A GUI WIZARD FOR DEVELOPING COMMAND & CONTROL APPLICATIONS IN CAVE

Sermsak Buntha Muhanna Muhanna Sohei Okamoto Sergiu Dascalu Frederick C. Harris

Department of Computer Science and Engineering
University of Nevada, Reno
{buntha, muhanna, okamoto, dascalus, fredh}@cse.unr.edu

ABSTRACT
Command and control systems have gained an increasingly vital role when it comes to displaying information about operational situations. In such situations, pictures have traditionally been presented on 2D media, rather than 3D displays. CAVEMANDER is a software platform that we created to improve the visualization and immersion of presenting command and control situations inside the CAVE. This paper describes a proposed GUI wizard used to develop command and control applications in CAVE based on the software architecture of CAVEMANDER. The wizard supports designing and creating command and control simulation scenes and scenarios. Details of the wizard's interface and functionality as well as a case study of a simulation scenario are provided in the paper.

KEY WORDS
CAVE, command & control, GUI wizard, human-computer interaction, simulation, software engineering, virtual reality.

1. Introduction

Traditional paper- and glass-based approaches are still being used in some command and control (C&C) applications in different countries around the world. Unfortunately, these traditional displays are usually inconvenient and may lead to late results, which decrease their usability for commanders. More enhanced 2D displays are being largely used to present pictures of the operational C&C situations, such as computer monitors and wall-mounted screens. Such displays have several limitations as well. Displaying too detailed, excessive information pertaining to a C&C situation, for example, can decrease the awareness and performance of the commanders [1]. As pointed out by Aoki, the complexity and multiplicity of interactive regions in Combat Information Centers can create problematic situations for the decision makers [2]. Also, reducing the amount of information displayed to the commander can result in the loss of the essential ones. 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 [3].

To find answers for such a dilemma, several approaches have been investigated and suggested by researchers. Information filtering, for example, is one of the methods to handle large information flows but several issues still exist [4]. A good context model can possibly estimate a set of relevant pieces of information [5] [6]. However, the estimated result cannot be relied on by a critical system, such as a C&C system. Another study suggested a method using a perspective-aware interface that can be used to enhance multi-display environments [7]. However, while users move their focuses from one screen to another, their train of thought might be lost [8]. Furthermore, it has been shown that using large display screens can provide better situation awareness [9]. Some studies have pointed out that large screen displays improve the situation awareness and the assessment of the situation during C&C activities [10] [11]. The benefits of 3D images have been indicated by numerous research studies as well. In [12], for example, authors showed that immersive environments can significantly improve the trainees' performance in complex procedures. 3D visualizations for air traffic control have shown an improvement in the awareness of the relative position between aircraft-aircraft or aircraft-airspaces landmarks [13].

CAVEMANDER was originally proposed in the [3] dissertation to overcome the challenge of finding a solution for displaying the right C&C data at the right time, while keeping the commander aware of all the essential parts of a situation. CAVEMANDER is a software platform that can be used to create C&C applications to be run in CAVE. It includes several reusable server and client software components, including a graphical user interface called ServGUI Wizard.

ServGUI Wizard, described in this paper, has the main role to enable the creation of simulation scene and scenario descriptors in XML format that can be used by other CAVEMANDER software components. It 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.

The paper, in its remaining part, is organized as follows: Section 2 describes the main components of the CAVEMANDER's architecture, Section 3 presents an overview of the CAVEMANDER approach, Section 4 explains in detail the ServGUI Wizard component of the CAVEMANDER, Section 5 provides an example of a simulation scenario created using the ServGUI Wizard and executed in CAVE, and Section 6 points to several directions of future work and concludes the paper.

2. CAVEMANDER Architecture

The CAVEMANDER software platform is based on the client-server software architecture. 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. The main subsystems of the CAVEMANDER architecture are the Server and the Client, as shown in Figure 1.

The Server has several components, including the CommHub, the Playback, the LOG database, the SimScene, and the ServGUI Wizard. ServGUI Wizard is a core component of the CAVEMANDER architecture, and is described in detail in Section 4. The other core component of the Server is the SimScene, which consists of a set of reusable software resources, including C&C unit descriptor classes, templates, and API command functions. To allow the instructors and the trainees to review and analyze C&C scenarios performed in CAVE, the Playback and the Log components are included in the Server core component. The CommHub is a software component that allows connecting the CAVE with real-world C&C applications via telephone lines, wireless connections, and other communication media.

