Editors: Lawrence Rosenblum and Michael Macedonia

The Delta3D Open Source Game Engine

Rudy Darken, Perry McDowell, and Erik Johnson

The Modeling, Virtual Environments, and Simulation (Moves) Institute

What is it that all game engines and visual simulation tools have in common? A lot, as it turns out. In fact, game engines have so much in common that you have to wonder if they should actually be a commodity—and they should be. That advanced feature that makes one game engine different from all the rest is part of the reason why game engines and visual simulation tools cost so much. Furthermore, most game engines have a unique development pipeline associated with them. The way content is developed and integrated is specific to that engine, implying limited (if any) portability and reuse. This business model is perfectly appropriate for the entertainment industry where having the latest graphics features can make or break a title, but to the training community, the model simply does not work. We need to think differently.

Why open source?

You might wonder why the military training community is investing in an open source game engine. A common misconception is that they want free software. While "free" is indeed an attractive feature, it is not the driving factor; it's flexibility and reuse. We need reusability on a grand scale because the volume of training applications that will be needed in the coming years is extraordinarily large. Many of these applications will be small compared to a commercial game. It's hard to make a business case to use an expensive game engine for a training application that might involve 15 minutes of training exposure. But the model becomes acceptable when the marginal costs of developing an application are in line with the content and not the architecture—what was once unaffordable is now quite practical and cost efficient.

By embracing open source, we drive investment into content and away from tools. This does not mean that specific tools cannot be built on top of the game engine. Game developers know that it's the tools that actually influence the look and feel of a game more than anything else. By focusing on content and reusability of content and code, we maintain a development pipeline that is open and flexible while maintaining the ability of a developer to have its own unique look and feel.

Why build a new engine?

The next question you might have is, why build a new open source game engine when there are already several open source products available. In a survey of these products, we found that they all tend to be genre specific and none have a particularly large user community. If you compare the size of the user community of any open source game engine with, say, MySQL, Python, or OpenSceneGraph, for example, there is no comparison. Game engine communities are relatively tiny. But what we noticed is that many developers were using multiple lower level tools to build their applications. A common request on the OpenSceneGraph bulletin boards is, Why doesn't someone put OpenSceneGraph and CAL3D together so I don't have to? This is the approach we took in building Delta3D.

