CAVEMANDER: Creating 3-D Command-and-Control Scenarios for the CAVE Automatic Virtual Environment

Muhanna[1] Muhanna Sermsak Buntha[2] Sohei Okamoto[1] Michael J.[1] McMahon, Jr. Sergiu Dascalu[1] Frederick C.[1] Harris, Jr.

[1] Department of Computer Science and Engineering
University of Nevada, Reno, USA
{muhanna, okamoto, mcmahon, dascalus, fredh}@cse.unr.edu

[2] Department of Weapons and Tactics
Royal Thai Navy Academy, Thailand
serm34@gmail.com

Abstract — Command-and-control training scenarios represent interactive exercises in efficient and timely decision making. Most often, maximizing the realism and impact of these scenarios requires the commanders be presented with as much relevant information as possible. While 2-D media are quite capable of presenting scenario information, 3-D media are able to present *and* relate that information in greater quantity and more meaningfully. Towards this end, we have created CAVEMANDER – a framework, API, and set of tools for developing immersive 3-D command-and-control scenarios in the Cave Automatic Virtual Environment (CAVE). This paper presents the main features of CAVEMANDER, focusing on the application of a software engineering process to develop artifacts (i.e., simulation scenarios) under our design-build-execute approach. This development sequence is illustrated using software engineering diagrams, a graphical user interface wizard developed as a part of CAVEMANDER to simplify use, and actual generated artifact file contents.

Keywords — CAVE; CAVEMANDER; Command-and-Control; GUI Wizard; Software Engineering; Visual Design; VR.

I. INTRODUCTION

Command-and-control – or C&C – is an interactive exercise in which commanders must interpret all information for a given scenario and, based upon that information, reach reasonable conclusions in an efficient, effective, and timely manner [1]. Computers are increasingly taking on a central role in these scenarios, replacing large tactical glass boards or vertical paper charts to more effectively represent and convey the situation to the commander [2]. 2-D media (e.g. wall projectors and/or monitors), however, remain the predominant conveyances of scenario information [3] [4]. These computer-assisted representations of information still suffer from some inherent limitations, including: difficulties understanding, assessing, and relating geometrical parameters, delays in interpreting various data visualizations, and reduced awareness of the C&C situations [5] [3]. To improve C&C scenarios and address these issues, the Cave Automatic Virtual Environment (CAVE) was selected to represent the scenario information [1] – the advantages of using a third dimension in data visualization [6], combined with the rewards introduced by the immersive element of virtual environments [7], make the CAVE an excellent choice for representing C&C scenario information.

Several software tools and libraries are currently available to create virtual reality applications. CAVELib [8], for example, is a cross-platform application programming interface (API), used to develop virtual reality applications for various systems, including the CAVE. Originally developed at the University of Illinois in Chicago, CAVELib is now commercially owned by Mechdyne Corporation.

FreeVR [9] is an API that is used, primarily, to develop CAVE-based applications. This system provides an interface to utilize multiple devices available in virtual reality environments. VR Juggler [10] is a virtual reality application API that supports the execution of a virtual reality application on any display device using various input devices without the need to change or recompile the source code. CoVE [11] is a cross-platform toolkit built on Open Scene Graph that supports multi-user collaboration and multi-tasking efforts.

Quest3D [12] is a real-time 3-D engine and development platform that is intended to be used for developing software, 3-D web pages, and virtual reality simulators through visual interfaces. Equalizer [13] is a cross-platform toolkit designed to optimize parallel rendering while remaining sufficiently scalable to execute on various system configurations. Equalizer allows an application to dynamically adjust configuration at runtime, adapting to multiple processors or graphics cards, or utilizing computing cluster resources without any modification to the core code. Avango [14] is a scene graph based framework for developing virtual reality applications, where the node functionality is extended to provide additional dataflow and network communication.

CaveUT [15] is a modification of the Unreal Tournament 2004 game engine, adapted to run on CAVE-like systems. Unreal Tournament 2004 is a network-based game engine wherein the server maintains the master copy of the virtual world and each user is considered a client connected to the server, each with a different view. Each user's view is based on the position of their head, allowing realistic view shifting. Utilization of the game engine as the basis of the virtual world allows the production and rendering of realistic models of avatars and objects in the scene, as well as real-time interaction amongst users in the virtual world.

