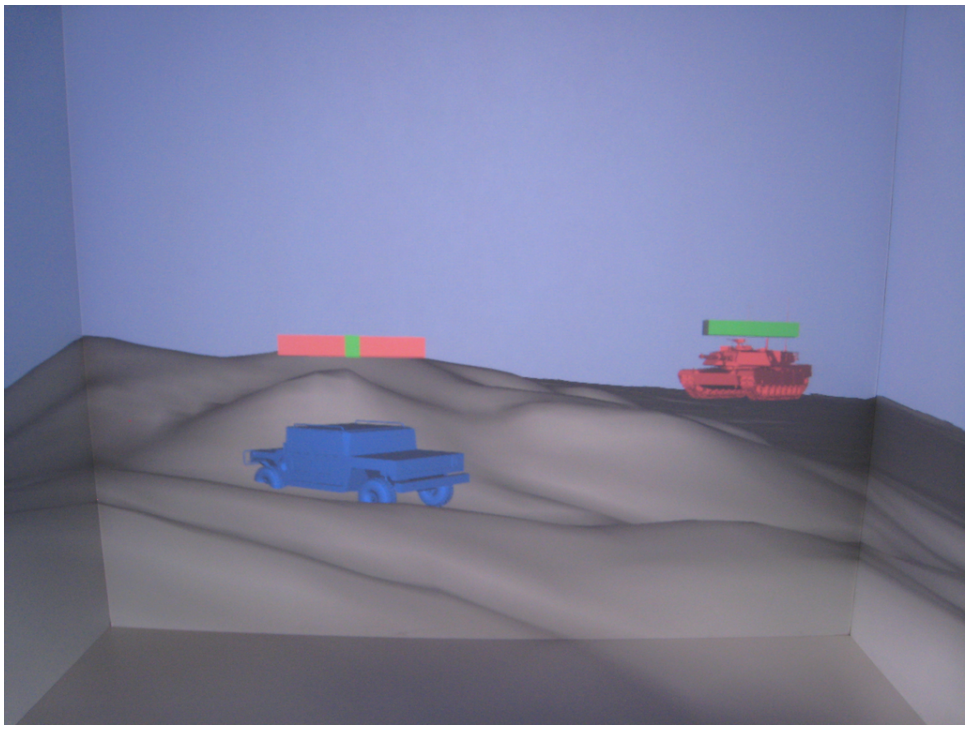


Figure 6.6: Sample Scenario Execution State — State 6 (Hummer Encounters an Enemy Tank)

is approaching to attack the enemy tank.

The above are only few excerpts from the scenario execution, intended to illustrate the use of CAVEMANDER for building and running simulation scenarios. Nevertheless, we hope that the reader can grasp from them the richness of information available to the commander during such simulations, and the capabilities that CAVE offers for C&C applications.

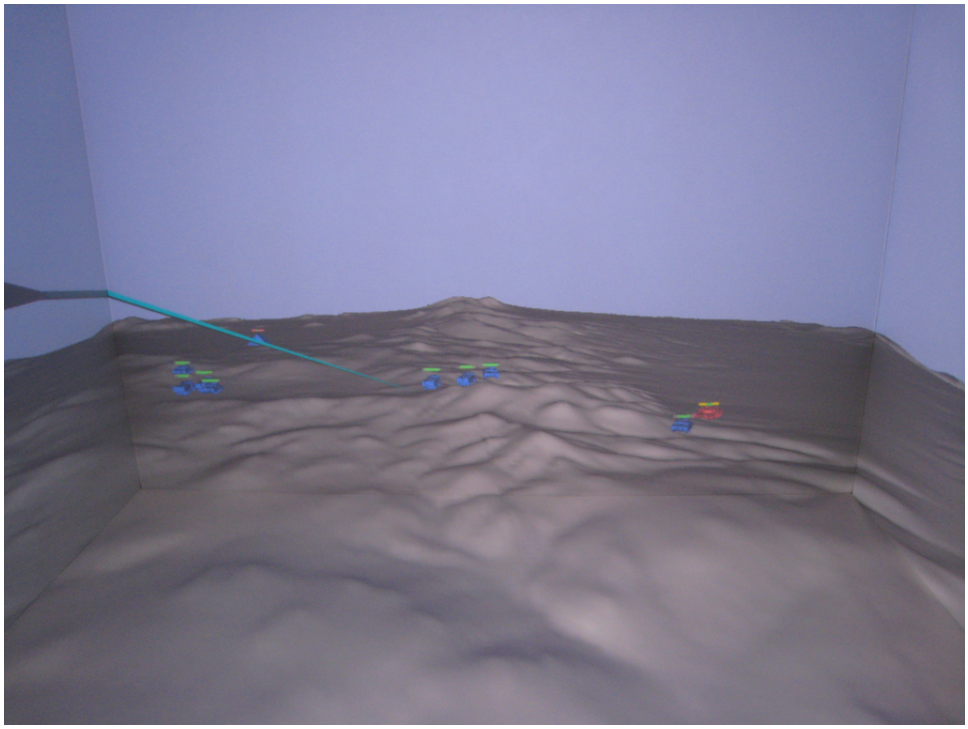


Figure 6.7: Sample Scenario Execution State — State 10 (Own Tank Approaching an Enemy Tank While a Convoy Reroutes to a Safe Path)

Chapter 7

RELATED WORK

In this chapter, several existing projects and software resources related to constructing VR applications are surveyed, followed by a comparison with our approach. In the first part of the chapter, work related to CAVE and other 3D virtual reality software tools is described in terms of several of their features. The second part of the chapter provides a discussion of various C&C systems and applications, focusing on their usability. To the best of our knowledge, the exploration of related work indicates that there is little available in terms of software development methods and related supporting resources for building CAVE-based C&C applications, a situation that CAVEMANDER has tried to address.

7.1 Work Related to CAVE and VR Software Applications

We have surveyed several existing software resources for developing CAVE and other VR applications. Among the most representative are the following.

CAVELib [59] is one of the first and the most popular libraries for developing applications designated for the immersive displays such as CAVE, providing a cross-platform API to hide the development complexity in system and device communication. CAVELib facilitates stereoscopic viewing and other common functionalities in VR applications, such as multi-processing, shared data management, and network communication. CAVELib was originally developed by University of Illinois, Chicago, and now is commercially owned by Mechdyne Corporation.