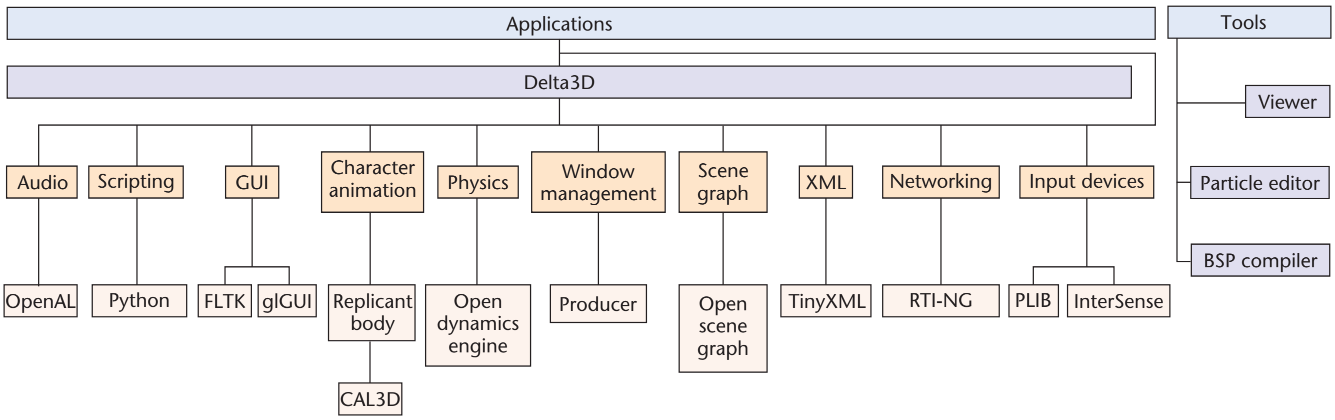
Designing the Delta3D engine

Delta3D is actually a thin, unifying layer that sits atop many open source products you might already use. It has a high-level, cross-platform (Win32 and Linux) C++ API designed with programmers in mind to soften the learning curve, but always makes lower levels of abstraction available to the developer. Programmers can develop content through the level editor—they can write Python script to the Delta3D API or to the underlying tools directly. Delta3D uses the standard Lesser GNU Public License (LGPL). It's completely modular and allows a best-of-breed approach whereby any module can be swapped out if a better option becomes available. Figure 1 shows the Delta3D architecture.

Delta3D also handles networking and has record and playback capabilities for capturing and replaying gameplay. We have a working prototype of distributed rendering via the Common Image Generator Interface for rendering to multiple simultaneous displays. In our laboratory, we have integrated distortion correction for a flight simulator in a CAVE using Delta3D.

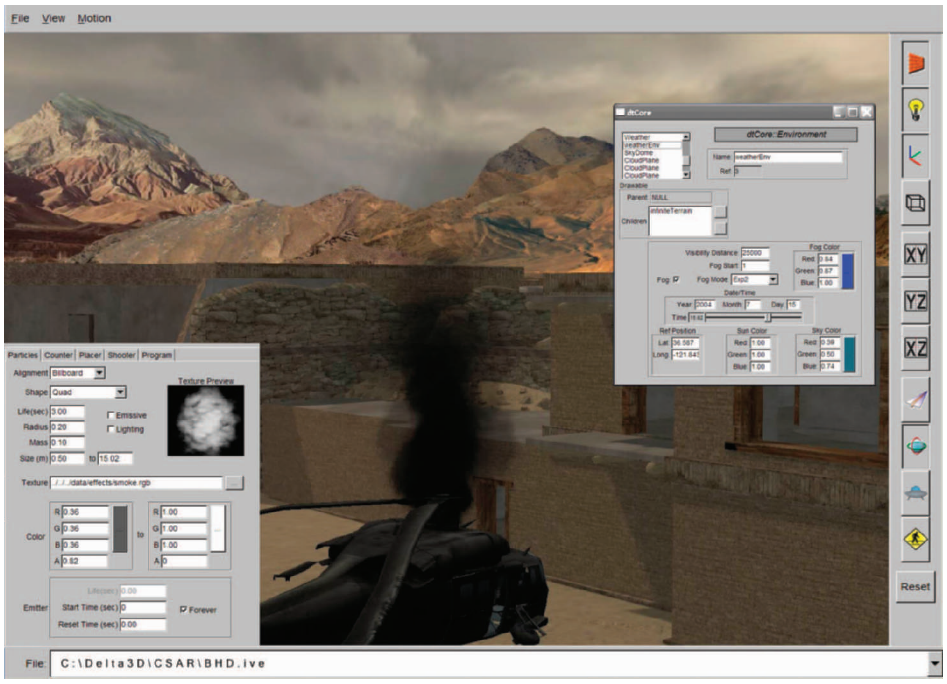
Delta3D has the following high-level tools for rapid application development (see Figure 2): There is an object viewer, a particle editor, a binary space partition (BSP) compiler, and a runtime debug GUI. A full-feature level editor is under development and will be available by the fall of 2005. Delta3D is multigenre. It accomplishes this through an application base class that optimizes for the general requirements of a first-person shooter, real-time strategy game, or whatever is desired.