The above systems, as well as others [16], naturally have both common and distinguishing features; each focuses on developing virtual reality applications, yet each adds its own unique set of features and functionality. CAVEMANDER is similar to these systems in that it provides tools and functionality to create virtual environment applications. However, compared with other approaches (mostly focused on offering comprehensive API support), CAVEMANDER also applies a software engineering methodology and provides a set of tools to create a structured, easily extensible development system. Furthermore, besides a specialized API built on top of FreeVR [9], CAVEMANDER's set of available resources include a graphical user interface wizard, which simplifies the creation of C&C scenarios that enable immersion in the virtual environment of the CAVE.

The remainder of the paper is organized as follows: Section 2 presents the motivations behind CAVEMANDER, including a brief overview of its main components. Section 3 describes, in detail, the CAVEMANDER software development methodology. Section 4 demonstrates the system using the included graphical wizard interface to quickly create a C&C scenario. Section 5 indicates several potential directions for future work, concluding with a summary of developments in the CAVEMANDER system.

II. CAVEMANDER

CAVEMANDER has been developed to achieve several goals. Firstly, CAVEMANDER is targeted toward advancing the state-of-art in the development of C&C simulations by utilizing the CAVE – one of the most modern environments for experiencing and integrating new software engineering and human-computer interaction solutions. CAVE and associated technologies have been, and continue to be, in a state of rapid research and development [17]. Second, CAVEMANDER is intended to fill the gap between software development methods and software designed specifically to build CAVE-based applications. Third, CAVEMANDER seeks to contribute to the limited pool of CAVE-supporting resources, adding tools and reusable code to advance CAVE software development efforts worldwide. Lastly, CAVEMANDER provides a structured methodology and toolset-based foundation for future research and development of C&C simulation scenarios using the CAVE.

The CAVEMANDER approach consists primarily of a systematic software engineering method for a specialized software platform that includes a set of software resources [1]. The following sections focus on CAVEMANDER's software architecture, scenario development methodology, software artifacts, and GUI-based wizard.

III. METHOD ACTIVITIES AND ARTIFACTS

An overview of the activities involved in using CAVEMANDER is presented in Figure 1 in the form of a UML activity diagram [18], indicating both activities and the artifacts generated by those activities. The activity flow illustrates the steps required to generate a scenario using CAVEMANDER; the artifacts generated and passed amongst each step show the flow of those objects [19]. The four primary activities are: 1) *scene definition*, 2) *simulation build up*, 3) *scenario creation*, and 4) *scenario execution*.

A *scene* is a collection of mobile and/or fixed unit types (e.g. tanks, supply bases, vehicles) and environmental factors (e.g. visibility and weather) related to a C&C scenario. A scene is defined by specifying the units and/or environmental factors involved, as well as any parameters or commands related to them. A *simulation* adds the software implementation of items involved in a *scene*. Specifically, it is a set of classes associated with the unit types included in a *scene*, such as the ClassTank or the ClassSupplyBase. Finally, a *scenario* is a customized instance of a simulation. Here, the concrete types provided by the *simulation* are combined with specific values that are applied to the environment and units. In other words, instances of the unit types, along with specific parameters and environment variables, are defined to create the scenario. For example, a specific scenario might include three tanks and two supply units, each having specific parameters, all working in an environment in which the visibility is no more than five miles.