FreeVR [85] is another API library focused on development of CAVE application, sharing a lot of functionalities with CAVELib, supporting wide variety of input and output devices. FreeVR is developed to be free and open source, thus easy to be shared among VR researchers.

VR Juggler [95] is a suite of API resources that provides an environment for developing VR applications in various types of display systems, from ordinary monitor-based computer to complex multiple projection systems including CAVE. VR Juggler allows the development of portable application that can be executed on different platforms and with different devices, without any modification in the code itself. VR Juggler was started at Iowa State University and is now commercially supported by Infiscape Corporation.

CoVE [70] is a cross-platform VR Toolkit built on Open Scene Graph, which focuses on multi-user collaboration in the virtual environment connected over the network. CoVE allows synchronized interaction with other users in a remote virtual system, in addition to providing rich virtual windowing library, which allows the integration of 2D-based control widgets in the virtual environment.

Equalizer[33] is a cross-platform toolkit designed to optimize parallel rendering that is scalable to various system configurations. Equalizer allows an application to adjust the configuration at runtime, adopting to multiple processors or graphics cards, or clustered

system without any modification to the core code.

Avango [6] is a scene graph based framework for developing virtual reality applications, where the node functionality is extended to provide additional data-flow and network communication, thus allowing the distribution of the application over the network, and component-based design with loose-coupling between objects.

CaveUT [47] is a modification of the Unreal Tournament 2004 game engine, adopted to run the game in the CAVE-like systems. Unreal Tournament 2004 is a network-based game engine, where the server maintains the master copy of the virtual world, and each user is considered as a client connected to the server, with a different view for each client. The view is based on the client head position, allowing realistic view shifting. The use of the game engine as the base of the virtual world allows the production and rendering of realistic models of avatars and objects in the scene, and real-time interaction among users in the virtual world.

Quest3D [1] is a graphical development platform to create real-time 3D application, allowing to create almost any 3D application in the visual interface without the need to code. Quest3D provides advanced graphical features such as lighting and shading, and interaction with various input devices and output devices, suitable for the CAVE application as well.

3D Visualizer [52] is a software package developed to visualize 3D gridded data sets using a volume and flow visualization tool. It uses Vrui (Virtual Reality User Interface) to handle the rendering and interaction in immersive display systems such as CAVE.

All the above products have their strengths in different areas, focusing on specific aspects of developing VR applications. Some of them share similar features, but form their foundations by using different mechanisms to be suitable for the specific development or running environment considered. CAVEMANDER, currently built on top of FreeVR [85], shares

the flexibility and scalability in CAVE system deployment with various input and rendering configurations. In our view, the major achievement of CAVEMANDER is that of providing a software platform on top of such functionality to support a software engineering method that facilitates the creation of scenario-based simulation applications for CAVE, where the users can develop such applications using primarily a graphical wizard, the CAVEMANDER Wizard, and immerse in the simulated environment displayed in CAVE. Figure 7.1 shows a summary of the features supported by the products and resources described above, where the features are selected in terms of their significance for CAVE-like systems and other VR applications.

Note that this table is intended for a higher level overview only, and we do not claim either that the set of features considered is exhaustive or that CAVEMANDER is better than other products under some of the criteria considered. We aim to show, however, that CAVEMANDER is different from them in some aspects, and hence it fills an existing gap in the current landscape of software construction methods and related resources for building CAVE-based C&C applications.

7.2 Work Related to C&C Applications

There are several projects and applications designed for C&C that provide specific interaction methods to access information and/or control objects on the map. For example, the Virtual Planning Room (ViPR) uses a large spherical screen to display complex information, allowing to focus on a set of detailed information on the foreground and others on the background [16]. There have been other approaches attempting to enhance the accessibility of the information by adding the 3rd dimension to the display. One such approach uses a display method integrating traditional 2D information in the 3D space, which increases the users ability to access the desired data quickly using spatial cues [24]. Also, by using 2D

Table 7.1: Comparison of Related Products

		Product									
		CAVELib	FreeVR	VR Juggler	CoVE	Equalizer	Avango	CaveUT	Quest3D	3D Visualizer	CAVEMANDER
Supported Features	Cross Platform*	√	√	√	√	√	√				√
	API	√	√	√	√	√				√	√
	Flexible Input Device	√	√	√			√		√	√	√
	Scalable Rendering	√	√	√		√	√	√	√	√	√
	Graphical Widgets				√		√		√	√	
	Multi-Process & Multi-Thread	√	√	√		√					√
	Shared Memory	√	√	√	√	√					√
	Network Communication	√	√	√	√	√	√	√	√		√
	SE Method										√
	GUI Wizard								√		√
	Game Engine							√	√		
	Scenario Simulation							√			√

* Cross platform feature is based on the availability and support on major operating systems.

and 3D in a hybrid display system for GIS applications, the users can see the 2D based map on top of the 3D world, directly associating the related maps together [15].

Most of the C&C applications attempt to display a large amount of (typically complex) information all at once using a single large display area or a 3D space. Another solution is to provide a means to switch views to see different data layers or levels of detail. However, most of the time these solutions still have some drawbacks and present some difficulties. Switching views can cause numerous brief disruptions in-between changing the views, in which users can experience difficulty in comparing two sets of data at the same time, or in maintaining their train of thoughts [39]. The use of multiple screens or large screen can reduce situation awareness as well. In addition, it is hard to understand slope and elevation in the virtual world by using traditional displays, and this approach does not provide an immersive experience to the commander either. On the other hand, one of a usability study surveyed shows that displaying information in CAVE significantly improves the accuracy and efficiency in the spatial understanding of complex 3D structures [82]. Another study indicates that immersive environments can significantly improve the training performance in performing complex procedures [89]. Thus, by providing an immersive environment that allows executing C&C operations, the proposed CAVEMANDER system enables users to understand the situation and access data in a more efficient manner and with higher data integrity.

