

Soar learning architecture to develop and test human-level artificial intelligence through controlling synthetic characters within the games.

The other two research projects exploit both the modeling and graphics capabilities of the game engines. In the augmented reality game ARQuake, game levels are written to match actual physical locations and players wearing see-through HMDs encounter and do battle with monsters superimposed over real scenery. CaveUT takes advantage of both the graphical and networking capabilities of the Unreal engine by using multiple players' viewpoints to construct a panoramic Cave-like display. There are probably as many potential applications for game engines as there are research problems requiring medium-fidelity 3D simulation or high-fidelity interactive graphics. Our hope is to raise awareness of the high-power/low-cost alternative game engines can offer.

Choosing an Engine

A researcher's choice between the Quake and Unreal engines is similar to that between Coke and Pepsi. Both are more than adequate for their intended purpose yet each has its avid fans and detractors. An examination of contributors here indicates our researchers are evenly divided with Kaminka and Jacobson lining up with the Unreal engine, Bylund and Espinoza choosing Quake III, and John Laird and colleagues working with both.

For research purposes it is hard to imagine an application for which one would be suited and the other not. However, because a choice must be made there are some differences that should be considered.

As the older game, the Quake III engine has developed a large collection of customized levels and modifications to play. Quake III also has a large cadre of users and user sites to use as resources for programming questions and assistance. It is not unlikely that someone in your organization may have Quake programming experience. The Quake engine has been optimized to the point where it is the fastest-running game engine and has what many believe the best overall polygonal architecture.

As the newer engine, Unreal benefits from current trends in programming and technology—see the sidebar “The New Cards.” Unlike Quake, which has descended from a succession of C language implementations, the Unreal engine is strictly object-oriented. This pays big dividends through mutators and other programming constructs that allow a programmer to make robust changes in game behavior without requiring detailed knowledge of the involved code. The Java-like scripting language, Unreal Script, is easy to use and well documented as is the well-designed UnrealEd development environment. Although the Quake engine offers true curves and direct access to graphics functions such as deformation shading, the Unreal

The New Cards

★ Michael Lewis

The most significant recent development in gaming and graphics-intensive computing has been the availability of supercomputer-level graphics on commodity-priced graphics processing units (GPUs). Leapfrogging earlier limits, Nvidia's Quadro2 Pro GPU delivers 31 million polygons per second at a fill rate of over one billion texture-mapped pixels per second, which makes it the world's fastest GPU by some measures. This speed comes despite a raft of new hardware operations including multiple shadings per pixel, multitexturing, texture and lighting, bump mapping and other compute-intensive operations not found until recently in animated PC graphics. The new Nvidia chips are so effective they have found outside applications as the graphical heart of Microsoft's new Xbox game player and at the core of the multifunction avionics display for the F-22 fighter aircraft.

The really big graphics applications like Caves, Power Walls (high-resolution wall-size displays), and

open-canopy flight simulators, however, rely not on a single GPU but on systems built from many graphics pipelines working together in parallel. The top-of-the-line Silicon Graphics Onyx 3800, for example, manages up to 16 simultaneous graphics pipelines for an upper limit of 210 million polygons per second and a fill rate of 12 billion pixels per second. Achieving coherent graphics on this scale requires elaborate coordination and management to rebalance loads among pipelines at the crucial transition in processing between object parallelism and image parallelism. The SGI InfiniteReality architecture designs graphics pipelines to accept broadcast primitives at just this point. General-purpose GPUs designed for PC graphics and embedded applications do not have the luxury of this “sort-middle architecture” because they are designed to work as self-contained standalone units.

Making the transition from competing with desktop workstations to taking on refrigerator-sized, rack-mounted capital investments requires solving the problem of parallel processing using commodity GPUs.

engine encapsulates a broader range of sophisticated graphics including bump maps and other tweaks recently added to graphics hardware. In the balance, the Unreal engine is slightly slower, has more sophisticated graphics, and is probably an easier environment for the inexperienced game programmer.