Delta3D has high-level environmental effects to include dynamic clouds, ephemerides, and weather. It also uses the stateless one-pass adaptive refinement extension (Soarx)[1] algorithm to render continuous levels of detail for terrain such that it effectively renders



1 Delta3D architecture. All the products in the bottom layer are existing open source projects. Delta3D unifies them into one consistent API with associated tools.

infinite resolution by adding noise to the base data set. A current research project underway at the Naval Postgraduate School involves automatically generating and placing vegetation on top of Soarx-rendered terrain. Using readily available data sources, we can generate extraordinarily large terrain models at continuous levels of detail with realistic vegetation that are just as suitable for flight simulation as they are for ground-based games.



2 Delta3D toolset includes a particle editor, an object viewer, and a visual debugger. In the backdrop are models developed for *America's Army* but rendered in Delta3D.

Delta3D community

The developer communities for many of the open source projects within Delta3D are large. Delta3D inherits from all of these. As new features are added to OpenSceneGraph, for example, Delta3D will have them. The same is true for all the open source tools that make up Delta3D. Delta3D developers also contribute back to the open source projects that it uses. We are a part of their communities, and they are a part of ours.

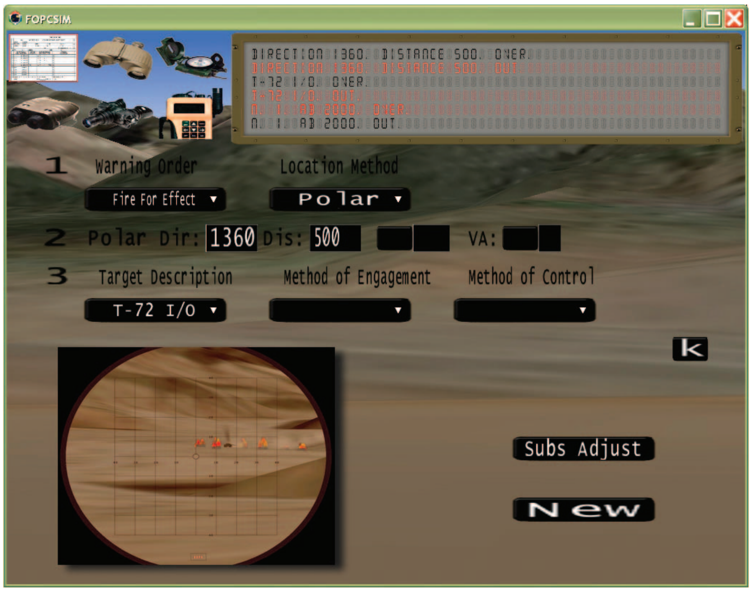
We built a set of tutorials to help new users get Delta3D installed and running properly. These are fairly basic but they get the new developer writing scripts and code and comfortable with the structure of the Delta3D development pipeline. If you know some C++, you'll be writing Delta3D applications in short order.

Beyond tutorials, the API is fully documented and online. The Delta3D Web site (http://www.delta3d.org) contains forums and other assistance from the Delta3D user and developer community. The Web site offers Delta3Dsource code and/or libraries, all the Delta3D dependencies (for example, the open source modules), and example programs and models. The Web site will be the clearinghouse for everything Delta3D was designed to make reusable—from code and scripts to geometric models, textures, and motion capture data.

The sponsors of Delta3D are investing in its development with the intention of building many applications in it in the coming years. This built-in development community will also serve to strengthen Delta3D and support for all developers and users.

Because Delta3D is open source and everything that makes up Delta3D is open source, it's possible (but not necessary) to develop an application entirely on open source. Modeling could be done with the Blender modeling and animation tool (http://www.blender3d.org). Development could be done with a GNU compiler such as GCC and any open source editing tool. Delta3D is suited for all runtime infrastructures and is compatible with the Linux operating system. Game development with no cost for development software and no runtime licensing costs is attainable.