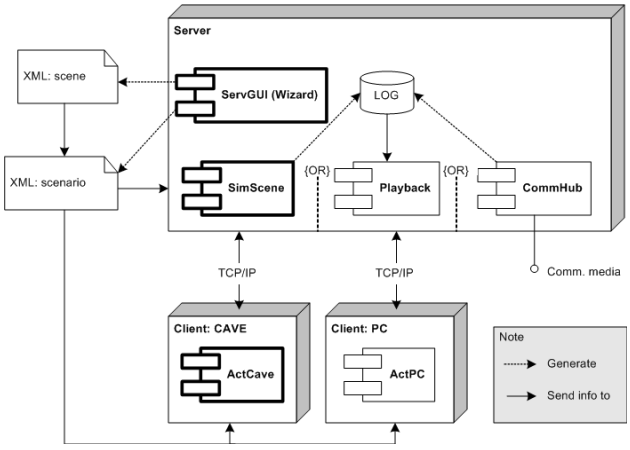

Figure 1: CAVEMANDER Architecture

The Client, on the other hand, has two main software components, ActCAVE and ActPC. These components are used to run simulation scenarios on CAVE and, respectively, on regular personal computers.

3. CAVEMANDER Approach

The software engineering approach of constructing CAVE-based C&C simulations consists of several activities, as illustrated in Figure 2. As shown in this figure, three main activities have to take place in a specific sequence in order to execute a scenario. The first activity is to define a scene. Here, the user is given the ability to either create a new scene or modify an existing one. The ServGUI Wizard is used in this activity to create a scene XML file based on the different property values of the scene entered by the user. The process of creating a new scene includes defining a scene concept (e.g., is this scene for a land military C&C or is it for a naval surface operation?), creating the scene's unit types along with their specifications, and defining the scene's environment factors. The second activity is to build a simulation, which results in creating a code included in the SimScene. This code can be either built from scratch or reused from a previously written code of an existing simulation available in the CAVEMANDER platform.

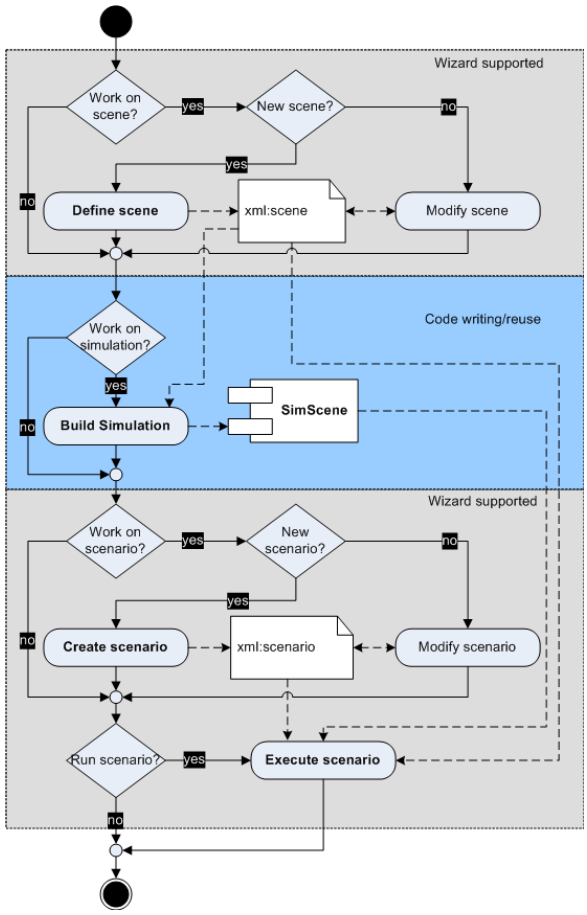

Figure 2: Overview of the CAVEMANDER Approach

The third activity is to create a scenario. This activity must be performed in order to create the simulation's first associated scenario. This activity, however, becomes optional if the user decides to use an already existing suitable scenario. The process of this activity starts by defining the configurations of the scenario (concept description, log file name, communication port number, name of the scene with which the scenario is associated, and the terrain on which the scenario will run). Once the configuration of the scenario is defined, the scenario's units, their ID values, and their initial locations need to be specified. Initializing the scenario's environment factors is the last step of this activity. This activity is also supported by the ServGUI Wizard to create a scenario XML file, an artefact that contains a detailed description of the scenario. After performing the three described activities as needed, the user is able to begin a simulation. This means that the resources created for the simulation are loaded on the server as well as on the client (the CAVE). During the simulation, the user has the possibility to slow down, speed up, pause, and stop the simulation.