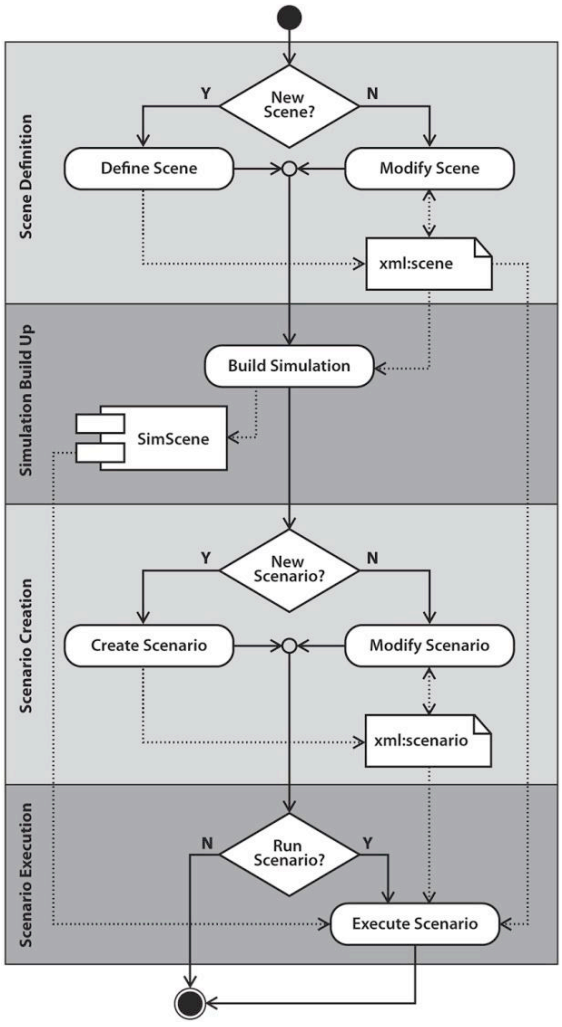

Figure 1. Overview of CAVEMANDER main activities and artifacts

In terms of the software development process, Scene Definition is a high-level design activity; Simulation Build-Up, however, is an implementation / coding, activity. In this step, existing/reusable code is integrated with new, customized code, to produce an executable software package: *SimScene*. Scenario Creation and Execution, however, are considered both design artifacts (in terms of describing the C&C situation) and implementation artifacts (as it executes in the CAVE).

A. Scene Defintion

Scene Definition is the first step in creating an executable scenario application. Users must either utilize an existing scene (optionally modifying it) or create a new scene. The scene definition activity is composed of three sub-activities: Scene Concept Definition, Unit Specification, and Environment Specification. Scene Concept Definition involves providing a text description of the scene (e.g. "A land C&C simulation" or "Naval operations").

The remaining two activities can be carried out in-parallel. Unit Specification entails defining the available models, properties, and commands for each unit type. Similarly, Environment Specification embodies the declaration of environmental factors and their properties. The result of this activity is the creation of an XML scene file, which contains the description of all elements that make up the scene (i.e. unit types and environmental factors).

B. Simulation Build-Up

During Simulation Build-Up, executable code is generated for the scenario - this code is contained in the package *SimScene*. As with the Scene Definition step, users must either use an existing simulation code package available from the CAVEMANDER system (optionally modifying it) or create a new one. Both cases involve code generation, requiring knowledge of a programming language.

Simulation Build-Up is divided into six sub-activities. First, an XML scene file – generated in the previous step – must be selected. Second, the classes needed to implement the unit types in the scene are either written or retrieved from the exiting CAVEMANDER library. Third, the classes needed to implement the environmental factors of the scene are either written or retrieved from the existing CAVEMANDER library. Fourth, the programmer includes all these implementation classes in the main simulation package: *SimScene*. Fifth, the programmer provides an update function for each time-step of the simulation. Finally, the code created in this activity is compiled and linked to the CAVEMANDER library, resulting in an executable file consisting of all elements that make up the scene.

C. Scenario Creation

Scenario Creation is the specification of the initial states and property values of the units and environmental properties of a specific scenario. As with the prior steps, users must either use an existing scenario (optionally modifying it) or create a new one. First, the concept of the scenario is specified as a text string (e.g. "Transporting supplies to front line units"). Second, the individual units in the scenario (not simply their types), their values (IDs), and their initial locations are specified. Third, the initial environment property values are specified. Fourth, the scenario is loaded on the client (CAVE) for later execution. The result of this step is an XML file containing the necessary property settings for all elements that make up the scenario.

D. Scenario Execution

In this activity, the resources required or generated for the simulation are loaded on the server and the client (CAVE) for execution. The user has the option to slow down, speed up, pause, resume, and stop the simulation. Support for a playback feature that allows reversing the progression of a scenario is included in the design of CAVEMANDER, however, the implementation of this feature is part of our future work plans.

IV. User Interface Wizard

ServGUI – the CAVEMANDER Wizard – is intended for novice users [17]. Using this wizard, ServGUI enables graphical scene definition, scenario creation, and scenario execution. In other words, ServGUI provides a visual tool that gives the user the ability to create software artifacts without the need to write code.

The user can interact with the ServGUI wizard by creating a scene or a scenario and running a simulation scene, playback, or communication hub. Figure 2 shows the *type* tab of the ServGUI wizard during the creation of a military scene. Here, the user is a adding a Hummer vehicle to the scene. They will then be able to define several properties and commands for the Hummer using the *prop* and *cmd* tabs, respectively. Figure 3 shows the user adding an attack command to the hummer using the *cmd* tab. Using the *env* tab, the user is able to provide several environmental factors for the scene.

