A Frankenstein Approach to Open Source: The Construction of a 3D Game Engine as Meaningful Educational Process

Brett E. Shelton, Jon Scoresby, Tim Stowell, Michael R. Capell, Marco A. Alvarez, and K. Chad Coats

Abstract—Using open source components to assemble a working 3D game engine is an attractive alternative to purchasing off-the-shelf technology. A student development team can use many different resources to investigate what underlying mechanisms are needed to build virtual environments. However, the techniques and processes involved when using open source components offer unique insights and educational opportunities. Leveraging and modifying existing software, and participating in the open source community, may alter the perspective of how game engines can be created. In this work, the process of building a simulation 3D game engine to support a training application for emergency response personnel is discussed. Evidence is presented that researching, gathering, and assembling open source components to build an open educational resource (OER), in this case a virtual 3D application, holds educational value. The research focuses on students whose interests cross disciplines of computer science, educational technology, instructional design, and game design.

Index Terms—Educational simulations, educational games, knowledge sharing, computers and education.

1 Introduction

THE increasing popularity of video games, and the billion dollar industry supporting their development and sales, has contributed to the use of video games for other purposes beyond entertainment, including simulation, training, and educational games. Technology for strengthening educational processes has been widely used for some decades now; however, the power of 3D game engines is an attractive alternative for the design and development of serious games aimed at educational purposes. From an academic perspective, the design and development of serious games presents two mainstream advantages: 1) it can be used for educating and increasing the skills of a multidisciplinary team of students responsible for the design and development phases under the supervision of a teacher, and 2) it can be used for educating people who will use the software after release.

A game engine, and the various components that work together within the engine, may be useful as an open educational resource. "The term Open Educational Resource(s) (OER) refers to educational resources (lesson plans, quizzes, syllabi, instructional modules, simulations, etc.) that are freely available for use, reuse, adaptation, and sharing" [1, p. 2]. By sharing knowledge and learning resources, the purpose of OERs can be achieved by improving access to learning opportunities. This paper discusses how the students involved with this exercise have created an OER from open source code and components.

Part of this discussion includes the challenges faced by a multidisciplinary group of students for developing a specific serious game engine and application. It is suggested that this process leads to valuable educational experiences for these students. Furthermore, it is shown that the use of open source components and tools play an important role in the educational process due to the inherent complexity in building a working application using pieces of code from different sources and communities. This is termed a Frankenstein approach to development. The students created a "new monster" out of pieces from other applications, while gaining a valuable experience in design, development, and participating in a dynamic and education-related community.

The aforementioned group was made up of undergraduate, graduate, and recently graduated students with different backgrounds (computer science, instructional technology, and art). They were involved in the design and development of a game for training emergency response personnel. The efforts of the software design and development team, one part of the multidisciplinary group of students, are discussed in the paper. The project can be compared to a computer science undergraduate capstone or independent study kind of activity, provided that opportunity exists to merge students with different talents. Developing and supporting such applications is part of an ongoing effort as evidenced by the number of grants aimed at sustaining this endeavor offered through federal and state initiatives. As part of this effort, the state Institute of Emergency Services & Homeland Security (IESHS) partnered with university researchers to build an "Emergency Service Training Simulation" to help in the training of first response communications and procedures. The Hazard Emergency & Accident Training (HEAT) project involves the creation of an elaborate 3D environment for multiplayer participation. This is helpful in building dramatic scenarios for experiencing and assessing emergency response personnel in a variety of situations. Using virtual 3D environments, the group of student design-developers created a module for emergency response units to practice crucial coordinated protocols in communication and action as part of a training simulation.

To accomplish the goals of the HEAT project, the group of students worked collaboratively in researching, gathering, and assembling components for building a virtual 3D application. It is claimed in this paper that this process holds educational value for students and teachers in diverse fields such as computer science, educational technology, instructional design, and game design because:

1. It provides a means to engage in a development community, often mandating the students' active participation in newsgroup and special interest group postings with other members [2].
2. It benefits the students to parse through code not written by them, extracting relevant pieces, modifying that code, and practicing debugging.
3. It requires a level of project management, juggling different pieces of different modules to eventually meet the goal of a cohesive project.
4. It requires interaction with subject-matter experts (SMEs), helping to prepare students for working in multidisciplinary environments and becoming effective members of design-development teams.
5. The final product holds potential for further study and development of simulations, games, and data visualization.
6. During software development, the students create a mental model of the learning process that affects the development of the software and their understanding of the final outcome.