Chapter 8

FUTURE WORK

While the result of our research and development work, CAVEMANDER (which includes our proposed software development approach and its set of related software resources) has taken a well defined shape and has grown in contents and scope, it represents nevertheless only a stage in a possibly larger process of advancement for CAVE-centered methodologies and related supporting resources. We are aware of several of the current limitations of CAVEMANDER and while we believe in its utility, value, and potential, we also know that more work is needed to strengthen it and increase its impact on the existing landscape of software methods and tools for immersive VR application development.

Thus, in this chapter several directions of possible work are outlined, some of them hopefully interesting and challenging enough to be undertaken by new graduate students at UNR willing to continue the work described in the present dissertation. These directions can be summarized as follows: new and more complex military and non-military simulations and real-world applications and scenarios; expanded, enhanced, and well-tested set of software resources; related usability studies; methodological improvements; and miscellaneous, longer-term extensions and enhancements. These are detailed in the sections that follow.

8.1 New Simulation and Real-World Applications

Earlier in this dissertation a sample simulation and scenario have been presented to illustrate the use of the proposed methodology and tool. However, even though it exemplifies the main capabilities of CAVEMANDER, the application described serves mostly only as a proof of concept. More evolved and complex simulation and real-world applications and related scenarios are needed to fully test the possibilities of CAVEMANDER, both in the military and non-military domains.

In the military C&C domain it would be useful to build a full-fledged simulation and training application that supports direct assessment of the trainees' performance using a precise grading system, for example based on points and levels, as encountered in computer games. In the non-military C&C domain, a very interesting and potentially very useful system built using CAVEMANDER could simulate various aspects of planet expedition planning and training. Yet other civilian applications could focus on the training and preparation for fighting wild fires or on planning and executing search and rescue missions (e.g., in flooded or other area affected by natural disasters).

In terms of applications, obviously the ultimate goal of CAVEMANDER would be to be used in real-world, real-time applications that involve employing the VR technology for solving practical problems. Steps towards using CAVEMANDER and CAVE in such applications could be explored in new graduate research work, including in work focused on placing the commander in CAVE as part of real-operation land- or sea-based C&C centers.

8.2 Additional Facilities and Software Components

As all software platforms, CAVEMANDER has inherently the potential for growing in terms of new and enhanced facilities, features, components, and other related resources (e.g., user manuals, tutorials, community projects, etc.). As indicated earlier in the thesis, we believe that short term plans for designing and developing the Playback, CommHub and ActPC software components of the CAVEMANDER platform are needed (for details on the roles of these components, please refer to Chapter 5 of the thesis). Furthermore, additional thorough testing of all planned CAVEMANDER resources are needed for increased reliability and improved user satisfaction.

Furthermore, making CAVEMANDER an open source project would most certainly be another initiative that would allow for the tool and the approach's growth through the software engineering community's collaboration and contribution.

8.3 Usability Studies

It is expected that highly useful feedback could be obtained from users, both developers and commanders, by performing usability studies and experiments with CAVEMANDER. In fact, we have considered and planned such studies, but the research and development work on the CAVEMANDER software itself has taken much more time and effort than estimated, leaving us little leeway to conduct these studies.

Nevertheless, the way we would perform such studies is outlined in Figure 8.1, which depicts the proposed usability experiment's general set-up, and Figure 8.2, which displays the high-level interaction involved in this experiment.

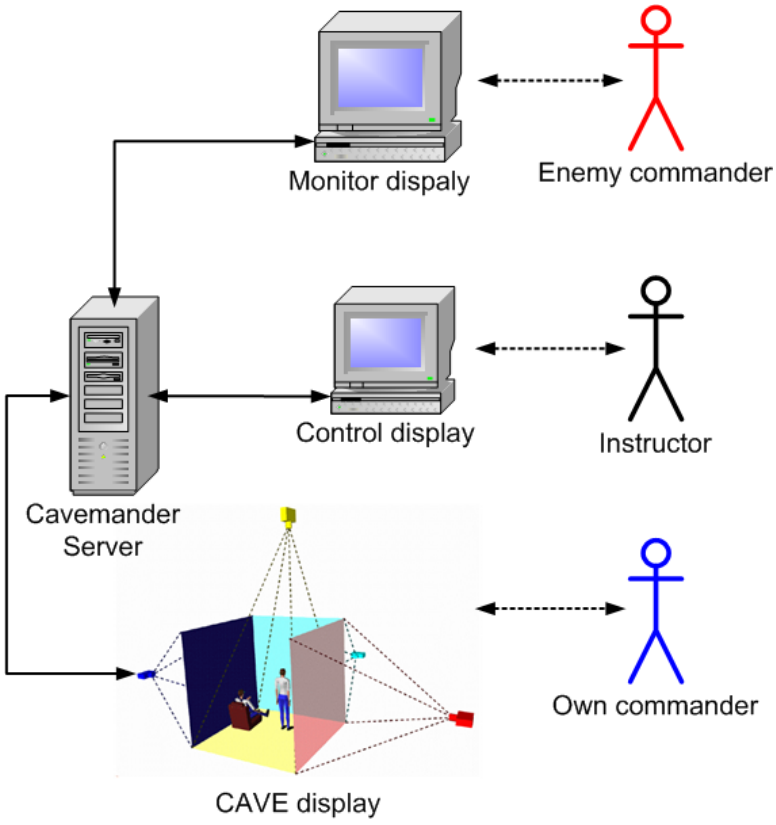


Figure 8.1: Proposed Usability Study General Diagram

In short, the instructor could design various scenario problems and two commanders (leading two opposite forces) could perform against each other. One commander would be using CAVE and CAVEMANDER to interact with the scenario problem proposed by the instructor, while the other commander would use a regular PC and 2D graphics for the same purpose. The instructor could evaluate the performance of the two instructors and all three kinds of actors (users) involved in the experiment could provide feedback and information on the usability aspects of the software tools used.