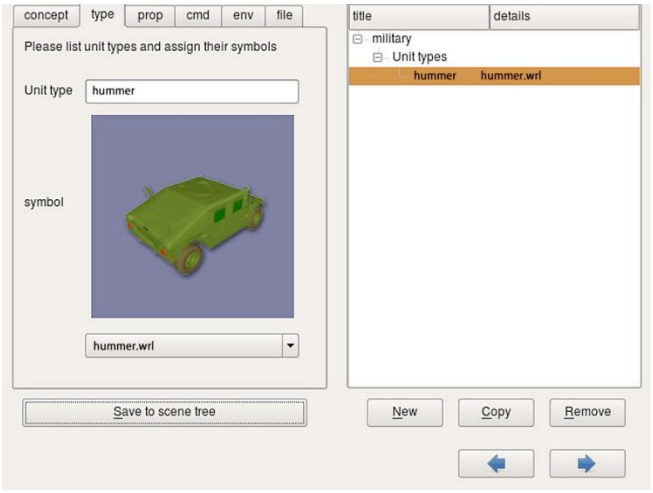

Figure 2. The *type* tab of the ServGUI wizard during the Scene Defnition activity

Once the scene is defined, it is saved as an XML file for use in simulation building, as shown in Figure 4.

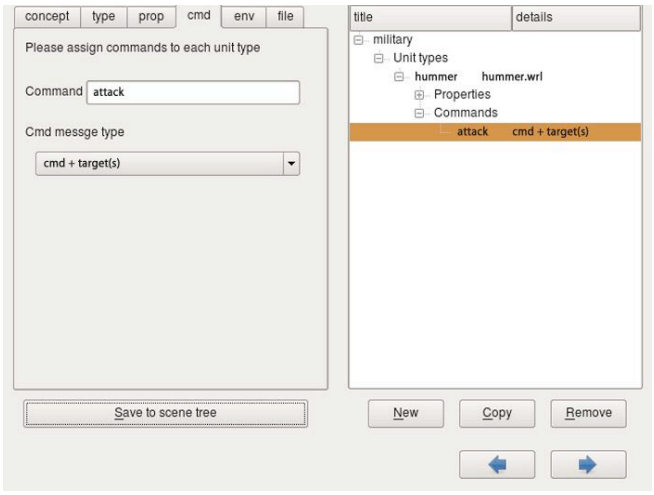

Figure 3. The *cmd* tab of the ServGUI wizard during the Scene Definition activity

Figure 4. Example of a scene artifact (XML file)from the Scene Definition activity

In the simulation build up software activity, the user assigns specific properties to each element of the scene. Figure 5 depicts the *unit* tab of the ServGUI wizard while the user is applying a different value for each of the Hummer properties. The specified values will be added to the simulation tree, which can be easily visualized by the user in the right-hand side of the user interface tab. Using this activity and the supported GUI, the user is also able to provide different values for environmental factors (e.g. visibility, temperature, wind speed, etc.).

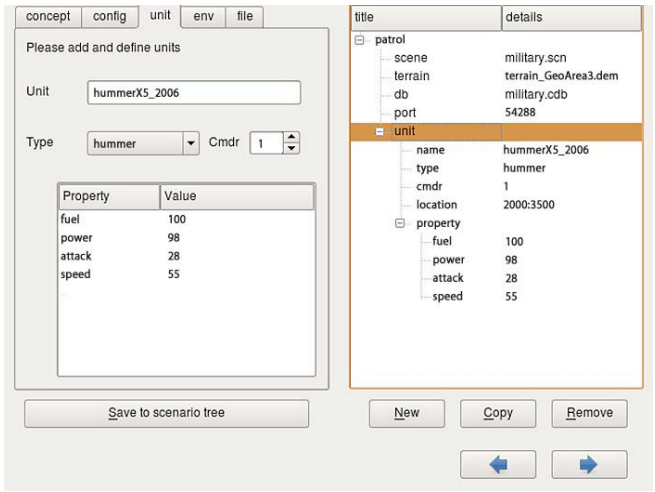

Figure 5. The *unit* tab of the ServGUI wizard during the Simulation Build-Up activity

In addition, the ServGUI wizard provides a tool that allows the user to visualize and interact with a simulation scenario during execution. This can be seen in Figure 6 where the user loads a scenario, executes it, and observes the values of the simulation elements and commands.