2 Theoretical Perspective

2.1 Virtual Environments and Education

Virtual 3D environments are becoming more commonplace in their use for educational applications [3], [4]. The range of their use has been noted across several disciplines and targeting

• *The authors are with Utah State University, Logan, UT 84322. E-mail: {brett.shelton, marco.alvarez}@usu.edu, {jon.scoresby, k.chad.coats}@aggiemail.usu.edu, stowellt@gmail.com, mcapell@ionicdesign.com.*

Manuscript received 13 Feb. 2009; revised 12 May 2009; accepted 21 Jan. 2010; published online 24 Mar. 2010.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLTSI-2009-02-0011.
Digital Object Identifier no. 10.1109/TLT.2010.3.

different audiences [5], [6], in both formal and informal learning environments [7], [8], [9]. Emphasis has recently been given to complex, problem based and discovery types of learning within virtual 3D worlds, given their inherent properties that allow for rich and immersive interactions, task-based functionality and character empathy [10], [11].

Other projects have noted the value of how constructing virtual environments, and even game environments, can provide students with special design-related and group-based learning experiences [12], [13]. Most of these projects have relied on preexisting development platforms and environments. In other words, students work with a development tool to build new pieces of virtual spaces that are prerendered and prepackaged—which requires little understanding of how the development platforms operate. Here, a project is presented that relies on the creation of such a platform as a learning experience, building a development tool upon which other education-related projects can be based.

Game engines provide such a platform through a collection of modules of code that do not specify the game behavior or environment [14]. Game engines are independent of the virtual environments. The engine includes modules handling input/output such as networking, rendering, sound, drawing, and physics primitives for virtual environments. Game engines play a crucial role for creating realistic virtual environments. A software platform that meets the requirements can serve a broad range of science and technology applications, but developing an entire platform is a difficult endeavor in itself [15].

Building the game engine allows students to be involved from the beginning of the developmental process. This involvement ensures that students are familiar with all parts of the developmental system that is, in turn, structured upon all subcomponent parts. The developmental system can be represented as a mental model of the process created by the students when developing new software [16]. Learning models are similar in cognitive psychology to creating models for the development of software to perform specific functions. During the hands-on development of a tool designed to achieve certain goals, students develop a model that can be used to first get the job done and then justify whether the software will do the job. In reviewing the development process, students can learn three important things, 1) practical knowledge about whether the design will, in fact, achieve the desired outcome better than using existing tools without getting into the formulation process itself, (2) a mental model that shows how the tool should work to produce the outcome, and 3) a mental educational model of how learning occurs. These three outcomes not only promote learning during the beginning of a specific project, they also accelerate learning for subsequent projects so students are able to generate better solutions faster.

2.2 The HEAT Project

It is important to give emergency response personnel a dynamic way to train while being cost-effective and safe at the same time. For example, users can save training simulations and return to practice something that may have gone wrong during the saved simulation. They need an environment to practice procedures and protocol in a realistic setting using realistic communication and strategies. A virtual 3D environment offers an advantageous way to fulfill the needs for this type of training. The next step is to decide which 3D engine can provide the solution to a particular set of instructional goals; in this case, for emergency response training. Many off-the-shelf 3D engines are expensive; others do not have the needed capabilities. Building a 3D engine from the ground up ensures that functionality will exist for a particular training simulation. Control is given to the designers and developers who are not constrained by the lack of ability of other 3D engines.

Large software projects are often made up of many smaller components. Using existing software libraries can greatly speed up the main application development as developers can focus more on the application concept itself (the "core" code), rather than the needed supporting pieces. There are many nuances of successfully mixing core code with these third party libraries. Along with mixing code, a combination of skills and personnel are required to integrate instructional goals with design goals. The opportunity to learn about the different disciplines that make up a design and development group is also available. Programmers learn about graphics, modeling, and animation. Modelers and animators learn about technology requirements of their graphics and relevant portions of programming. Instructional designers sharpen their technology development skills, while artists and programmers learn about decision making, usability, and educational design.

The interdisciplinary student development team was made up of three main groups based in computer science, artists/modelers, and instructional designers. The team worked on the initial design document and planned how the instruction should be implemented into the simulation. The members of the team researched the libraries and implemented them in the 3D engine. Working with the core code, they built the engine by adding the different libraries they found on the web while also implementing their own code to make it perform the specific instructional goals of the HEAT project. The artists built the models (3D characters, houses, trucks, etc.) and worked on the textures (clothing and materials) used in the simulation. Those not primarily involved with development worked on building the instruction for the simulation, participated in planning meetings, and also helped with staying connected with the SMEs and clients.