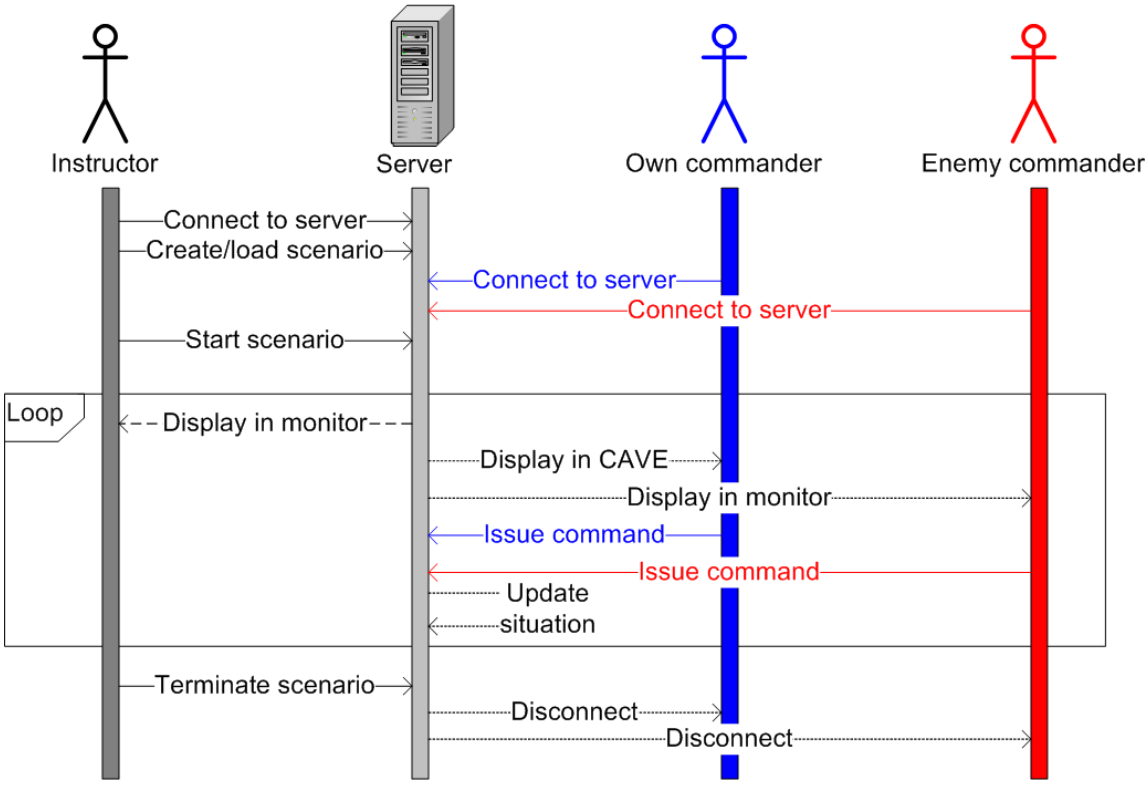


Figure 8.2: Proposed Usability Study Interaction Diagram

8.4 Methodological Improvements

Extensions in terms of methodological improvements are also possible in several directions. First of all, in what concerns the methodology, we have focused on the design and the implementation of new CAVE-based software applications using CAVEMANDER. However, the general software development process (software life cycle) contains besides these two key phases some additional phases and activities, including planning, requirements specification, analysis, deployment, and evolution (maintenance) [88] [76] [4]. All these phases can be addressed in more detail, with specific guidelines, specific tasks to be performed, and specific artifacts that can be produced in these tasks. On the long term, a full software process designed specifically for CAVE-based development would be not only an interesting but also a worthwhile project for someone with expertise and interest in software engineering.

Furthermore, the development process could integrate elements of Model-Driven Development (MDD) [21] [25] [31] such that some of the process' activities would consist of automatic generation of code from software models, e.g. UML models [4] [104] [30]. This, together with faster and less expensive solutions for semi-automated graphics development activities would be similar to the lines of work followed in [99], which describes the GENETICS open source project aimed at creating with less overhead enhanced natural environments and terrain for interactive CAVE simulations.

8.5 Miscellaneous Extensions

The possibilities for extending and enhancing CAVEMANDER are quite broad. This software platform and development approach could benefit from a number of additional capabilities and features including, but not limited to, speech recognition [36] [45] [87] [96] [64] and voice synthesis [20] within the CAVE, inclusion in the VR environment of advanced devices such as head-mounted devices, eye gaze trackers [106] [71] and command gloves [72], and integration of AI and game engine components as parts of an extended CAVEMANDER software development platform.

Chapter 9

CONCLUSIONS

In this final chapter of the thesis a brief review of our research and development work's main findings and experiences is presented, together with a summary of our thesis' contributions.

9.1 Learning from Our Work

The research and development work presented in this thesis has explored several interesting, challenging and promising areas and technologies of computer science and engineering. Placed at the intersection of immersive virtual reality, software engineering, and human-computer interaction, this thesis' topic has focused on providing a new software construction method and a set of related software resources for developing C&C applications in the CAVE.

Based on our prior experience and background, the motivation of our work came from several sources, including our desire to explore one of the most advanced computer technology

available today, the CAVE, and contribute to advancing the state-of-the art in developing C&C simulations for this technology. We have surveyed existing C&C as well as VR literature and noticed that there is a real need for software engineering methods and related supporting tools for building software for C&C systems that can run in CAVE. Thus, aiming to address this situation, we have worked on a software construction method and a set of related tools and resources, which made up the CAVEMANDER project described in this thesis. Details of the method's activities and artifacts, as well as of the related software architecture and its components were presented in the thesis. Among the software resources we developed, the CAVEMANDER Wizard is of particular utility, as its GUI interface and interaction design allow users that are not computer experts to create and run simulation scenarios. We have illustrated the proposed method and its associated software with an example of a C&C simulation scenario from the military domain. While the focus of our work was on military applications, CAVEMANDER can be used in the future for non-military applications, for example in training for search and rescue missions. Based on a survey of current VR software products and other resources as well as on an overview of existing C&C applications we compared CAVEMANDER with related work and placed our approach in the context of existing research and development projects aimed at supporting software construction for CAVE.

The work presented in the thesis has been challenging and while we believe it contributed to the above described research and development context, we are aware that it represents only the beginning of several potentially very rewarding directions of future work. Such directions can include newer and more complex simulations, newer and more advanced features and software components for CAVEMANDER, extensions to the proposed method, increased focus on HCI aspects with special emphasis on usability studies, and automated code generation.