4. The Server GUI Wizard

Figure 3 presents the use case diagram [14] of the CAVEMANDER ServGUI Wizard component. The Instructor (or, in UML terminology, the actor [15]) 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.

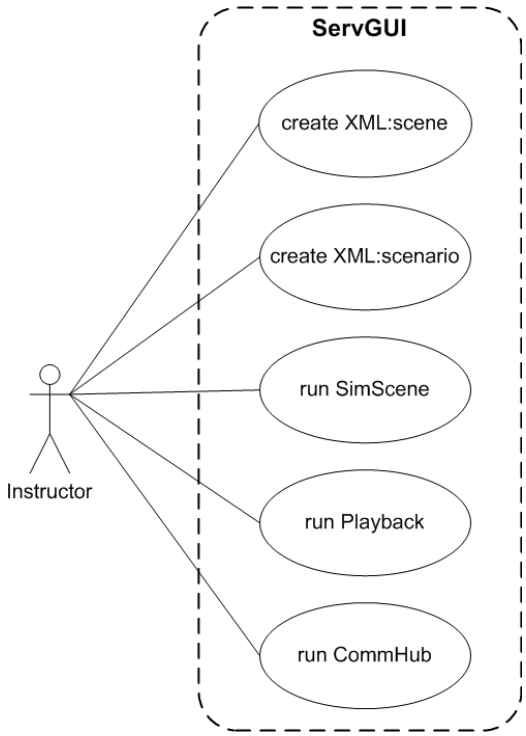

Figure 3: Use Case Diagram of the CAVEMANDER Wizard

Figure 4 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 for supporting the task related with the clicked button.

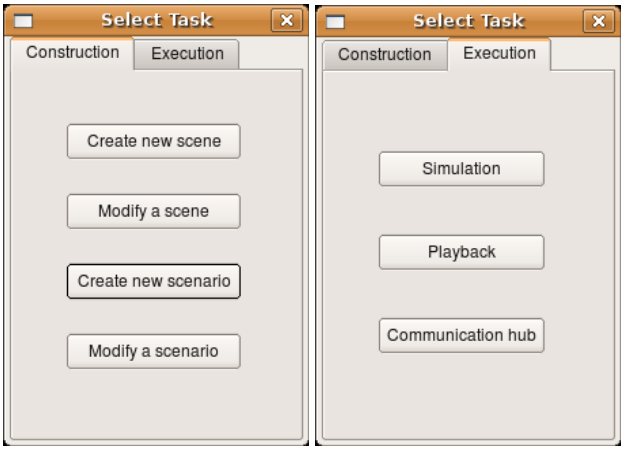

Figure 4: Widgets for Task Selection

4.1 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. 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 6.

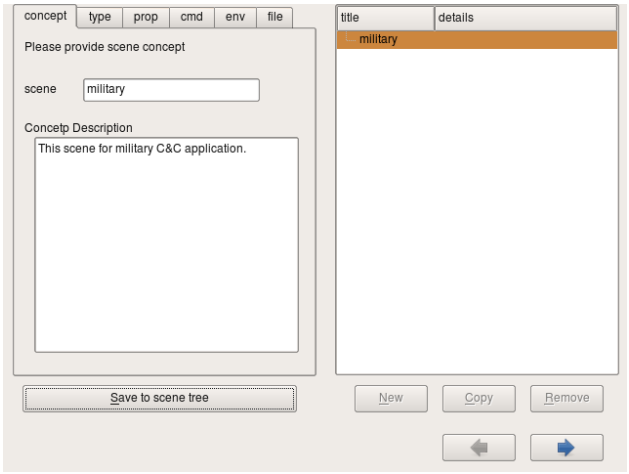

Figure 5: The Scene Concept Tab

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 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.