Efforts were made to document the development of the 3D engine. Documentation included the process of choosing which components would be implemented during design-development iterations, the subsuccesses and setbacks. Especially helpful in reviewing this educational process was the careful recording of the design-development process as the project progressed, in order to improve the project experience for future students. Documenting the iterative process helped map student learning strategies and outcomes back to the stages of product design and development [17]. Bannan-Ritland [18] articulated a model for design research that combines the creative elements of development while adhering to research methods accepted within education (p. 21). While the Integrative Learning Design Framework (ILDF) is meant to provide a program-level perspective based on large-scale interventions, offering a smaller-scale intervention using this framework can make a significant contribution to informing the design and redesign of educational computer applications through iterative means. The value of using the ILDF approach lies within identifying the factors that impact one emerging design approach to instructional systems, in this case, a 3D game engine. One may then consider how the expectations of design and development of the game engine were filled (and unfulfilled) during implementation, helping inform the original design decisions. During these iterations, a mental model of the learning process revealed itself. From this experience, the team recognized they had actually created and followed a process or model that may be built upon to improve their current work and also something that can be used as a foundation for future work. It is through these research techniques, including documentation of design and development iterations, the recording of interactions between group members from various disciplines, and comparing the resulting end products with the corresponding design goals, that the findings are offered.

3 Evidence

The discovered developmental model or 3D Engine Component Evaluation Model (3DECEM) is discussed and examples are given throughout the paper. This model has helped the team as they developed the 3D engine and the HEAT simulation by providing opportunity for reflection and guiding their efforts to make the software perform in the best possible manner (see Fig. 1).

As part of the design-development process, the students first had to research what kinds of resources existed and what

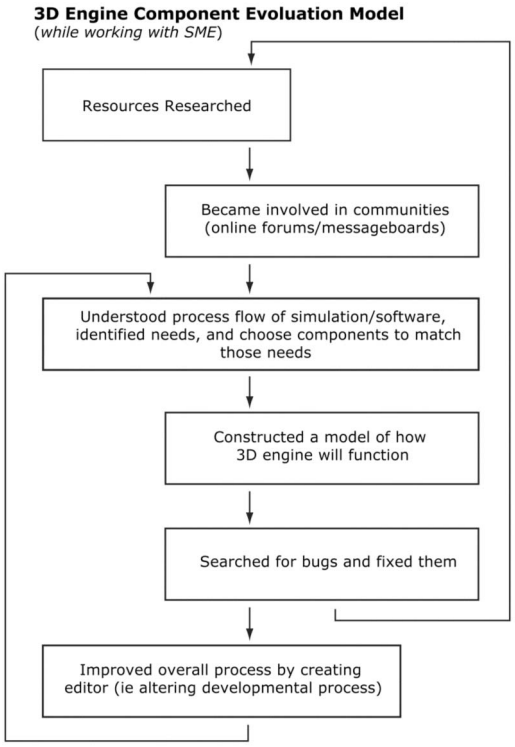

Fig. 1. Mental learning model for 3D engine development.

is available for them to use. Students looked at commercial components as well as open source and free use components as part of the data gathering process. While not nearly exhaustive of the available engines, the following tables show excerpts of their research into existing "game engines" and other physics components they determined would be needed to fit the goals of their project (see Tables 1 and 2).

Open source components do not have licensing fees, and often have support from a large and active community. For example, Ogre 3D has very active online forums and the original author of Ogre 3D often participates in these forums by answering questions. When choosing an open source component, the team carefully examined the license accompanying the source code to make sure it was not too restrictive for the team's needs. For example, the General Public License (GPL) states that any software or application that incorporates the code must also fall under the GNU license. For HEAT, the team members did not necessarily want to be forced to use such a restrictive license, so they tried to choose libraries that would allow them to either statically link to them ("merge" their code with that of HEAT), or dynamically link to them (by using the library as a .dll file) without being forced to take on their license [1]. The Lesser General Public License (LGPL) is one such less restrictive license. The team also had to make sure that there were active support forums for the open source components chosen. Some components and support forums which were active in the past are not currently used as much.