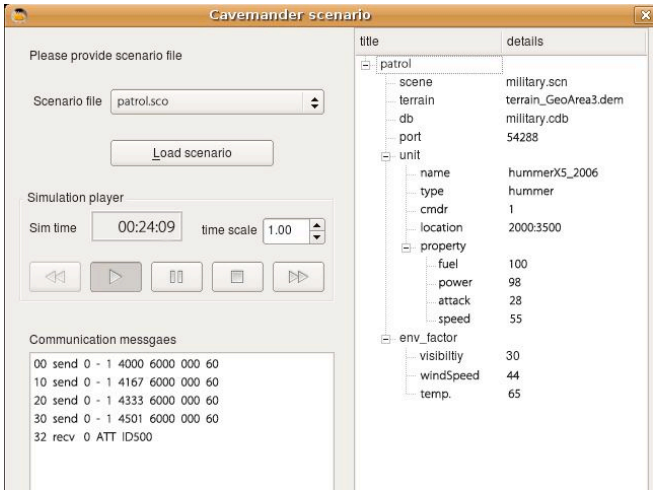

Figure 6. CAVEMANDER GUI for running scenarios

Figure 7 and Figure 8 show an example of a running simulation scenario built using CAVEMANDER. Figure 7 shows the scenario running in the FreeVR CAVE simulator [9], whereas Figure 8 shows the running scenario inside a 4-wall CAVE at the Desert Research Institute in Reno, Nevada.

V. Future Work and Conclusion

CAVEMANDER has shown promising results during development, with significant potential for use by researchers, application developers, and other CAVE users [1]. It introduces a state-of-the-art software engineering approach to build C&C applications in the CAVE through

the use of a visual GUI wizard that can effectively produce several software artifacts in the construction phase of such applications.

We have identified several potential directions for future work that could enhance the current CAVEMANDER system and related tools. First, new and more complex military and non-military simulations / application scenarios need to be developed. Second, several additional, enhanced, and well-tested sets of software resources should be provided. Third, we plan to conduct usability studies with focus on creating immersive C&C training scenarios. Finally, we intend to further investigate other potential process and architectural improvements. More details about the CAVEMANDER approach, related tools and applications, including a complete C&C scenario, can be found in [1].

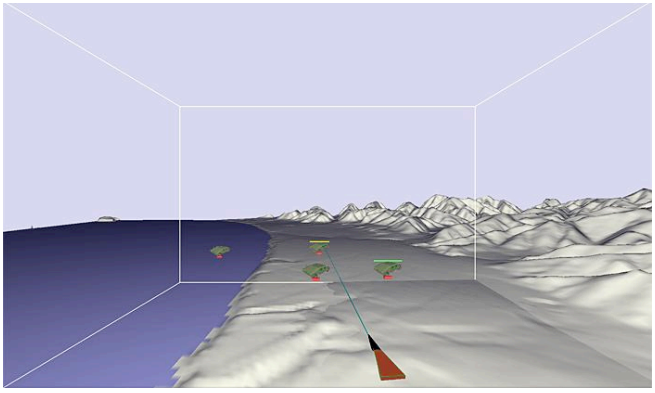

Figure 7. Sample Scenario Execution State in FreeVR CAVE simulator

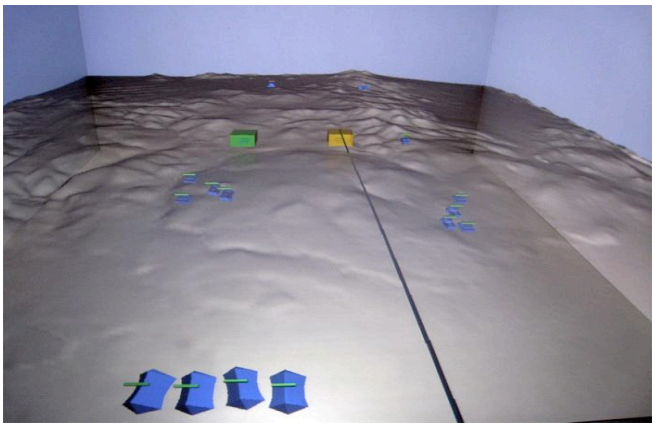

Figure 8. Sample Scenario Execution State in a 4-Wall CAVE

The focus of this paper has been on presenting the process and tools provided by CAVEMANDER to design and develop applications for the CAVE. These advances are significant, showing (amongst other points) that a novice user can visually design and build software artifacts that take an essential role in the construction phase of a CAVE-based system. In this way, CAVEMANDER takes a notable step toward providing non-expert computer programmers the ability to build simulation scenarios that run in the CAVE, thus allowing them to conduct VR immersive training sessions and research studies with reduced effort.