From studying and working on the above components of our thesis we have learned about the complexity and significance of the computer science areas and technologies covered, in

particular about VR, CAVE, C&C, software engineering, and HCI. While we have proposed a new software construction method that emphasizes an organized approach for building software based on reusing existing resources and guiding step-by-step non expert users in the process, we also realized that even more planning and application of software engineering principles, methods, and tools would have been beneficial to our work. Also, as in many reported software projects, we realized that underestimation of time and effort needed to complete software development is one of the most prevalent, if not the most prevalent problem in software engineering, and we wished that we have been more aware of it from the beginning of our work. Finally, we have learned that applied research work requires a continuous process of searching, exploring, studying, investigating, and trying new solutions. Overall, we believe that the experience gained in working on the CAVEMANDER project is a very valuable and important one, which will certainly help us participate better prepared and contribute more effectively to future research and development projects.

9.2 Summary of Contributions

The work on the CAVEMANDER project has brought several contributions, that can be summarized as follows (starting with the ones we consider more important):

- Proposal of a software construction method consisting of several design and implementation activities and producing several software artifacts that allow the creation of C&C simulation scenarios that can run in CAVE. This method fills an existing gap in the available software techniques for developing C&C applications in immersive VR environments and provides the basis for a future full-fledged specialized software development process;
- Creation of a related set of a resources, organized as components of an expandable client-server architecture that emphasizes reuse and user-centered interaction with

the computer. This set of resources includes a GUI Wizard, a software tool that enables users of all backgrounds to easily create and modify simulation scenes as well as easily create, update and run simulation scenarios. It also includes implemented reusable and expandable software components (SimScene and ActCAVE) and related API methods in both the server and client sides of the architecture. The design of the CAVEMANDER architecture also includes provisions for several useful near future components, in particular a Playback component for re-running at different speeds already executed scenarios (for training and analysis purposes), and a CommHub component for connecting CAVEMANDER with information from actual operation fields placed outside the CAVE;

- An example of simulation application from the military domain, illustrating the scene definition, simulation build up, scenario creation, and scenario execution activities supported by CAVEMANDER, and making use of the project's available software resources;

- A survey of existing challenges in displaying information in C&C applications that led to the identification of five categories of possible solutions (out of which we have chosen the one we thought most promising, the immersive VR);

- An exploration of the VR realm, with emphasis on CAVE, that has focused on the technology's main characteristics, devices, and typical applications, and resulted in a proposed classification of VR systems as well as in a set of identified HCI challenges for CAVE-based simulations;

- An overview of related work, with a comparison of existing software products that helped place the CAVEMANDER project in the context of research and design efforts for VR software development;

- Identification of several interesting, challenging, and worthwhile directions of future work that can lead to new research and development projects in the Software Engineering Laboratory (SOELA) at UNR or elsewhere.

Bibliography

[1] Act-3D B.V. Quest 3D: visual 3D development software, 6 2009. Accessed at `http://www.quest3d.com`.

[2] Aghevli, A., Bachmann, A., Bresina, J., Greene, K., Kanefsky, B., Kurien, J., McCurdy, M., Morris, P., Pyrzak, G., Ratterman, C., et al. Planning applications for three Mars missions with Ensemble. In *International Workshop on Planning and Scheduling for Space* (2006).

[3] Aoki, P. M. Back stage on the front lines: perspectives and performance in the combat information center. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2007), ACM, pp. 717–726.

[4] Arlow, J., and Neustadt, I. *UML and the Unified Process: practical object-oriented analysis and design, 2nd.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2005.

[5] Athans, M. Command and Control (C2) theory: A challenge to control science. *IEEE Transactions on Automatic Control 32*, 4 (1987), 286–293.

[6] Avango. Avango, 6 2009. Accessed at `http://www.avango.org/`.

[7] Baudisch, P. Interacting with large displays. *Computer 39*, 4 (April 2006), 96–99.

[8] Baumgartner, N., Retschitzegger, W., and Schwinger, W. A software architecture for ontology-driven situation awareness. In *Proceedings of the 2008 ACM Symposium on Applied Computing* (Fortaleza, Ceara, Brazil, 2008), ACM, pp. 2326–2330.

[9] Bayyari, A., and Tudoreanu, M. E. The impact of immersive virtual reality displays on the understanding of data visualization. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (Limassol, Cyprus, 2006), ACM, pp. 368–371.

[10] Benyon, D., Turner, P., and Turner, S. *Designing interactive systems.* Addison-Wesley, 2005.

[11] Bidasaria, H. B. Development of techniques for visualization of scalar and vector fields in an immersive environment (CAVE). In *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2* (Kennesaw, Georgia, 2005), ACM, pp. 355–356.

[12] Bigley, G. A., and Roberts, K. H. The incident command system: High-reliability organizing for complex and volatile task environments. *The Academy of Management Journal 44*, 6 (2001), 1281–1299.

[13] Biocca, F., and Levy, M. R. *Communication in the age of virtual reality.* L. Erlbaum Associates Inc., 1995.

[14] Boslaugh, D. L. *When Computers Went to Sea: The Digitization of the United States Navy.* Wiley-IEEE Computer Society, Los Alamitos, CA, USA, 2003.

[15] Brooks, S., and Whalley, J. L. A 2D/3D hybrid geographical information system. In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (Dunedin, New Zealand, 2005), ACM, pp. 323–330.

[16] Broughton, M. Virtual planning rooms (ViPR): a 3D visualisation environment for hierarchical information. In *AUIC '06: Proceedings of the 7th Australasian User Interface Conference* (Darlinghurst, Australia, Australia, 2006), Australian Computer Society, Inc., pp. 125–128.

[17] Brown, P. J., and Jones, G. J. F. Context-aware retrieval: Exploring a new environment for information retrieval and information filtering. *Personal Ubiquitous Comput. 5*, 4 (2001), 253–263.

[18] Buntha, S. Command control communications computers and intelligence software for naval surface warfare. Master's thesis, Department of Computer Science and Engineering, University of Nevada, Reno, December 2004.

[19] Buntha, S., Dascalu, S., Debnath, N., Okamoto, S., and Buntha, P. Software environment for tactical training with trajectory planning through multiple moving targets. In *Proc. IEEE International Conference on Electro/Information Technology* (2007), pp. 163–168.