However, we designed Delta3D to coexist with commercial software too. Delta3D commoditizes all the common elements of game engines and simulation tools. Regarding the unique elements, the intent of Delta3D is



```
DIRECTION 1360. DISTANCE 500. OVER.
DIRECTION 1360. DISTANCE 500. OUT.
T-72 I/O. OVER.
T-72 I/O. OUT.
N. 1 AB 2000. OVER.
N. 1 AB 2000. OUT.
```

1 Warning Order: Fire For Effect
Location Method: Polar
2 Polar Dir: 1360 Dis: 500 VA:
3 Target Description: T-72 I/O
Method of Engagement:
Method of Control:

3 **FOPCSIM is intended to have a game feel even though it is a training system. It needs to be easy to install and use by any soldier or marine.**

to enable tool developers to build special-purpose tools that work with Delta3D but are not open source. This conforms to the LGPL license because no modification is made directly to Delta3D. We encourage this approach because it adds value to the engine while preserving the proprietary nature of commercial software.

Applications

Delta3D has already been used for several training game applications. The most mature of these is an application that trains Marine Corps forward observers how to call for and adjust artillery fire. We originally built the forward observer PC simulator (FOPCSIM) on a proprietary visual simulation toolkit (see Figure 3). However, end users requested the software, and even though we were willing to give away the application source code, the runtime licenses associated with the development tools made this impossible. We redesigned and implemented FOPCSIM on Delta3D. It now has users around the US and in four countries.

Another example is a prototype training application for shipboard firefighting. This would be used in a full-scale online learning environment where a sailor might learn how to fight a fire online using a standard Web-based learning management system (LMS) that might include Web pages, images, animations, or other multimedia. Then, to demonstrate proficiency in the task, another application could be integrated with the LMS so that trainers could track the student's progress. This capability is a critical element of the Navy's strategy for future training and education.

What's ahead for Delta3D?

Delta3D is just getting started. We are encouraged by the tremendous response to the project. Downloads and forum activity are on the rise daily. It seems as though many developers were thinking the same things we were in terms of a unification of existing open source tools. Delta3D is a great start, but much remains to be done.

We are developing first-generation development tools to support the Delta3D development pipeline but independent developers will need more. We expect genre-specific tools to emerge soon starting with the application base classes that define genre structure at a high level. The networking capabilities in Delta3D will also need improvement. It handles standard military simulation interoperability and generic socket connectivity, but will need massive multiplayer online gaming capabilities. Developing the LMS connectivity described earlier is also a priority for us in the next year.

We are often asked how support for Delta3D can be made scalable as the community grows. If you buy a commercial product, you can get printed manuals, go to onsite classes, and call a support line with specific questions. Currently, we rely on and participate in the open source community for support. We are actively assisting commercial partners to support Delta3D in the same way that Red Hat supports Linux or MySQL.com supports MySQL.

Delta3D is an idea whose time has come. It's time to commoditize the game engine and visual simulation market by providing a powerful open source tool that allows developers to focus on content rather than the tools themselves. Our clients pay once for an application and have the freedom to distribute the application without licensing restrictions of any kind. Delta3D leverages the success of existing open source tools such as OpenSceneGraph. Why build a new community when the one we need already exists?

We encourage all to download Delta3D and give it a try. There is no panacea to the game development pipeline quandary, but we believe that what is required fits the promise of the open source software movement. As rapidly as the gaming world is changing, it will take a community to keep up with it. Join us in that community. ■

Acknowledgments

Delta3D has been developed through the support of the Navy Modeling and Simulation Management Office, the Naval Education and Training Command, and the Joint National Training Capability Program Office. Terrain generation has been supported by the National Geospatial-Intelligence Agency. The authors also acknowledge their development partners at the Naval Air Command Orlando and BMH Associates.

Reference

1. A. Balogh, *Real-Time Visualization of Detailed Terrain*, master's thesis, Budapest University of Technology and Economics, 2003; http://web.interware.hu/bandi/ranger.html.

Readers may contact Rudy Darken at darken@nps.edu.

Readers may contact the department editors at lrosenbl@nsf.gov or michael_macedonia@peostri.army.mil.