

Java Based Simulation Content in E-Learning Development & Deployment Issues and Options

Author:
Christopher P. Giordano

DiSTI
Product Manager
11486 Corporate Blvd.
Suite 190
Orlando, FL 32817 U.S.A.

+1.407.206.3390 ext 29
cgiordano@simulation.com
<http://www.simulation.com>

With today's stringent e-learning deployment and security requirements, coupled with an ever growing need for higher visual and interactive fidelity and real-time simulation content, e-learning content developers are looking for viable, reusable and non-proprietary COTS options. This paper explores issues and options for e-learning content development and secure non-proprietary deployment.

E-Learning Overview

E-learning is Computer Based Training (CBT) for the enhancement, retention, utilization or initial acquisition of a required skill set. E-learning spans the traditional set of virtual object development from Courseware and Virtual Maintenance Training to Operational or Procedural Training. Typical deployment of e-learning solutions has been for classrooms, as web-based applications, or even on portable or mobile devices to be used in the field. These e-learning applications can either be automated or instructor assisted. Many applications utilize both which is referred to as a “Blended Curriculum”.

New concepts been designed which encompass both Courseware and Virtual Maintenance Training applications into one single blended solution. Blended e-learning solutions allow developers to reuse content form one presentation format (e.g. web-based to embedded training) in another with changes only in the environment that uses the assets, and not to the content itself. This dramatically enhances the consistency across training platforms while decreasing the time to market and improves ease of maintenance for a training system. The focus of this paper will be on the assets used to create this simulation content for e-learning applications. In order to follow this concept, a general knowledge of the learning system architecture must be discussed.

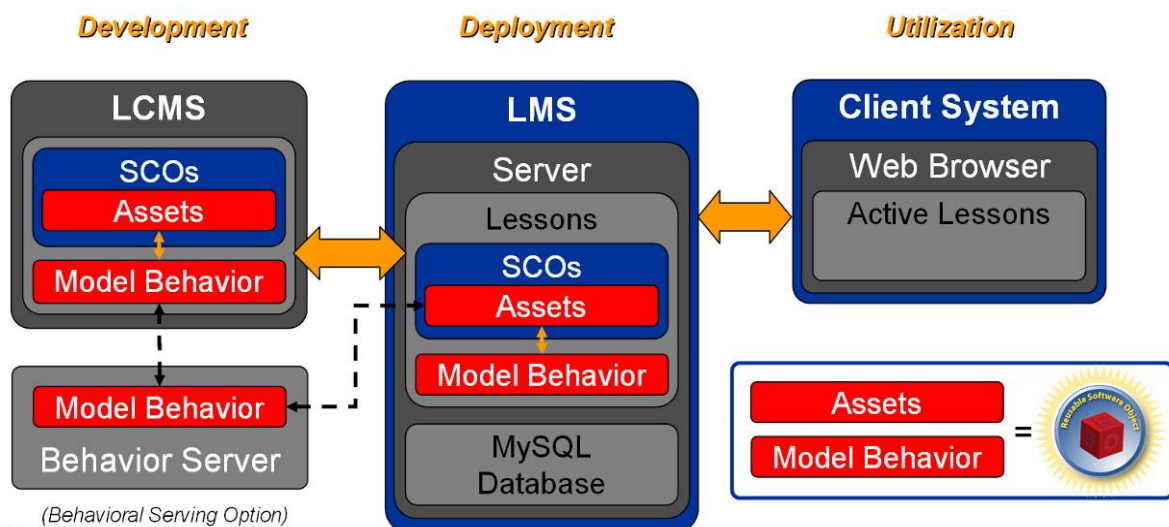


Figure 1 – Learning System Architecture

There are three phases for discussion: Development, Deployment and Utilization. The development phase typically happens in conjunction with the Learning Content Management System (LCMS). This is where the learning objects and assets intended for reuse are stored. Instructional designers use the content housed on the LCMS to build the lessons that are deployed to the Learning Management System (LMS). A LMS is essentially a server housing a MySQL database, which is typically used for account management, content sequencing, and the lessons that will be served to a client system. Lessons are usually conformant to the SCORM packaging and deployment standards for lessons and lesson content, and define Sharable Content Objects (SCOs) containing assets with the model behavior. The LCMS may or may not reside on the same system as the LMS. Model behavior is any information that governs the innate use of an asset. Also, it's important to note that the model behavior may reside on the LMS or on a separate behavior server. The LMS feeds the SCOs, containing Assets and Model Behavior, to the Client System which accesses the e-learning solution through a web browser.