It was necessary to choose a group of technology components to support the instructional design of the scenario. A flowchart was built, restructured, and rebuilt through a number of iterations that would help shape how players and the teacher will make the best possible use of the simulated environment. The sequence of the simulation scenario includes features of playback and debriefing to review critical decisions, and offer multiple outcomes based on those decisions. Based on information in these tables and other research, the student team determined Ogre 3D and Ageia PhysX were the best options to build a 3D engine that would have the functionality and ability to do what was necessary for training the emergency personnel.

TABLE 1
Comparison Table of Possible 3D Engines

	Quake 3	Ogre 3D	Unreal2 Runtime	Unreal3 Engine
Initial Price	Free	Free	$8900/per developer, $5000 for C++ headers	?
Price to Sell	Free (GPL)/$10K	Free (LGPL)	$350,000	$1,000,000?
Physics Capability	virtually non-existent, hard to interface to external library	External library, free or otherwise	Karma(built in physics), doesn't seem robust enough	Built in Ageia PPU, good physics middleware
Development Tools	Yes, Q3Radiant, hard to use	No	Yes, UnrealEd is powerful	Yes! Best tools of all
Graphics	5 years old, not up to modern standards but adequate	Potentially top of the line	Better then Quake 3, but still several years old	Top of the line

TABLE 2
Comparison Table of Possible Physics Engines

	Ageia PhysX	Newton Game Dynamics	Bullet	ODE Open Dynamics Engine	Havok
Price	Free for PC development	Free	Free	Free	Free for non-commercial PC development
License	Proprietary	Restricted	Zlib	BSD Berkeley Software Distribution	Proprietary
Source code is	Closed (character controller code released)	Closed	Open	Open	Closed

With the goal to create cutting-edge technologies, the team felt that using these systems would give users the best experience possible. HEAT is also meant to be used by others as well as for the development of educational materials. One of the biggest reasons an off-the-shelf 3D engine was not chosen was financial. If an engine can be built from open source or free use software, then the final product could be passed on to fellow student developers for further use.

The virtual 3D environment was built by using mostly open source libraries or software packages that were found and downloaded from the Internet. Starting with the open Ogre 3D graphics viewing solution [19] as a base, the student development team added a number of additional libraries to turn Ogre into a functioning 3D "game engine" called the HEAT editor (see Figs. 2, 3, and 4). HEAT is a compilation of dozens of components gathered from a variety of free use and open sources, creating an application that mirrored a Frankenstein monster—it may not look very pretty, but it seems to work.

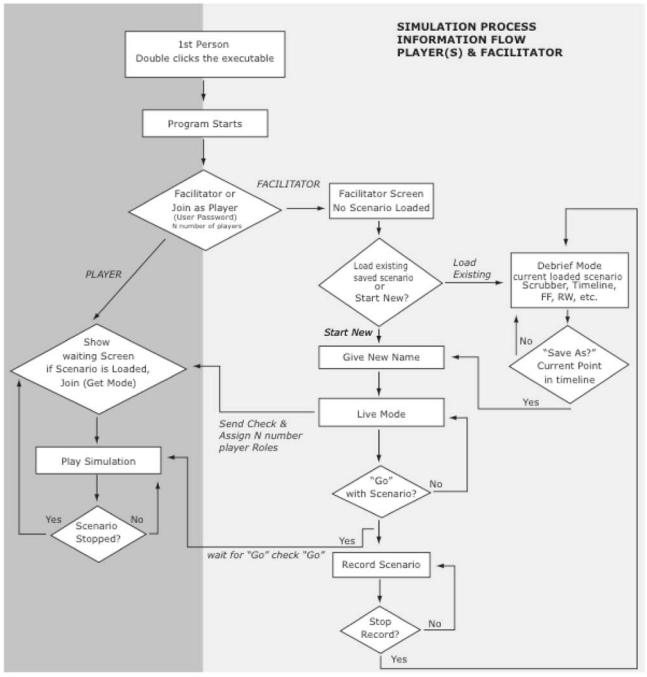

Fig. 2. HEAT fire fighter scenario sequence.