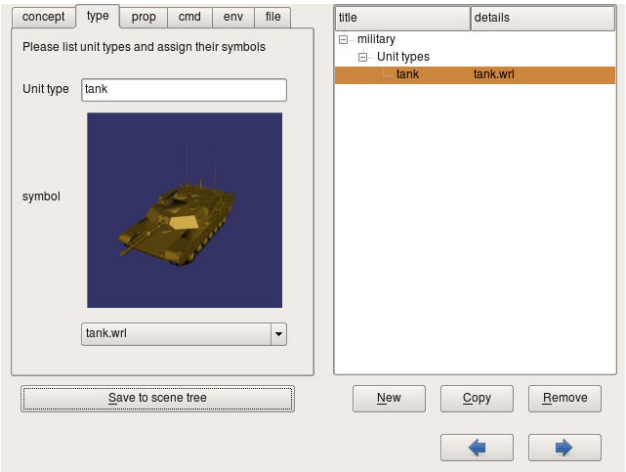

Figure 6: The Scene Type Tab

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

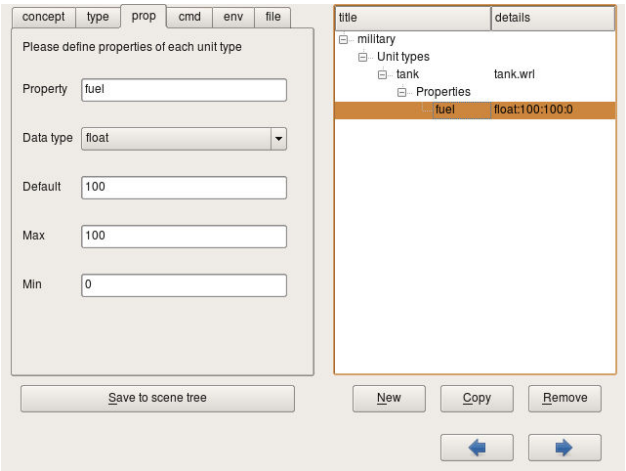

Figure 7: The Scene Properties Tab

The next step for the ServGUI 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 8. 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* (e.g. *stop*), *cmd+positions* (e.g. *moveTo* 40, 60), or *cmd+target* (e.g. *attack* tank02). 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 9. 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.

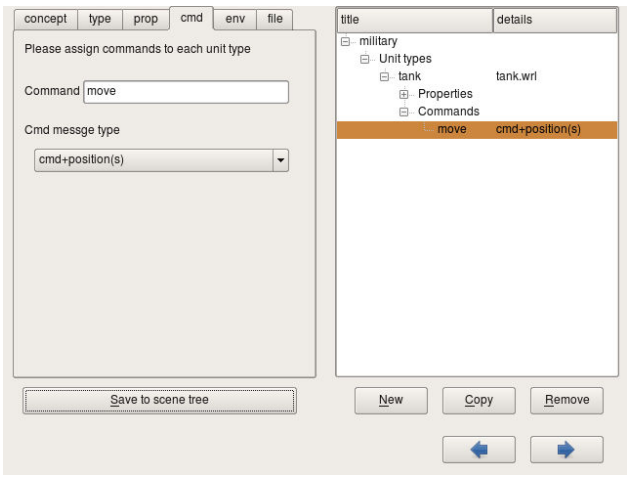

Figure 8: The Scene Commands Tab

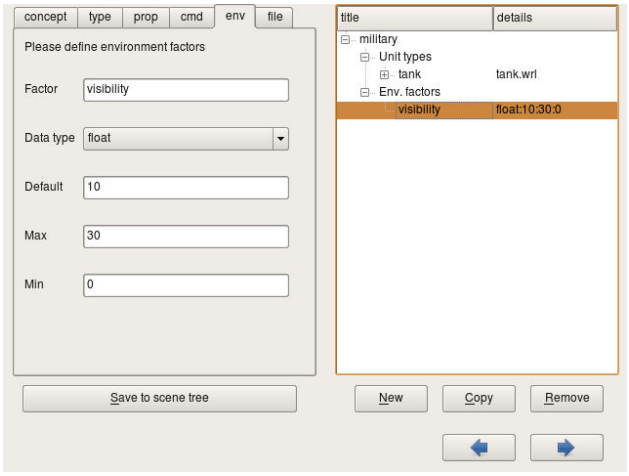

Figure 9: The Scene Environment Factors Tab

Finally, the user can save the scene tree to a scene file that is generated as an XML scene file, as shown in Figure 10. The extension for such files is "*scn*".

4.2 Wizard Functions to Define 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 11.

The next step in creating a scenario is to provide the configuration data of the scenario. Figure 12 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.

