Utilizing Readily Available and Open Source Libraries to Create a 3D Game Engine

Tim Stowell, Utah State University, USA
Jon Scoresby, Utah State University, USA
Michael R. Capell, Utah State University, USA
Brett E. Shelton, Utah State University, USA

ABSTRACT

Market 3D engines have all the capabilities needed for developing full-featured 3D simulation and game environments. However, for those in education and small business, it remains a formidable task to acquire the resources needed to purchase or create a development platform with cutting-edge capabilities. Leveraging existing and open-source software libraries can greatly enhance the main application development, freeing developers to focus more on the application concept itself rather than the needed supporting pieces. This article explores the nuances of successfully mixing core code with these third-party libraries in creating a fully functioning development environment. Many steps with accompanying checks-and-balances are involved in creating a game engine, including making choices of which libraries to use, and integrating the core code with third-party libraries. By offering insights into our open source driven process, we help inform the understanding of how game engines may be generated for other educational and small-budget projects.

Keywords: Game Engine, Virtual Environments, Reusable Libraries, Open Source Code Organization, System Integration, Serious Games

INTRODUCTION

Market 3D engines have all the capabilities that one would need to build full-featured 3D environments. Those that develop these market 3D engines often invest vast resources of money and people and, therefore, market 3D engines cost a significant amount to license. Educational institutions rarely have the funds to purchase a current top-of-the-line engine, choosing instead to research alternate avenues. These avenues focus mainly on current open source engines or older engines that could be purchased for a lower price (Cruz, Wieland, & Ziegler, 2006). These engines would not support many current development needs, or were too dated to use for cutting-edge applications. Choosing instead to create a 3D engine from scratch allows the
flexibility to customize functionality and development structure.

Our group teamed up with the Institute of Emergency Services and Homeland Security (IESHS) to develop software they could use in their training. Generally, our task is to build an incident command training simulation for firefighters. The simulation has been generated as a multiuser first-person perspective training program. Even though the simulation is designed for multiple user participation, the training is meant for the incident commander of that emergency situation. Currently the training will take place in the IESHS computer lab, but future development will allow this training to be expanded to firefighters to participate in the simulation via the Internet. The simulation is partitioned into two primary roles, a facilitator and client user. The facilitator starts and records the simulation whereas the client is meant for the individual firefighters participating in the simulation. The facilitator can also load previously recorded scenarios for learning. Part of the development is to expand this training to EMT, police officers, and other emergency personnel. To encompass all of these areas, we use the moniker Hazard Emergency & Accident Training (HEAT) to describe the needs of our game engine. The HEAT engine aims to be a full-featured, 3D training environment. As such, it requires a number of pieces that many 3D game engines also share including:

- 3D graphical capabilities
- VOIP (Voice Over Internet Protocol)
- Sound and physics

Developing software for any one of the above components alone is a large task. To save time and money, it is often desirable to use “off-the-shelf” software components that already include the needed functionality. However, there are many issues to consider when making the choice to use available libraries because of potential issues that may exist within open code. For the HEAT engine, we used the following process when building necessary functionality of high-end systems, with each point to be described in more detail throughout the article:

1. Perform background research into the library functionality
2. Make choices based on that research
3. Start developing core code (code developed in-house for the main project, not relying on off-the-shelf components)
4. Begin an iterative process of integration of core-code with third-party libraries (Shelton, Scoresby, Stowell, Alvarez, Capell, & Coates, in press)

We next discuss the process we used for searching and deciding on what open source libraries to use to develop the 3D engine. Reading the forum discussion boards helped in making decisions—what has worked for others, and which library code to avoid (see Figure 1). It is difficult to contrast our particular open source approach with existing ones. There may be documentation available that discusses how to build 3D game engines, but there are few if any that discuss how to mix and match open source libraries for 3D engine development. Most companies keep their processes a closely guarded trade secret; therefore, we must learn from forums, discussion boards, and through implementing scientific method. We emphasize throughout the article the focus on open source techniques and shared information, which can be generally considered different than the development processes used within large, in-house development teams.

---

**BACKGROUND RESEARCH INTO THE LIBRARIES**

There are extremely large numbers of software projects available on the Internet; one only has to browse the popular site sourceforge.net to get a feel for vast amount of code that developers have made available. The Internet contains multiple software libraries of almost every type imaginable, and most are just a click...
Related Content

DAU CardSim: Paper Prototyping an Acquisitions Card Game
David Metcalf, Sara Raasch and Clarissa Graffeo (2010). Serious Game Design and Development: Technologies for Training and Learning (pp. 81-101).
www.irma-international.org/chapter/dau-cardsim-paper-prototyping-acquisitions/41068/

Designing Games for Learning
www.irma-international.org/chapter/designing-games-learning/20143/

Towards the Development of a Games-Based Learning Evaluation Framework
Thomas Connolly, Mark Stansfield and Thomas Hainey (2009). Games-Based Learning Advancements for Multi-Sensory Human Computer Interfaces: Techniques and Effective Practices (pp. 251-273).
www.irma-international.org/chapter/towards-development-games-based-learning/18799/

Investigating Real-time Predictors of Engagement: Implications for Adaptive Videogames and Online Training
www.irma-international.org/article/investigating-real-time-predictors-of-engagement/125444/

Design Factors for Effective Science Simulations: Representation of Information
www.irma-international.org/article/design-factors-effective-science-simulations/2159/