Assets are the focus of this paper and for the purposes of this discussion will be termed Reusable Software Objects (RSO). RSOs are modular software components that can take many forms ranging from simulations, to network interfaces, and include any function module that executes within the defined framework of a SCO or on any supporting server. This paper will be focusing, in part, on Human Machine Interface (HMI) objects that can contain the visual aspect and the innate behavior of the HMI in one reusable component. RSOs may also contain support for training scenarios such as failure modeling and related performance evaluation mechanism or scoring logic that is required. With the development philosophy discussed in this paper, model behavior can be a separately packaged,

deployed or hosted with regard to the Asset to allow for a smaller HMI RSO. This also serves to make each RSO more flexible in its potential reuse within other environments and supports a variety of distributed learning environments. For example, in figure 2, an individual instrument is inserted as an RSO into a panel; the panel is then inserted into a cockpit and the cockpit into the aircraft. Each part of the aircraft is its own individual component that lends itself to easily reconfiguring the same design for multiple aircraft variants because each RSO contains its own innate behavior. Once all variants are created, the student now has a reconfigurable asset for training on any aircraft variant in any OpenGL based e-learning environment.



Figure 2 – Reusable Software Object

This methodology is a departure from legacy e-learning solutions that are based on complex scripted animations that are costly to produce and maintain with the additional complication of large download sizes. The distance learning community is evolving away from complex versions of these legacy animations, to fully interactive simulation-based content. Simulation content differs from animations in that they allow the learner to fully immerse themselves in a free-play supporting, interactive 3D simulation that behaves identically to the real world object. These simulations are not bound by the restrictions imposed by a scripted animation framework.

Leading this migration is the U.S. DoD which has been striving for Higher Fidelity in their e-learning solutions while embracing software engineering principles, such as reusability and modular content. The trend has been toward modular, fully reusable content with natural navigation and interaction that can be developed once and used in several different visual environments. The premise is that these objects can be maintained quickly and easily and allow for greater consistency across training platforms. The use of existing source data for asset creation (such as 3ds Max®, OpenFlight™, Designer's Workbench™, CAD & CATIA Data Etc.) has also been of primary concern to further reduce costs and shorten development times for an increasing complex, highly technical, training environment. Efforts to reuse content is also extending into the Interactive Electronic Technical Manual (IETMS) domain is also receiving considerable attention in the form of standards cooperation efforts between SCORM and S1000D™. S1000D is an international specification for technical publications developed by the Technical Publication Specification Maintenance Group.

The use of detailed source data leads to improved fidelity within lesson assets. This existing source data can be accessed through modern Commercial Off The Shelf (COTS) simulation content

development tools to create fully interactive 3D Content with the natural navigation driven by the requirements for higher fidelity.

Natural Navigation is a technique that supports free-play content. This allows for a more realistic interaction with Assets, allowing the student to complete the wrong action as they might in the real-world. Animations derived assets are too lock-step and usually require a great deal of time and difficulty to update and maintain as they require timeline renderings that must be changed to add behaviors. Trainers and students are looking for a more natural interface to learning objects and developers are looking to have software objects that can be reused throughout the life cycle of an object, including the e-learning field.

E-Learning Content Assets – Development and Deployment Issues

There are several major issues that a developer of simulation content, for e-learning, will face. Hardware performance, object size constraints with regard to bandwidth limitations, proprietary solutions and players, and old unsecured technology are among the most common obstacles to overcome. There is also the issue of LMS integration and any possible future issues with migration to a newer LMS that can handle heavy student population growth and the need for feature enhancements.