Unfortunately, using off-the-shelf components can lead to problems when used naively. For example, initially the team chose to use the wxWidgets GUI library to set up the main window, tabs, etc. This library wraps the lower-level Win32 windowing code. WxWidgets was not designed for high-performance applications like games or simulations, but rather aimed to make windows GUI programming more convenient and easier. To this end, some of the ways the library handled windows input events was not as efficient as the team wanted. Because of this inefficiency, the team tried going to the other extreme and started creating the windowing code from scratch based solely on the low-level win32 C code. This worked, and allowed full control for speed over how the engine updates ran. However, the team soon learned that coding all of the GUI components they wanted was extremely laborious. Eventually, having now both the knowledge of the high-level wxWidgets library and a deeper knowledge of how that library worked "under the hood," the team combined the convenience of wxWidgets with the speed of low-level code by utilizing aspects of both.

Another example of how naively using a code library without understanding the underlying mechanisms is Ogre 3D. Ogre 3D aims to hide a lot of the tedious lower-level graphics code by providing higher-level access and abstracting the main ideas. While this makes graphics programming much easier, it can soon lead to performance problems. For example, the team was working on finding a way to render burn marks on surfaces that had come in contact with fire. This rendering required dynamic updates for every rendered frame. Subsequently, the team had to learn how the underlying DirectX graphics Application Programming Interface worked (API, third party code that HEAT "hooks" into for functionality) to understand how Ogre 3D was using it. Knowing how the API worked is important to achieve interactive rendering frame rates. The way Ogre 3D performed was not to blame for the team's performance problems but rather the team's lack of knowledge of how Ogre 3D functioned. As a way to manage the generic capabilities of external components, their varied interfaces and uses, and to provide a uniform interface through which objects created in the engine can rely on, the development team often attempted to write "wrappers," encapsulating the existing functionality of the external libraries while providing a standard for use by HEAT. In a sense, using higher-level convenient "wrappers" can lead to a false sense of security

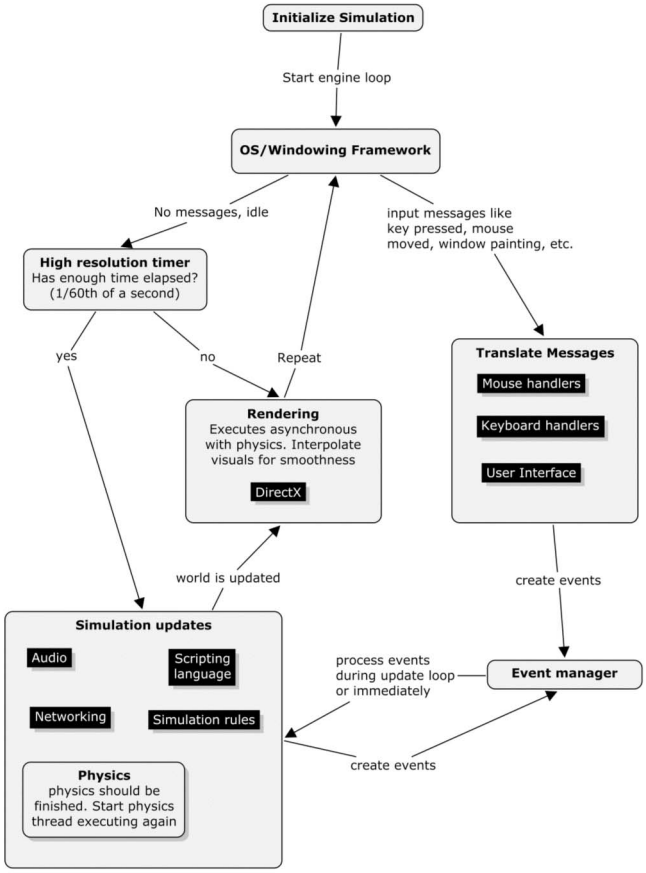

Fig. 3. HEAT engine components and execution process.

that allows one to achieve quick results, but not necessarily have a full understanding. Overall, this can lead to problems as the project gets larger in scope.

As progress on HEAT continued, the team had to consider the ideas and desires of the SMEs when deciding what components to add to HEAT. Because the fluid nature of discussions with the SMEs and the knowledge input was an iterative process, the team's approach to deciding which "parts" to borrow for HEAT had to