[20] Buza, O., Toderean, G., Domokos, J., and Bodo, A. Voice synthesis application based on syllable concatenation. In *Proc. IEEE International Conference on Automation, Quality and Testing, Robotics AQTR 2008* (22–25 May 2008), vol. 3, pp. 473–478.

[21] Calic, T., Dascalu, S., and Egbert, D. Tools for MDA software development: Evaluation criteria and set of desirable features. In *Proc. Fifth International Conference on Information Technology: New Generations ITNG 2008* (2008), pp. 44–50.

[22] Cavazza, M., Lugrin, J.-L., Pizzi, D., and Charles, F. Madame Bovary on the holodeck: immersive interactive storytelling. In *MULTIMEDIA '07: Proceedings of the 15th International Conference on Multimedia* (New York, NY, USA, 2007), ACM, pp. 651–660.

[23] Cliburn, D. C. A virtual reality laboratory for undergraduates. *J. Comput. Small Coll. 24*, 2 (2008), 57–63.

[24] Cockburn, A., and McKenzie, B. Evaluating the effectiveness of spatial memory in 2D and 3D physical and virtual environments. In *CHI '02: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2002), ACM, pp. 203–210.

[25] Cooper, K., Dai, L., Dascalu, S., Mehta, N., and Velagapudi, S. Towards aspect-oriented model-driven code generation in the formal design analysis framework. In *Postprintings of the International Workshop on Systems and Software Architectures (IWSSA-2007 at SERP-2007)* (Las Vegas, NV, June 2007).

[26] Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A. Surround-screen projection-based virtual reality: the design and implementation of the CAVE. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA, 1993), ACM, pp. 135–142.

[27] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., and Hart, J. C. The CAVE: Audio visual experience automatic virtual environment. *Commun. ACM 35*, 6 (1992), 64–72.

[28] Dascalu, S., and Buntha, S. Simulation software for naval surface warfare training. In *Proc. World Automation Congress WAC '06* (2006), pp. 1–6.

[29] Dascalu, S., Buntha, S., Saru, D., and Debnath, N. Software tool for naval surface warfare simulation and training. *J. Comp. Methods in Sci. and Eng. 6*, 5,6 Supplement 2 (2007), 427–444.

[30] Dascalu, S., Fritzinger, E., Debnath, N., and Akinwale, O. STORM: Software tool for the organization of requirements modeling. In *2006 IEEE International Conference on Electro/Information Technology* (2006), pp. 250–255.

[31] Duarte, J., Tonanez, M., Cernuzzi, L., and Lopez, O. IDEAS03: Evaluation of software development through an MDA tool: a case study. *IEEE (Revista IEEE America Latina) Latin America Transactions 6*, 3 (July 2008), 252–259.

[32] Dudfield, H., Macklin, C., Fearnley, R., Simpson, A., and Hall, P. Big is better? human factors issues of large screen displays with military command teams. In *Proc. People in Control Human Interfaces in Control Rooms, Cockpits and Command Centres The Second International Conference on (IEE Conf. Publ. No. 481)* (2001), pp. 304–309.

[33] Eilemann, S., Makhinya, M., and Stalder, C. Equalizer: Parallel rendering, 06 2009. Accessed at `www.equalizergraphics.com`.

[34] Emergency management Institute. ICS-100.a: Introduction to the incident command system, 7 2009. Accessed at `http://emilms.fema.gov/IS100A/index.htm`.

[35] Emery, L., Catchpole, K., Macklin, C., Dudfield, H., and Myers, E. Big is better? Empirical results of an assessment of command teams with large screen displays. In *Proc. People in Control Human Interfaces in Control Rooms, Cockpits and Command Centres The Second International Conference on (IEE Conf. Publ. No. 481)* (2001), pp. 86–91.

[36] Erzin, E. Improving throat microphone speech recognition by joint analysis of throat and acoustic microphone recordings. *IEEE Transactions on Audio, Speech, and Language Processing 17*, 7 (Sept. 2009), 1316–1324.

[37] Funke, G. J., and Galster, S. M. The effects of spatial processing load and collaboration technology on team performance in a simulated C2 environment. In *ECCE '07: Proceedings of the 14th European Conference on Cognitive Ergonomics* (New York, NY, USA, 2007), ACM, pp. 37–43.

[38] Goodman, P. H., Buntha, S., Zou, Q., and Dascalu, S.-M. Virtual neurorobotics (VNR) to accelerate development of plausible neuromorphic brain architectures. *Front Neurorobotics 1* (2007), 1–7.

[39] Grudin, J. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Seattle, Washington, United States, 2001), ACM, pp. 458–465.

[40] Hanani, U., Shapira, B., and Shoval, P. Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction 11*, 3 (2001), 203–259.

[41] Headquarters United States Marine Corps. MCDP 6: Command and control, 7 2009. Accessed at `http://www.au.af.mil/au/awc/awcgate/mcdp6/toc.htm`.

[42] Heim, S. *The Resonant Interface: HCI foundations for interaction design.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2007.

[43] Holling, C., and Meffe, G. Command and control and the pathology of natural resource management. *Conservation biology 10*, 2 (1996), 328–337.

[44] Huang, X., and Huang, Y. R. Using contextual information to improve retrieval performance. In *Proc. IEEE International Conference on Granular Computing* (2005), vol. 2, pp. 474–481 Vol. 2.

[45] Ion, V., and Haeb-Umbach, R. A novel uncertainty decoding rule with applications to transmission error robust speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing 16*, 5 (July 2008), 1047–1060.

[46] Jacobson, J. Using "CaveUT" to build immersive displays with the unreal tournament engine and a PC cluster. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics* (Monterey, California, 2003), ACM, pp. 221–222.

[47] Jacobson, J., Renard, M. L., Lugrin, J.-L., and Cavazza, M. The CaveUT system: Immersive entertainment based on a game engine. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (New York, NY, USA, 2005), ACM, pp. 184–187.

[48] Johnson, A., Roussos, M., Leigh, J., Vasilakis, C., Barnes, C., and Moher, T. The nice project: learning together in a virtual world. In *Proc. Virtual Reality Annual International Symposium IEEE 1998* (1998), pp. 176–183.