In consideration of hardware performance, the developer must determine a baseline system specification in terms of processing power, visual performance and network capacity and verify that the courseware will function acceptably within that system's parameters. For example, a student station configuration that entails a minimum of Pentium III 750 MHz CPU, 128Mb Memory, 16 Gb Hard Drive and an AGP 8Mb Video Card, as recommended by Aviation Industry CBT Committee (www.aicc.org) may have difficulty running higher fidelity simulation content. While systems with such poor performance characteristics are increasingly rare, the presence of such legacy hardware within older facilities forces a tradeoff for the developer in terms of high fidelity versus system commonality support.

Discussions of performance and capacity inevitably leads to the issue of facility infrastructure in terms of network availability and capacity and these issues will impact size restrictions for the courseware and the contained RSOs. When looking at available bandwidth, RSO size constraints must be imposed and carefully monitored for deployment hence assets must be as small as possible. The smaller an RSO must be, the less fidelity it will have and simulation content has historically been very large. Rendering performance versus RSO size for higher fidelity content must also be taken into consideration. Student stations need to have the hardware necessary to run high fidelity content and the developer must again code to the lowest common denominator for all possible configurations of student stations and clearly define the system's hardware and software requirements. These issues are for more manageable today than in the past with more sophisticated tools and design techniques that allow complex simulations to be built in ways that do not burden the network as in the past. It is also possible to package content in more efficient modular components that deploy only as needed versus the legacy method of dumping the entire model to the student station in one large file.

Instruction development firms also need to consider staffing issues as newer forms of training content move to the forefront of our industry. Developers may need to learn the basics of a new coding language for visual development in a COTS toolkit. Languages that are considered proprietary to a specific toolkit should be avoided with preference given to open COTS technologies. ANSI standard C, C++ and Java, are commonly accepted and understood programming languages throughout the software and web development industries. In order to interpret and draw objects, behaviors and interactions, COTS tools will typically have their own proprietary players. The issue with proprietary players is that the e-learning content developer rarely knows what the player is internally executing or accessing. For this reason, and in order to mitigate the risk of a compromised network or system, secure environments have strict guidelines on what types of controls or players the student will be able to download and use. Even when selecting a player as common as Flash or a language as recognized as Java, version compatibility issues often confound developers when deploying to a broad audience or customers in excessively constrained operating environments.

Next, there is the issue of "Sunset Technology" or older technology that has become or is in the process of becoming obsolete. Microsoft Active-X controls, for example, are heavily relied upon for

content players and have historically been perfect places to mask malicious content mostly because they can easily gain access to system resources and manipulate them in most any way. Often the problems are not malicious but rather come from version conflicts with other system software which is exacerbated by automatic updating mechanisms. Though this risk is somewhat mitigated by the introduction of digital signatures and the notion of only using trusted content servers, there is still some level of risk. Most e-learning environments are governed by strict security guidelines and do not allow 3rd party proprietary players without first passing rigorous and costly verification and testing procedures. Java and Microsoft's .NET technology attempts to eliminate this problem by using what they call "managed code" that limits access to system resources. Even these solutions are not perfect as both technologies have a history of problems with backwards compatibility with content developed using earlier versions of these tools. Developers, however, are still faced with the need to migrate existing Active-X content to these newer environments and address these concerns as best they can. These newer alternatives are more secure relying on technology, such as interpreted byte code, which does not allow direct access to system resources.

Lastly, there is the consideration of LMS integration inconsistencies. As of July 15th 2006, there were roughly 24 SCORM 2004 compliant LMS's *(Joint ADL Co-Lab presentation). It is important to note that not every LMS provides consistent services or performance. They all have their strengths and weaknesses and one of the realities of the current SCORM world is that LMS portability is far from assured. The SCORM 2004 standard focuses on data delivery, content sequencing and the organization of student performance and feedback data. With so many LMSs and flexible standards governing their support and use of Assets, any solution based on a single LMS may be incompatible with another LMS making migration from one LMS to another a potentially timely and difficult process. For instance, if the developer selects to use a LMS that is fit for 5,000 users and has the required features at that point in time, then needs to migrate their e-learning content, assets and lessons after implementation to a new LMS with larger user capacity and different features in the future, there may be inconsistencies with how they handle loading, serving and interacting with assets that are more sophisticated than simple text and images.