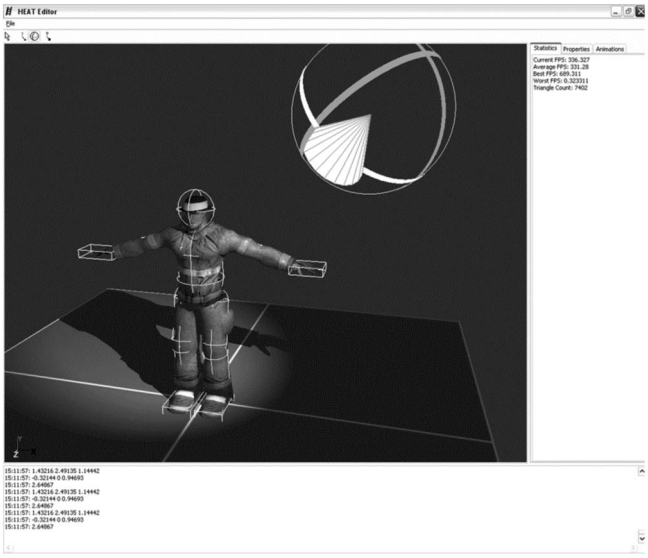

Fig. 4. Image of a fire fighter in the new editor, showing shadows, being textured, and having physics capsules applied.

take uncertainty into account. The team relied on the ability of each external library to meet not only the requirements they were currently faced with, but also had to consider the suitability of the library in a more general context, while attempting to anticipate possible future requests and even future projects before making a final decision. To make these decisions, the team consulted the documentation for each library as well as any public forums dedicated to the component and online development communities that focused on the development of 3D systems.

As development continued, the programming team found it helpful to construct a model that increased their understanding of how a 3D game engine worked. This model helps show the process of the game engine running after the user starts the simulation. Integrating one of the open source libraries, a script interpreter, allows one to edit the simulation during runtime. This can be seen in the simulation updates node in the model. Rendering the world interpolates between the last known state of the world and the current state in small increments, depending on how much time elapses during the 1/60th interval. Interpolation makes objects appear to move smoothly. Interpolation is necessary, because the graphics card is constantly drawing the scene (much faster than 60 times a second), and since objects' positions are only updated at 1/60 second intervals, it makes objects appear to move in a blocky fashion. By interpolating, it allows objects to move in smaller increments while waiting for the next update.

Further development and testing allowed the team to find bugs in the most current version of HEAT. After finding and fixing the bugs, the team realized that the system was not stable enough for them to feel good about moving forward with the development of the simulation. Team members worked on fixing the main 3D engine by programming it to be more stable and run more efficiently. The developers also completed a new tool that allows the modelers to test their models before each is imported into the virtual environment. This new visual editor will save time for the modelers by working with visuals as they appear and are positioned in the HEAT environment. The new editor takes pressure off of the programmers so they can focus on programming the simulation and not reporting on how to fix the models.

4 Findings and Discussion

Listed in the introduction are six educational components that together provide evidence of the value in creating a 3D virtual environment for learning. Listed below are examples illustrating how the value emerged for the members of the student development team.

1. *Engage in a development community*: In an attempt to prevent the slowdown with the rendering and physics, the student development team investigated more efficient scene management schemes. An Ogre community member was working on something similar and the team found the answers in the online community forums helpful. The team had the chance to reciprocate some of this help in several cases. For example, the team found a helpful post regarding how to set up rag doll physics animations. After implementing this technique, others posted questions regarding how to accomplish the technique, and the team members assisted them by referring them to the earlier post and giving assistance.
2. *Forces students to improve their parsing and code editing skills*: When researching what GUI library to use, the lead programmer found two libraries and evaluated both. Finding what he liked from both libraries, he took those parts, edited the code and added them to HEAT. Another team member has researched code and found ways of helping the engine become more stable and efficient. Another example is that a team member was required to set up documentation for the project, and so investigated project management software, learned more about wikis, and refamiliarized himself with Adobe Dreamweaver and Adobe Photoshop to create webpages and graphics. By using open source software, the development team also had an additional resource in the code itself, which gave the team members insight and experience with the tools and code they were using and how those tools were actually built.
3. *Practice project management:* When working on a large project with multiple programmers, it can be easy to lose track of the most current work. To address this problem, the development team used the open source software *Subversion (SVN)* to keep track of their progress and their most current versions of code. The team also participated in weekly meetings to discuss progress and known issues. To improve development efforts, the development team implemented a weekly developers' meeting to help each other with the known issues.
4. *Interaction with SMEs/multidisciplinary work*: From the beginning of this project, the development team kept in close contact with local fire stations and IESHS to gather important information regarding how command centers communicate with those fighting fires in emergency situations. Gathering this information helped in the design process for the educational tool—what features and functions would be needed within the game engine. After installing phase one of HEAT, the team has stayed in contact with IESHS to gather feedback on how to change and improve the current simulation.
5. *End product can be used for further study by team members:* In addition to the aforementioned HEAT simulation and other such educational uses for the engine, students involved with the development of the engine anticipate applying the software in other capacities, as diverse as commercial simulation or game applications and the visualization of data in other disciplines. Examples include a 3D hostage training simulation, 3D fire forensics simulation, and converting educational text-based games into 3D games.
6. *Developing a mental map of learning during the software development process:* During team meetings, team members often discussed why one process worked and another did not. By discussing software developmental processes, the team members have created a mental learning model of what is needed to create a 3D engine and possibly use that model for future work.