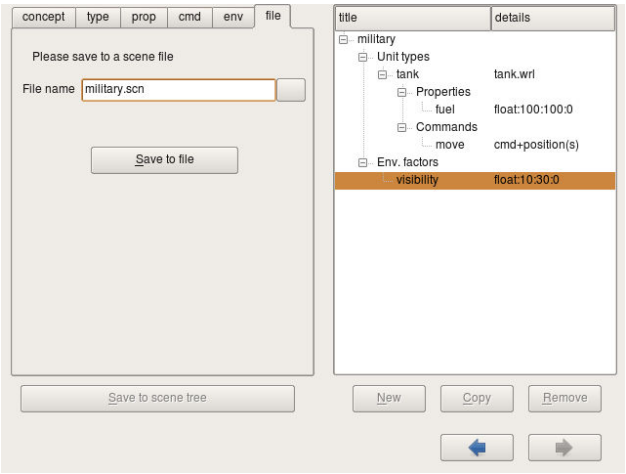

Figure 10: The Scene File Tab

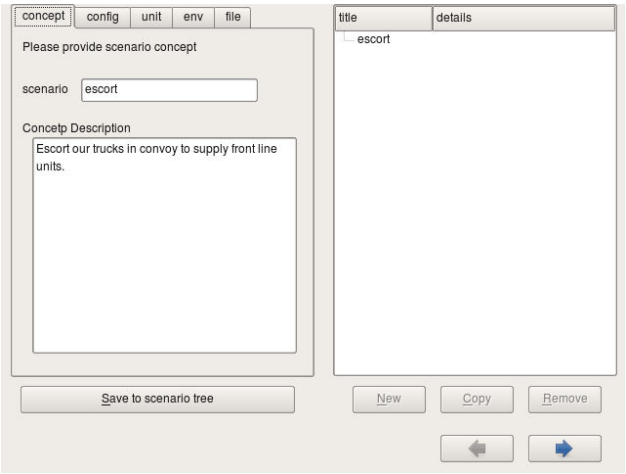

Figure 11: The Scenario Concept Tab

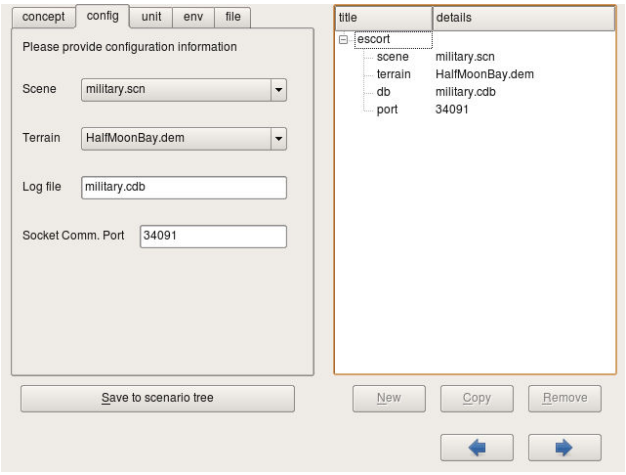

Figure 12: The Scenario Configuration Tab

Figure 13 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.

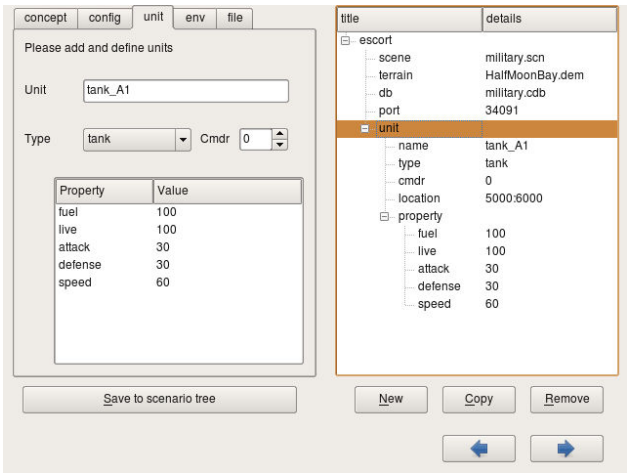

Figure 13: The Scenario Units Tab

To customize each scenario, the user can also change the initial value of each environment factor if needed, as shown in Figure 14.