E-Learning Content Assets – Development and Deployment Options

Several options for creating assets are now available to developers as a result of recent advances in both hardware and software technologies. Hardware has become increasingly smaller, faster and less expensive. With the introduction of dual and quad core CPUs to the home personal computer (PC) market at affordable prices, hardware now supports a myriad of technologies that were previously unreachable by the common PC.

Newer COTS development tools have had very positive effects on the learning object content developer's (or asset developers) ability to rapidly create and maintain high fidelity e-learning simulation content. Software development toolkits now exist that allow learning object developers to create smaller HMI RSOs with much higher fidelity simulation content and real-world interaction behavior as opposed to the legacy animations available for e-learning asset development. The smaller sizes allow for higher portability and efficient deployment via the internet. Also, newer object oriented toolkits offer extensible and highly reusable component architectures, allowing for both vertical reuse, in the life cycle of one craft or vehicle's program, and horizontal reuse, in the life cycle of a completely different craft or vehicle. For instance, an engine temperature gauge in an aircraft may be the same as another completely different aircraft or rotorcraft. This gauge can be developed once for the first visual program and used across both visual programs. Further, the same gauge can span the life cycle of each craft's program, in the courseware, virtual maintenance training, visual simulation, even the physical simulator with virtual panels. In looking at this object oriented reuse approach, one notices a natural fit with object oriented non-proprietary programming languages. A non-proprietary development path is one that is commonly understood by all programmers, software engineers and computer scientists without the need for special training.

In looking at a non-proprietary development system, the developer utilizes skills previously acquired in formal education, such as object oriented C++ or Java or even ANSI C. All are programming standards that are open, portable and non-proprietary. Keeping with these standards, new COTS asset development tools allow for component reuse from simulation, maintenance training, part task training, courseware, or prototyping programs and generate non-proprietary human readable object oriented code from a programming interface design tool.

The e-learning community has been steering away from proprietary web players for a number of reasons. First, a developer must learn a new syntax where not all proprietary players have a well defined API reference guide. Second, and most important, the e-learning community is faced with heavy security restrictions for deploying non-standard content readers. These reasons have pushed developers to search for an industry standard content player and interpreted byte code standards, such as Java, seems to be the safe alternative to the native object code of an ActiveX control for a 3rd party web player. Interpreted byte code is a safe mechanism for the rendering, displaying and delivering interactive components for high fidelity simulation content in e-learning. The removal of proprietary 3rd party web players also supports a broader range of environment integration solutions and reduces migration and compatibility issues when supporting a variety of LMS and LCMS packages.

The last option to consider in the development of e-learning simulation content is the use of cross platform visual content graphics layers. There are two options available for developing graphical content for a visual description language, DirectX, which is solely Windows based, or OpenGL, a cross platform graphics layer. The use of a cross platform graphics layer allows the e-learning environment to function outside of Microsoft Windows and allows for more options in the student station's e-learning environment. For example, students running Firefox on a Linux based workstation would be able to interact with cross platform OpenGL graphics just as they would on a Windows operating system with Internet Explorer. However, content developed in DirectX can only be used through a Windows interface. OpenGL also has the advantage of better utilizing the graphics hardware in a computer by allowing the graphics memory and graphics processing unit (GPU) do the processing of the vertices, texture rendering and polygons handling. This frees up the system's central processing unit (CPU) and system memory for other computational tasks. OpenGL is essentially a graphics standard with all of today's PCs and is operating system independent. JOGL is the official binding for OpenGL in a Java development environment and is currently one of the Java communities most exciting new technologies. This will allow OpenGL to be utilized directly within a web browser without the need for at risk 3rd party proprietary players packaged in ActiveX.