[49] Joint Electronic Library. DoD dictionary of military and associated terms, joint publication 1-02, as amended through 17 October 2008, 7 2009. Accessed at `http://www.dtic.mil/doctrine/jel/doddict/`.

[50] Kahler, R., Cox, D., Patterson, R., Levy, S., Hege, H.-C., and Abel, T. Rendering the first star in the universe - a case study. In *Proc. IEEE Visualization VIS 2002* (2002), pp. 537–540.

[51] Kreylos, O., Bawden, G., Bernardin, T., Billen, M. I., Cowgill, E. S., Gold, R. D., Hamann, B., Jadamec, M., Kellogg, L. H., Staadt, O. G., and Sumner, D. Y. Enabling scientific workflows in virtual reality. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and its Applications* (Hong Kong, China, 2006), ACM, pp. 155–162.

[52] Kreylos, O., and Billen, M. I. 3D-Visualizer: Interactive gridded data volume visualization software, 7 2009. Accessed at `http://keckcaves.org/software/VISUALIZERCG/index.html`.

[53] Kuka, D., Lindinger, C., Hörtner, H., Berger, F., and Zachhuber, D. Applying "second life" to a CAVE^{TM}-like system for the elaboration of interaction methods with programmable interfaces. In *ACM SIGGRAPH 2007 Posters* (San Diego, California, 2007), ACM, p. 157.

[54] Lewis, M., and Jacobson, J. Introduction. *Commun. ACM 45*, 1 (2002), 27–31.

[55] Lidal, E. M., Langeland, T., Giertsen, C., Grimsgaard, J., and Helland, R. A decade of increased oil recovery in virtual reality. *IEEE Comput. Graph. Appl. 27*, 6 (2007), 94–97.

[56] MacEachren, A. M., Edsall, R., Haug, D., Baxter, R., Otto, G., Masters, R., Fuhrmann, S., and Qian, L. Virtual environments for geographic visualization: potential and challenges. In *Proceedings of the 1999 Workshop on New Paradigms in Information Visualization and Manipulation in Conjunction with the Eighth ACM Internation Conference on Information and Knowledge Management* (Kansas City, Missouri, United States, 1999), ACM, pp. 35–40.

[57] Macedonia, M., and Zyda, M. A taxonomy for networked virtual environments. *IEEE Multimedia 4*, 1 (1997), 48–56.

[58] Mancero, G., Wong, W., and Amaldi, P. Looking but not seeing: implications for HCI. In *ECCE '07: Proceedings of the 14th European Conference on Cognitive Ergonomics* (New York, NY, USA, 2007), ACM, pp. 167–174.

[59] Mechdyne Corporation. CAVE software, 6 2009. Accessed at `http://www.mechdyne.com/integratedSolutions/software/products/CAVELib/CAVELib.htm`.

[60] Mine, M. Virtual environment interaction techniques. *UNC Chapel Hill Computer Science Technical Report TR95-018* (1995), 507248–2.

[61] Nacenta, M. A., Sakurai, S., Yamaguchi, T., Miki, Y., Itoh, Y., Kitamura, Y., Subramanian, S., and Gutwin, C. E-conic: a perspective-aware interface for multi-display environments. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA, 2007), ACM, pp. 279–288.

[62] Nakatsu, R., and Tosa, N. Active immersion: the goal of communications with interactive agents. In *Proc. Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies* (2000), vol. 1, pp. 85–89 vol.1.

[63] NASA. International space station, 07 2009. Accessed at `http://www.nasa.gov/mission_pages/station/multimedia/ISS_FCR.html`.

[64] Nordstrom, K., Rutledge, G., and Driessen, P. Using voice conversion as a paradigm for analyzing breathy singing voices. In *Proc. PACRIM Communications, Computers and Signal Processing 2005 IEEE Pacific Rim Conference on* (24–26 Aug. 2005), pp. 428–431.

[65] Norris, J., Powell, M., Fox, J., Rabe, K., and Shu, I.-H. Science operations interfaces for Mars surface exploration. In *Proc. IEEE International Conference on Systems, Man and Cybernetics* (2005), vol. 2, pp. 1365–1371.

[66] Norris, J., Powell, M., Vona, M., Backes, P., and Wick, J. Mars exploration rover operations with the science activity planner. In *Proc. IEEE International Conference on Robotics and Automation ICRA 2005* (2005), pp. 4618–4623.

[67] Office of the Chief of Naval Operations. Maritime operations in the information age, 7 2009. Accessed at `http://www.navy.mil/navydata/policy/vision/vis00/v00-ch1b.html`.

[68] Office of the Executive Secretary (ExecSec) of the Department of Defense (DoD). Command, control, communications, computers, and intelligence, 7 2009. Accessed at `http://www.dod.mil/execsec/adr95/c4i_5.html`.

[69] Olanda, R., Pérez, M., Morillo, P., Fernández, M., and Casas, S. Entertainment virtual reality system for simulation of spaceflights over the surface of the planet Mars. In *VRST '06: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2006), ACM, pp. 123–132.

[70] Open Tech, inc. CoVE - an open source virtual reality toolkit, 6 2009. Accessed at `http://cove.opentechinc.com`.

[71] Ott, R., Gutiérrez, M., Thalmann, D., and Vexo, F. Advanced virtual reality technologies for surveillance and security applications. In *VRCIA '06: Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and its Applications* (New York, NY, USA, 2006), ACM, pp. 163–170.

[72] Parian, K., Hegie, J., Kimmel, A., Dascalu, S. M., and Harris, F. C. WiELD-CAVE: Wireless ergonomic lightweight device for use in the CAVE. In *18th International Conference on Software Engineering and Data Engineering (SEDE-2009)* (2009).

[73] PCMAG.CO. The independent guide to technology, 7 2009. Accessed at `http://www.pcmag.com/encyclopedia/`.

[74] Prabhat, Forsberg, A., Katzourin, M., Wharton, K., and Slater, M. A comparative study of desktop, fishtank, and cave systems for the exploration of volume rendered confocal data sets. *IEEE Transactions on Visualization and Computer Graphics 14*, 3 (2008), 551–563.

[75] Preddy, S. M., and Nance, R. E. Key requirements for CAVE simulations. In *WSC '02: Proceedings of the 34th Conference on Winter Simulation* (2002), Winter Simulation Conference, pp. 127–135.