ACKNOWLEDGMENT

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] Buntha, S. CAVEMANDER: An Approach and Software Platform for Building Command-and-control Applications in CAVE, PhD Dissertation, University of Nevada, Reno, 2009.

[2] 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 Procs of the 20th Annual ACM Symp. on User Interface Software and Technology, Newport, RI, USA, 2007, ACM, pp. 279-288.

[3] Funke, G. J., and Galster, S. M., The effects of Spatial Processing Load and Collaboration Rechnology on Team Performance in a Simulated C2 Environment. In ECCE '07: Proceedings of the 14th ACM European Conference on Cognitive Ergonomics New York, NY, USA, 2007, ACM, pp. 37-43.

[4] 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 Proceedings of The Second IEEE International Conference on People in Control Human Interfaces in Control Rooms, Cockpits and Command Centres, 2001, pp. 304-309.

[5] Mancero, G., Wong, W., and Amaldi, P., Looking but Not Seeing: Implications for HCI. In ECCE '07: Procs. of the 14th European Conf. on Cognitive Ergonomics, NY, USA, 2007, ACM, pp. 167-174.

[6] Sowndararajan, A., Wang, R., and Bowman, D. A., Quantifying the Benefits of Immersion for Procedural Training. In IPT/EDT '08: Procs. of the 2008 Workshop on Immersive Projection Technologies/ Emerging Display Technologiges, NY, USA, 2008, ACM, pp. 1-4.

[7] Schuchardt, P., and Bowman, D. A. The benefits of Immersion for Spatial Understanding of Complex Underground CAVE Systems. In VRST '07: Procs. of the 2007 ACM Symposium on Virtual Reality Software & Technology, New York, USA, 2007, ACM, pp. 121-124.

[8] Mechdyne Corporation. CAVE Software, April 2010. Accessed at http://www.mechdyne.com/integratedSolutions/software/products/CAVELib/CAVELib.htm.

[9] Sherman, W. R. FreeVR Homepage, March 2010. Accessed at http://www.freevr.org/.

[10] VR Juggler. The VR Juggler Suite, April 2010. Accessed at www.vrjuggler.org.

[11] Open Tech, Inc. CoVE - An Open Source Virtual Reality Toolkit, February 2010. Accessed at http://cove.opentechinc.com.

[12] Act-3D B.V. Quest 3D: Visual 3D Development Software, April 2010. Accessed at http://www.quest3d.com.

[13] Eilemann, S., Makhinya, M., and Stalder, C. Equalizer: Parallel Rendering, March 2010. Accessed at www.equalizergraphics.com.

[14] Avango. Avango, April 2010. Accessed at http://www.avango.org/.

[15] Jacobson, J., Renard, M. L., Lugrin, J.-L., and Cavazza, M., The CaveUT System: Immersive Entertainment Based on a Game Engine. In ACE '05: Procs. of the 2005 ACM SIGCHI Intl. Conf. on Advances in Comp. Entertainment Technology, 2005, pp. 184-187.

[16] Kreylos, O., and Billen, M. I. 3D-Visualizer: Interactive Gridded Data Volume Visualization Software, March 2010. Accessed at http://keckcaves.org/software/VISUALIZERCG/index.html.

[17] Buntha S., Muhanna, M., Dascalu, S., Harris, F. C., and Okamoto, S., A GUI Wizard for Developing Command-and-Control Applications in CAVE, Procs. of the 4th IASTED Intl. Conf. on Human-Computer Interaction, St. Thomas, US Virgin Islands, USA, 2009, pp. 301-308.

[18] Arlow, J., and Neustadt, I., UML and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd Edition, Addison-Wesley, Boston, MA,USA, 2005.

[19] Sommerville, I., Software Engineering, 8th Edition, Addison-Wesley, Boston, MA, USA, 2006.