An important option for deployment is the ability to separate and centralize model behavior from the asset. Each HMI object has several different aspects of the behavior it exhibits. There is the innate behavior and the simulation logic behavior. Innate behavior is the natural action of an object. For example a push button is either pushed in or out and may be momentary or a toggle push button and it may be a complex 3D object or a simple 2D texture changing from a visual in state to an out state. The simulation logic of that pushbutton can be a separate aspect of the pushbutton's behavior and dictates how that push button effects it's environment.

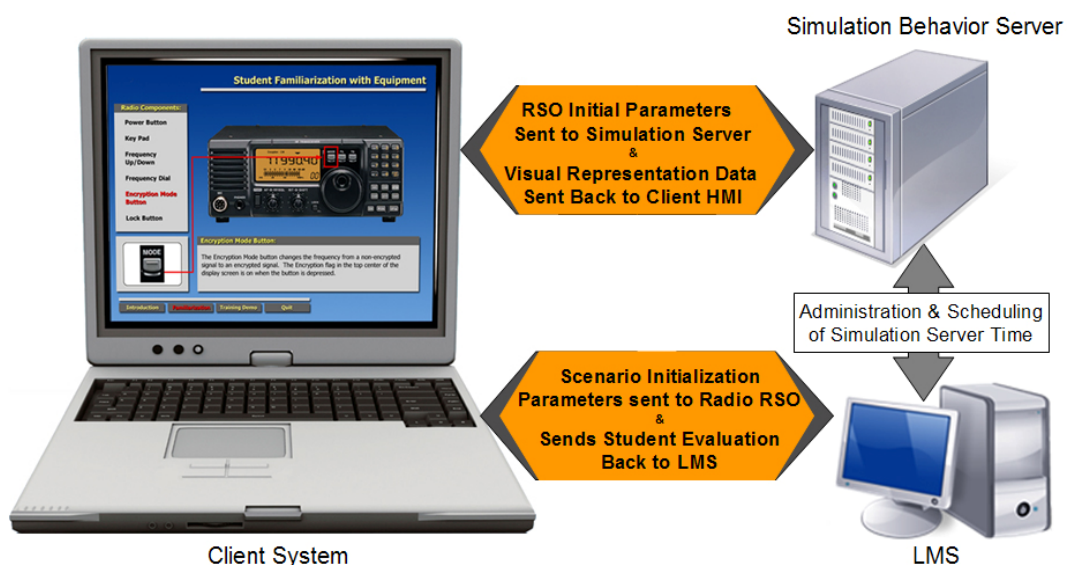


Figure 3 – Simulation Behavior Server Configuration

For instance, when the student pushes the button, that might turn on 1 indicator light or it might effect a dozen other systems in the device. With the RSO methodology, where the visual content and the innate object behavior are contained in the same object, it is simple to house the simulation logic behavior, or model behavior, on a separate server and have the LMS pass a SCO container to the lesson with the innate behavior and the asset, however calling to the model behavior from a separate server. Each RSO knows how to function as an independent object and waits to receive data for a new set of options or rules to govern the environment and what systems the object should interact with in the simulation. This in turn makes the asset much smaller and more manageable allowing the developer to use more of the asset's space for higher fidelity content and faster transportation over limited bandwidth. This also separates the simulation behavior in order to be able to house it on a centrally located server minimizing the need for individual run time licensing on the client browser. The licensing can then be floating from the server and checked out as necessary by the client browser.

Another deployment option to discuss is LMS independent integration. One of the issues that content developers face to day is the migration of data from one LMS to another with completely different requirements and interfaces as discussed in detail earlier. LMS independent integration is the ability to create an asset, or RSO, with a standard safe interface to user defined behaviors in Class Properties & Class Methods. This allows for easy integration in most any environment, independent of the LMS's requirements or interfaces.