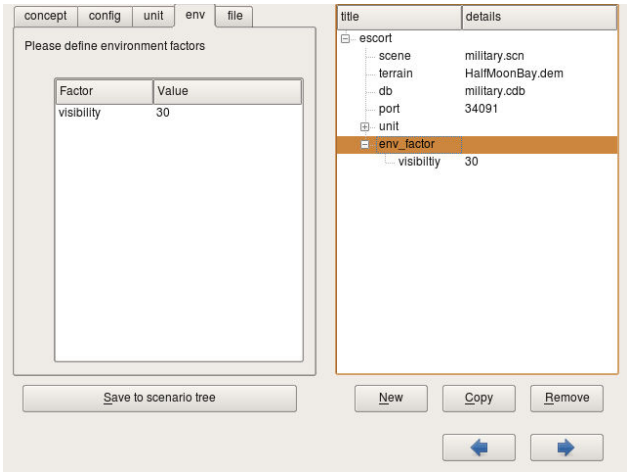

Figure 14: The Scenario Environment Factors Tab

As with the scene file, the user finishes creating a new scenario by saving the scenario tree, to a scenario file that is generated as an XML file, as shown in Figure 15. The extension of such files is “sco”.

4.3 Running the Scenario

When a scenario is created and ready to be run, it can be loaded from the Wizard's scenario window, as shown in Figure 16. Once a scenario is loaded, the data representing that scenario is 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.

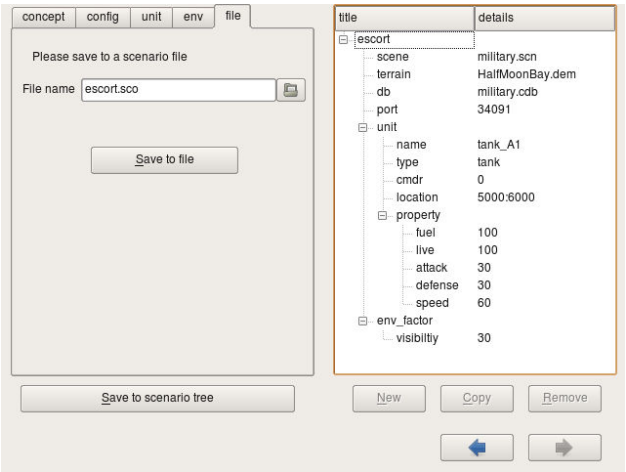

Figure 15: The Scenario File Tab

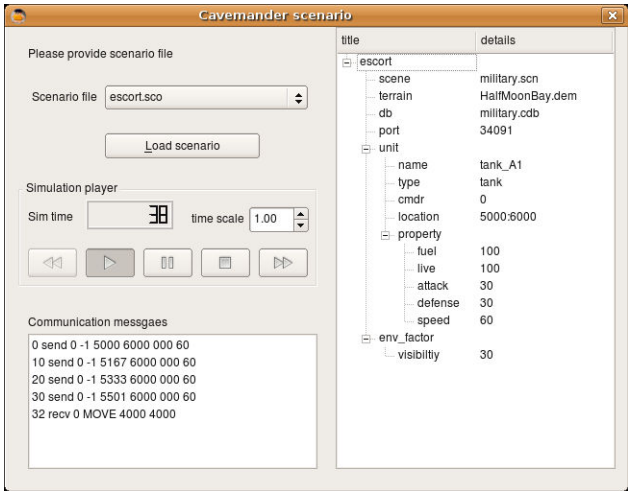

Figure 16: CAVEMANDER GUI for Running Scenarios

5. Simulation Scenario: A Case Study

To show an example of creating a simulation scenario using the ServGUI Wizard of CAVEMANDER, we developed a military C&C scenario, which can be used for training in military operations inside the CAVE system. The purpose of this scenario is to escort and distribute supplies to frontline units. The scenario includes 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 defence power but it is the only vehicle that can carry supply.

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

For simplicity, only few of the many possible details involved in scene definition are presented. Figure 17 shows a sample ServGUI Wizard snapshot from the scene definition.