[76] Pressman, R., and Ince, D. *Software engineering: a practitioner's approach*. McGraw-Hill New York, 2005.

[77] Richey, S. K. Operational command and control of federal domestic emergency response operations. Tech. rep., Department of Joint Military Operations, Naval War College, 2000.

[78] Roussou, M. Learning by doing and learning through play: an exploration of interactivity in virtual environments for children. *Comput. Entertain. 2*, 1 (2004), 10–10.

[79] Rozzi, S., Amaldi, P., Wong, W., and Field, B. Operational potential for 3D displays in air traffic control. In *ECCE '07: Proceedings of the 14th European conference on Cognitive ergonomics* (New York, NY, USA, 2007), ACM, pp. 179–183.

[80] Ryan, M. Immersion vs. interactivity: Virtual reality and literary theory. *SubStance 28*, 2 (1999), 110–137.

[81] Salmon, P. M., Walker, G. H., Ladva, D., Stanton, N. A., Jenkins, D. P., and Rafferty, L. Measuring situation awareness in command and control: comparison of methods study. In *ECCE '07: Proceedings of the 14th European Conference on Cognitive Ergonomics* (New York, NY, USA, 2007), ACM, pp. 27–34.

[82] Schuchardt, P., and Bowman, D. A. The benefits of immersion for spatial understanding of complex underground CAVE systems. In *VRST '07: Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2007), ACM, pp. 121–124.

[83] Sharp, H., Rogers, Y., and Preece, J. *Interaction design: Beyond human computer interaction*. John Wiley & Sons, 2007.

[84] Shendarkar, A., Vasudevan, K., Lee, S., and Son, Y.-J. Crowd simulation for emergency response using BDI agent based on virtual reality. In *Proceedings of the 38th Conference on Winter Simulation* (Monterey, California, 2006), Winter Simulation Conference, pp. 545–553.

[85] Sherman, W. R. FreeVR homepage. Website, March 2009. Accessed at `http://www.freevr.org/`.

[86] Sherman, W. R., and Craig, A. B. *Understanding Virtual Reality: Interface, Application, and Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[87] Smaragdis, P., and Shashanka, M. A framework for secure speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing 15*, 4 (May 2007), 1404–1413.

[88] Sommerville, I. *Software Engineering, 8th.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2006.

[89] Sowndararajan, A., Wang, R., and Bowman, D. A. Quantifying the benefits of immersion for procedural training. In *IPT/EDT '08: Proceedings of the 2008 Workshop on Immersive Projection Technologies/Emerging Display Technologiges* (New York, NY, USA, 2008), ACM, pp. 1–4.

[90] Stern, C., Noser, H., Weissmann, J., and Stucki, P. Application scenarios for scientific visualization and virtual reality using a cave infrastructure. In *Proceedings of the workshop on Virtual environments 2003* (Zurich, Switzerland, 2003), ACM, pp. 319–320.

[91] Sun Microsystems, Inc. Java technology and the mission to Mars, 7 2009. Accessed at `http://www.sun.com/aboutsun/media/features/mars.html`.

[92] Sutcliffe, A., Gault, B., Fernando, T., and Tan, K. Investigating interaction in CAVE virtual environments. *ACM Trans. Comput.-Hum. Interact. 13*, 2 (2006), 235–267.

[93] the Office of the Assistant Secretary of Defense (NII). The command and control research program (CCRP), 07 2009. accesssed at `http://www.dodccrp.org/`.

[94] Turoff, M. Past and future emergency response information systems. *Commun. ACM 45*, 4 (2002), 29–32.

[95] VR Juggler. The vr juggler suite, 07 2009. Accessed at `http://www.vrjuggler.org`.

[96] Wan, C.-Y., and Lee, L.-S. Histogram-based quantization for robust and/or distributed speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing 16*, 4 (May 2008), 859–873.

[97] Wang, Y., Otitoju, K., Liu, T., Kim, S., and Bowman, D. A. Evaluating the effects of real world distraction on user performance in virtual environments. In *VRST '06: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2006), ACM, pp. 19–26.

[98] Warnstadt, L. C. S. Improving command and control of civil support operations. Master's thesis, Joint Forces Staff College, Advanced Joint Professional Military Education, 2008.

[99] Wells, W. D. Generating enhanced natural environments and terrain for interactive combat simulations (GENETICS). In *VRST '05: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2005), ACM, pp. 184–191.

[100] Wick, J., Callas, J., Norris, J., Powell, M., and Vona, M. Distributed operations for the Mars exploration rover mission with the science activity planner. In *Proc. IEEE Aerospace Conference* (2005), pp. 4162–4173.

[101] Wikipedia: The Free Encyclopedia. File:ICS Process.PNG, 7 2009. Accessed at `http://en.wikipedia.org/wiki/File:ICS_Process.PNG`.

[102] Wikipedia: The Free Encyclopedia. File:ICS Structure.PNG, 7 2009. Accessed at `http://en.wikipedia.org/wiki/File:ICS_Structure.PNG`.

[103] Wikipedia: The Free Encyclopedia. File:ISS flight control room 2006.jpg, 7 2009. Accessed at `http://commons.wikimedia.org/wiki/File:ISS_Flight_Control_Room_2006.jpg`.

[104] Zhang, B., and sho Chen, Y. Enhancing UML conceptual modeling through the use of virtual reality. In *Proc. 38th Annual Hawaii International Conference on System Sciences HICSS '05* (03–06 Jan. 2005), pp. 11b–11b.

[105] Zhang, S., Demiralp, c., Keefe, D. F., DaSilva, M., Laidlaw, D. H., Greenberg, B. D., Basser, P. J., Pierpaoli, C., Chiocca, E. A., and Deisboeck, T. S. An immersive virtual environment for DT-MRI volume visualization applications: a case study. In *Proceedings of the Conference on Visualization '01* (San Diego, California, 2001), IEEE Computer Society, pp. 437–440.

[106] Zhu, Z., and Ji, Q. Novel eye gaze tracking techniques under natural head movement. *IEEE Transactions on Biomedical Engineering 54*, 12 (Dec. 2007), 2246–2260.