Java – A Viable Option

Java is an object oriented programming language that is an industry standard for the delivery of rich 3D interactive web content. Since Java is interpreted byte code, no ActiveX control is necessary for using the Java Runtime Environment (JRE). The Deployment of Java based web content is very secure for even the most stringent government network security policies. There is no code signing required for Java, as there is for ActiveX controls or 3rd party proprietary web players. Though developers can still digitally sign the Java applet content, it is not necessary. The object oriented design nature of Java and the ability for object oriented COTS tools allows for segmented components in the design framework. These segments can be thought of as RSOs and could contain just the visual graphics, just the innate behaviors for objects or a combination of both graphics and behaviors.

Current COTS tools allow developers to segment the model behavior from the visual simulation asset. Because of this separation, developers can continue to use their existing model behaviors with the new RSO front end. Reusable Software Objects allow for leveraging C, C++, Java, or even existing legacy code for model behavior with the Java based visual front end. The architecture can be much more flexible and extensible in this way. Another benefit to this design strategy is the reduction in the size of content assets, allowing for smaller HMI RSO content to be downloaded through less bandwidth. This also allows for faster load times and in some cases, better performance. Newer COTS tools also generate Java code which makes calls to a new technology from Sun called JOGL, the OpenGL bindings for Java. With the introduction of Java with JOGL, performance is even further enhanced via the visual processing offload to the GPU via the use of OpenGL rendering through the OpenGL API (JSR-213).

Conclusion

While historically, the e-learning content community has turned from Java for a content delivery system, the performance risks of using Java have been mitigated by the recent introduction of the Java OpenGL bindings, JOGL. This allows the graphics hardware to process most of the vertices, polygons and textures, as opposed to the system resources doing all the processing.

Java is object oriented, interpreted byte code, allowing for very secure deployment of high fidelity 3D interactive simulation content via the web. Its non-proprietary nature promotes ease of development as an industry standard for object oriented programming. Newer COTS products exist that allow for the easy maintenance, utilization of existing content and easy integration into these web based applications.

The introduction of Java with JOGL allows the e-learning community to explore new avenues for simulation content development. This leads learning object developers into the next generation of

blended learning solutions that allow virtual maintenance trainers, computer based training (CBT 1, 2, 3, and 4) and procedural training to be served from the same e-learning application in a classroom environment or on a portable laptop workstation. By employing the latest technology and methodology discussed in this paper, the developer is able to rapidly and efficiently create high fidelity, interactive, 3D simulation content for use in a wide variety of e-learning solutions.

About The Author

Christopher Giordano has been working in the simulation industry since 1997 and is currently the Product Manager for all DiSTI's COTS software products and the Worldwide Director of Software Support. He holds two degrees from UNC Wilmington (Bachelor of Science Finance) and the University of Central Florida (Bachelor of Science Engineering). At DiSTI, Chris was the Program Manager and Lead Software Engineer for 42 unique visual programs such as the F-35 JSF Courseware, AVCATT Rearchitecture Program, Boeing's F-15K IOS, VELCAC and SLEP LCAC Operator Trainers, HH60/HH65 reconfigurable trainer to name a few. He also teaches the GL Studio training course.

References

"Military Learning Management System Selection Criteria", Joint ADL Co-Labs - Michael Sazma Ed.D.

U.S. Navy Integrated Learning Environment – 10-24-2004 version 1.4

AICC Courseware Delivery Station (www.aicc.org) – v9.1 (ARG 002)

Acknowledgement Statement

All of the trademarks or registered trademarks are the property or their prospective owners or originators. DirectX®, ActiveX®, Windows Vista® and Windows® are trademarks or registered trademarks of Microsoft®. OpenGL® is a registered trademark of the SGI. Java™, the Java Logo and JOGL™ are trademarks or registered trademarks of Sun Microsystems. GL Studio® and Natural Navigation™ is a registered trademark of DiSTI. 3ds Max® is a registered trademark of AutoDesk®, OpenFlight™ (FLT) is a trademark of MultiGen-Paradigm, Inc. Designer's Workbench™ (DWB) is a trademark of Centric Software. S1000D™ is a trademark of the Technical Publication Specification Maintenance Group.