The student developers often worked around issues that arose by engaging in trial and error, "make it fit," and hacking methods to get this application to function. These experiences gave them the opportunity to learn new ways and sometimes much better ways of producing functioning parts of HEAT. On the other hand, "making it fit" also led to problems. At times, the team would force the simulation to function due to time constraints and not implement the new methods into the engine for a more general purpose. In later development, the team realized that the forced functionality limited the power of the 3D engine and created more work for them later on in the process. When these and other problems arose, many times the team found solutions by trying something new they learned from a forum they frequented. Other times they gathered information that gave them ideas on how to complete the next component of the engine.

In instructional technology and design projects, student developers are often required to take a step back and look at the big picture—as it exists, and how they want it to progress. During the development cycles, some functionality may not be possible to complete, which in turn affects the nature of the instruction of the end product. This requires an iterative design-development structure within the model of instructional design. But this also gives the instructional designers the chance to think about what is

desired, what is good, and what needs to be changed in the design for a good educational tool.

Development of a 3D engine gives insight into what it takes to build good educational technology. By participating in the development of this new technology, the students have become engaged and active participants in the open source community during the development. In this way, the students' learning experience returned full cycle. The students used the community to learn how to program certain functions. When they developed a new technique to perform these functions, the students passed that knowledge back to the community to help others. They learned how important this process is to sustain and build an open source community; one in which the students would eventually become important members.

The development team continually struggles with the attempt to build not only the generic 3D engine that was originally envisioned, but also to construct a training simulation using that technology at the same time. Such a pursuit at times feels very intimidating, but has endowed the development team with a better perspective on the software development process as a whole and has reinforced the classic engineering axiom of solid early design and iterative follow-through. This axiom was fully realized by the development team when planning the functionality of the 3D engine. Sometimes the current need in functionality of the simulation dictated the functionality of the engine. This type of development has made part of the functionality of the engine HEAT-specific. The team recognized that it was much better to plan and develop the functionality of the engine before integrating the functionality into the simulation. This process of design and development would ensure generic functionality of the 3D engine for future simulations. The setbacks the team has encountered through trial and error have refined their process of component selection, wrapper design and implementation, and have impressed upon them the importance of knowing when to pursue another avenue to accomplish their goals.

Even though the team has struggled with the development of the engine and simulation software, the mental model (3DECEM) process identified during this experience has given the team members knowledge and skills that will help them in their current and future work. The team members have gained a practical knowledge about the design process and if that process leads to good software design. As development continues on these projects, the team often reviews their mental model to make the needed changes that ensures the software performs as desired. With this model, the team members have an educational experience to build upon and now know what must be learned and what it takes to build a process for good design. The knowledge gained from this experience comes from identifying what does and does not work, thereby accelerating learning and helping students find good solutions.

5 Future Research and Educational Importance

The result of the project was the HEAT application that provides experience for emergency leadership and a safe environment to help improve teamwork and team communication in an emergency situation. The opportunities to repurpose the game engine exist—it can potentially be modified into a program for training with other emergency scenarios such as brush fires, avalanche search and rescue, hazardous materials response, and other natural catastrophes. Because it is a "game engine," once development of a sound infrastructure is completed, new designs for educational games and simulations may be implemented. Following the process of the student development team has offered new insights into how the building of a complex, open application is of value. The process of using open source code and components to create an OER may influence the way computer science students learn and are taught as open source material becomes even more prevalent.

Future research will concentrate more on how student designers and developers can leverage the open resources available to them, and give back to the open source community of developers. This new research may include how often team members visit forums, what level of help the open community gives, and to what experiences do the developers assign value. Other research could include comparison and verification of the efficacy of the development model discussed in this paper to other development models used during software engineering. Understanding more about how students are successful in this type of project-based curriculum will assist in creating an environment and scenario that benefits cross-disciplinary projects within computer science, educational technology, art and design.