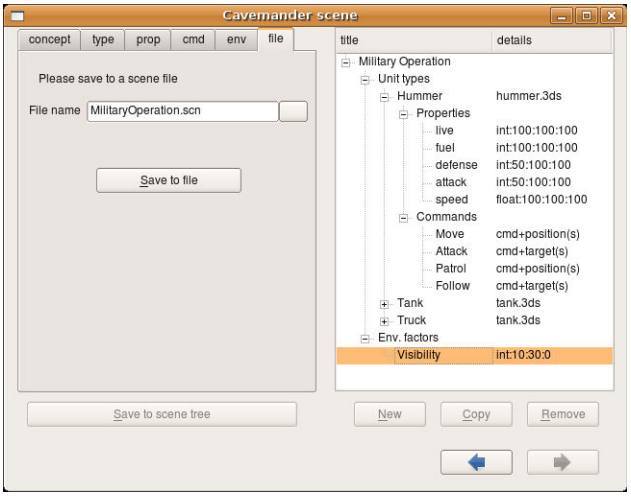

Figure 17: Sample GUI Snapshot from Scene Definition

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 18 provides a sample snapshot from the many possible detailed activities during the scenario creation using the ServGUI Wizard. Once the scene is defined and the scenario is created, the simulation is executed. Samples of snapshots of selected states are shown in Figures 19, 20, and 21. Figure 19 shows a view of the initial state when the simulation begins inside the CAVE system. All the tanks, hummers, and trucks are at their base and initial positions.

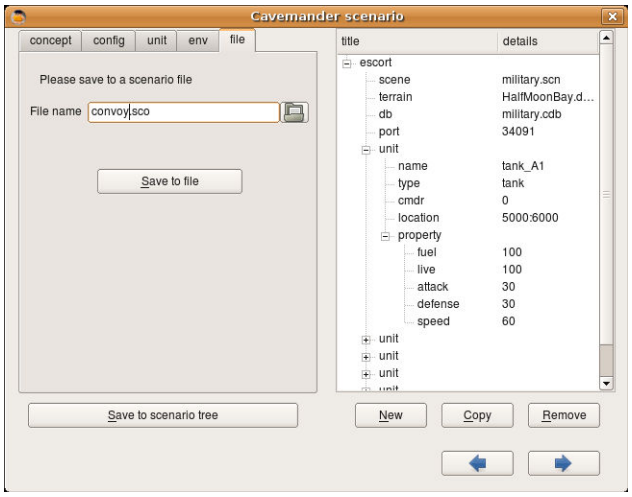

Figure 18: Sample GUI Snapshot from Scenario Creation

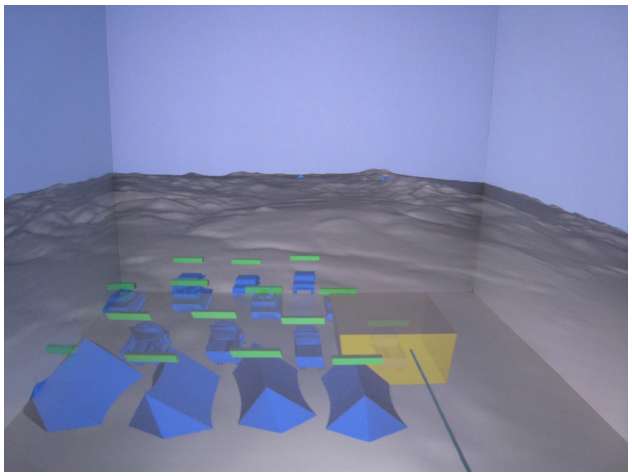

Figure 19: Sample Scenario Execution State (Initial State)

Figure 20 displays the view when the scenario is in the state of directing each convoy to follow the safe route to the designated platoons, where two convoys are formed with tanks and trucks, and start moving along the safe route surveyed by the hummers ahead.

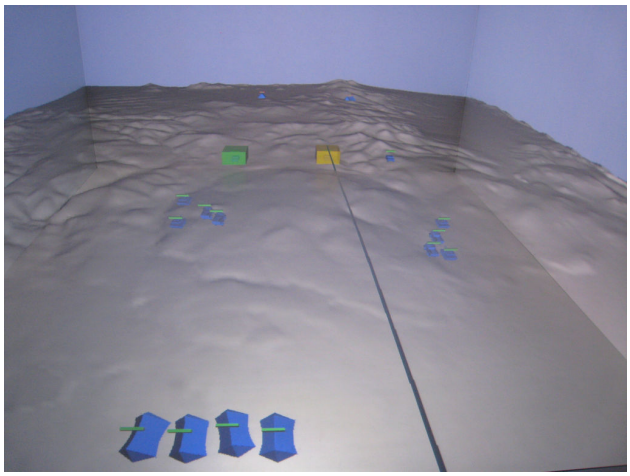

Figure 20: Sample Scenario Execution State (Moving Convoys)

Figure 21 shows another sample of a state of executing the scenario inside the CAVE system. Here, the Hummer encounters an enemy tank.