Getting Started

Now that you have chosen your engine all you need to get started is a copy of the base game. Depending on where you shop the game may cost anywhere from \$20–\$50. You will need a license for each machine you use so, for instance, a QuakeSim simulation would take a single license, a GameBot installation with two teams would require two licenses, while a five-screen CaveUT installation would require six licenses. A crucial feature of the game engines is that almost all code, except the proprietary graphics engine, is open source. Under its GNU license, this code may be freely distributed, copied, and modified. Several of the systems discussed here are already available online. CaveUT can be downloaded from www2.sis.pitt.edu/~jacobson/ut/-CaveUT.html and GameBot code, documentation, discussion, and even opponents, are located at www.planetunreal.com/gamebots/. While not game-based, the WireGL code for building PC-based Power Walls is available at graphics.stanford.edu/software/wiregl/. The primary Quake site

is www.idsoftware.com, while www.averstar.com/~bowditch/QUAKE2/links.html contains links to many secondary Quake sites. The manufacturer's site for Unreal Tournament is www.epicgames.com; a "getting started" FAQ is available at www.unreal.org/Cleaned/FAQ/FaqMain.htm. A quick search will turn up a multitude of sites with information, tools, and programming tips for either of these engines. While neither id Software nor Epic Games is in the business of supporting research, their user communities can provide active sources of help and information for game-using researchers. ■

REFERENCES

1. Abrash, M. Quake's game engine: The big picture. *Dr. Dobbs' Journal* (Spring, 1997).
2. Bishop, L., Eberly, D., Whitted, T., Finch, M., and Shantz, M. Designing a PC game engine. *IEEE Computer Graphics and Applications* (1998), 46–53.
3. Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsuura, H., Noda, I., and Asada, M. The RoboCup synthetic agent challenge '97. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Nagoya, Japan, (1997).

MICHAEL LEWIS (ml@sis.pitt.edu) is an associate professor in the Department of Information Science and Telecommunications at the University of Pittsburgh.

JEFFREY JACOBSON (jacobson@sis.pitt.edu) is a Ph.D. candidate in the Department of Information Science and Telecommunications at the University of Pittsburgh.

© 2002 ACM 0002-0782/02/0100 \$5.00

Because the graphics pipelines themselves are inaccessible and closely tied to other parts of the PC architecture, the most convenient approach lies in clustering PCs to achieve "sort-first" machine-wise parallelization. While this architecture is limited by network bandwidth and is less efficient than the "sort-middle" approach of high-end graphics vendors, the huge cost savings and the speed advantage of commodity GPUs makes it hard to resist.

We are aware of two such solutions. WireGL (now Chromium),¹ developed at Stanford University, is a general-purpose system for scalable interactive rendering on a cluster of workstations. WireGL replaces the OpenGL libraries, which allows OpenGL calls to be parallelized and rendered through the cluster. Reported performance of 70 million triangles per second for a 32-PC configuration (16 compute and 16 rendering nodes) compares well with the achievable performance esti-

mate of 29 million polygons per second for a 16-pipeline SGI InfiniteReality configuration. Using PCs and data projectors found in most labs, WireGL can allow researchers to experiment with previously rare and expensive large-scale high-resolution displays.

The same hardware can be used to run CaveUT (described in this issue by Jacobson and Hwang) to create an immersive panoramic Cave-like display. CaveUT follows the same cluster rendering approach as WireGL but takes advantage of the Unreal engine's networking and graphics synchronization capabilities to do the processing. Because game players' views must be computed independently by clients, the opportunity for load balancing is lost. Nevertheless, given lightweight communication protocols, the power of the new commodity GPUs and gigahertz-plus GPUs, display refresh rates have become the limiting factor. ■

MICHAEL LEWIS (ml@sis.pitt.edu) is an associate professor in the Department of Information Science and Telecommunications at the University of Pittsburgh.

¹Humphreys, G., Eldridge, M., Buck, I., Stoll, G., Everett, M., and Hanrahan, P. WireGL: A scalable graphics system for clusters. In *Proceedings of the 2001 Conference on Computer Graphics* (Los Angeles, CA, 2001), 129–140.