Acknowledgments

The authors wish to thank Jeff Maxfield, Dennis Goudy, James Hunter, Greg Rynders, Gary Noll, Hugh Connor, Devon Bartlett, Dave Smellie, Alan Hashimoto, and Lori Postner for their help and support with this paper. This work was supported in part by an IMRC USU Innovation Grant, UVU, and the Utah Institute for Emergency Services and Homeland Security. For more information about HEAT and other gaming simulation topics, visit http://imrc.usu.edu.

References

[1] S. Gurell, *Open Educational Resources Handbook for Educators 1.0.* Center for Open and Sustainable Learning, 2008.

[2] D.H. Jonassen and S.M. Land, "Preface," *Theoretical Foundations of Learning Environments,* D.H. Jonassen and S.M. Land, eds., pp. iii-vii, Lawrence Erlbaum Assoc., 2000.

[3] K. Squire, "Wherever You Go, There You Are: Place-Based Augmented Reality Games for Learning," *The Educational Design and Use of Simulation Computer Games,* B.E. Shelton and D. Wiley, eds., pp. 264-295, Sense Publishers, 2007.

[4] R. Van Eck, "Six Ideas in Search of Discipline," *The Educational Design and Use of Simulation Computer Games,* B.E. Shelton and D. Wiley, eds., pp. 31-61, Sense Publishers, 2007.

[5] J.P. Gee, *What Video Games Have to Teach Us about Learning and Literacy.* Palgrave MacMillan, 2003.

[6] L. Barron and J. Bransford, "The Jasper Experiment: Using Video to Furnish Real World Problem Solving Contexts," *Arithmetic Teacher,* vol. 40, pp. 474-479, 1993.

[7] C. Aldrich, *Simulations and the Future of Learning: An Innovative (and Perhaps Revolutionary) Approach to E-Learning.* Pfeiffer, 2004.

[8] Y.B. Kafai, "Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies," *Games and Culture,* vol. 1, no. 1, pp. 36-40, 2006.

[9] C. Steinkuehler, "The Mangle of Play," *Games and Culture,* vol. 1, no. 13, pp. 1-14, 2006.

[10] S.A. Barab, M. Thomas, T. Dodge, R. Carteaux, and H. Tuzun, "Making Learning Fun: Quest Atlantis, A Game without Guns," *Educational Technology Research and Development,* vol. 53, no. 1, pp. 86-107, 2005.

[11] C. Dede, D. Ketelhut, and B. Nelson, "Design-Based Research on Gender, Class, Race, and Ethnicity in a Multi-User Virtual Environment," *Proc. Am. Educational Research Assoc. Conf.,* http://muve.gse.harvard.edu/muvees2003/documents/AERADede04.pdf, 2004.

[12] J. Robertson and J. Good, "Story Creation in Virtual Game Worlds," *Comm. ACM,* vol. 48, no. 1, pp. 61-65, 2005.

[13] S.A. Barab, K.E. Hay, M. Barnett, and T. Keating, "Virtual Solar System Project: Building Understanding Through Model Building," *J. Research in Science Teaching,* vol. 37, no. 7, pp. 719-756, 2000.

[14] M. Lewis and J. Jacobson, "Game Engines in Scientific Research (Introduction)," *Comm. ACM,* vol. 45, no. 1, pp. 27-31, 2002.

[15] A. Mol, C. Jorge, and P. Couto, "Using a Game Engine for VR Simulations in Evacuation Planning," *IEEE Computer Graphics and Applications,* vol. 28, no. 3, pp. 6-12, May/June 2008.

[16] J. Piaget, *Structuralism.* Harper & Row, 1970.

[17] T. Stowell, J. Scoresby, M. Capell, and B.E. Shelton, "A Process for Utilizing Readily Available Software Libraries to Create a 3D Simulation Game," *Int'l J. Gaming and Computer-Mediated Simulations,* vol. 1, no. 4, pp. 20-49, 2010.

[18] B. Bannan-Ritland, "The Role of Design in Research: The Integrative Learning Design Framework," *Educational Researcher,* vol. 32, pp. 21-24, 2003.

[19] "OGRE 3D: Open Source Graphics Engine—What Is OGRE?" http://www.ogre3d.org/index.php?option=com_content&task=view&id=19&Itemid=79, 2010.