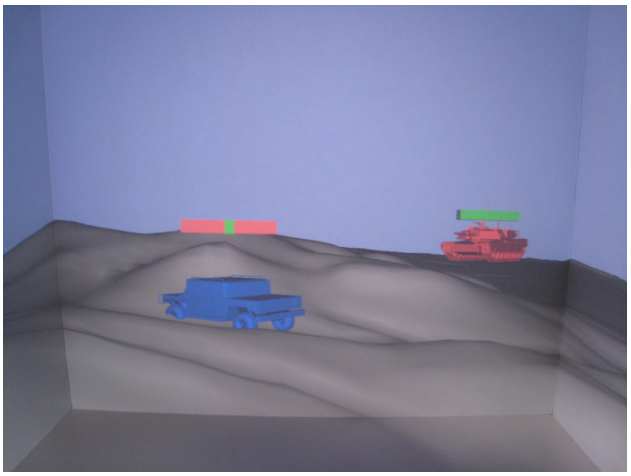

Figure 21: Sample Scenario Execution State (Hummer Encounters an Enemy Tank)

6. Future Work and Conclusion

Although CAVEMANDER and, in particular, the ServGUI Wizard component, has shown good and interesting results for us and for other potential researchers and developers, it represents only a stage in a possible larger process of advancement for CAVE-centered methodologies and related supporting resources [3]. Several aspects of the ServGUI Wizard can be further enhanced and developed, including the following.

An example of such enhancement is to allow for displaying a snapshot of the scene or scenario being created using the ServGUI Wizard. This might give users the ability to get an immediate feedback about the scene or scenario they are creating. Furthermore, conducting a usability study [16] [17] that identifies the different behaviour and interaction styles of using the wizard, would have a great impact on enhancing it to better understand the user's needs. More C&C applications built using the ServGUI Wizard will also help us enrich the tool's functionality.

ServGUI Wizard has been presented in this paper as one of the main software resources of CAVEMANDER. Overall, this GUI tool supports a better approach of designing and developing applications for CAVE systems, in particular C&C simulation scenarios. The wizard presented in this paper is part of a recent innovative and original approach for building C&C applications in CAVE. Its utility and value come especially from the fact that it supports end user development of C&C applications. Such users may not necessarily be computer experts, and may work in various

domains of activity, including the military, search and rescue missions, or various medical fields.

Acknowledgements

This work was made possible through the support 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.

References

[1] 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.

[2] 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.

[3] Buntha, S. CAVEMANDER: an approach and software platform for building command and control applications in CAVE, PhD Dissertation, *University of Nevada, Reno*, 2009.

[4] 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.

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

[6] Huang, X., and Huang, Y. R. Using contextual information to improve retrieval performance. *Proceedings of the IEEE International Conference on Granular Computing,* Vol. 2, 2005, pp. 474-481.

[7] 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. *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology* (Newport, Rhode Island, USA, 2007), ACM, pp. 279-288.

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

[9] 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.

[10] 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. *Proceedings of the Second International Conference on People in Control Human Interfaces in Control Rooms, Cockpits and Command Centres* (IEEE Conf. Publ. No. 481) (2001), pp. 304-309.

[11] 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. *Proceedings of the Second International Conference on People in Control Human Interfaces in Control Rooms, Cockpits and Command Centres* (IEEE Conf. Publ. No. 481) (2001), pp. 86-91.

[12] 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 Technologies* (New York, NY, USA, 2008), ACM, pp. 1-4.

[13] 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.

[14] Sommerville, I., *Software engineering* 8th edition, Addison-Wesley, Boston, MA, USA, 2006.

[15] Arlow, J., and Neustadt, I., *UML and the unified process: practical object-oriented analysis and design* 2nd edition, Addison-Wesley, Boston, MA, USA, 2005.

[16] Shneiderman, B., Plaisant, C., Cohen, M., and Jacobs, S., *Designing the user interface: strategies for effective human-computer interaction* 5th edition, Addison-Wesley, Boston, MA, USA, 2009.

[17] Hollingsed, T. and Novick, D., G., Usability inspection methods after fifteen years of research and practice. *Proceedings of the 25th annual ACM international conference on design of communication* (El Paso, Texas, USA, 2007), ACM, pp. 249-255.