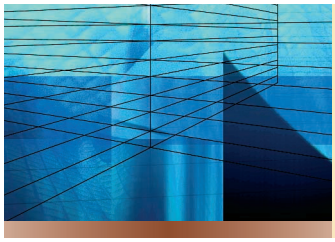
The Panda3D Graphics Engine

Mike Goslin and Mark R. Mine, *Walt Disney Internet Group*

Disney's new engine has already proven useful for internal and external projects.



Disney's VR Studio developed Panda3D, a graphics engine and programming environment, to be flexible enough to support everything from real-time graphics applications to the development of high-end virtual reality theme park attractions or video games. The acronym itself lists Panda's primary features as a platform-agnostic, networked, display architecture.

As a result of its powerful and flexible design, much of the Panda system's architecture reflects the occasionally opposing goals of performance and versatility. A full-featured engine, Panda includes the following subsystems:

- particle systems,
- physics and collisions,
- soft- and hard-skin character animation,
- 2D and 3D nonuniform rational B-spline support,
- 2D graphical user interface support,
- multipass rendering, and
- a real-time shader framework.

Since its release as an open source software project in 1999, Panda has served as the platform for many internal theme park design projects as well as Disney's *Toontown Online* (www.toontown.com), released for the PC in 2002.

PLATFORM-AGNOSTIC

Panda makes extensive use of abstraction layers to facilitate portability. The system represents services such as rendering, audio, peripheral devices, graphical user interfaces, and scripting languages in an abstract form so that it can use different platforms and external software packages interchangeably.

How Panda employs a scene graph for rendering provides a good example of abstraction layer use. All code specific to rendering on a particular platform is contained within a well-defined class called a *graphics state guardian*. After the system transforms and culls the scene graph, it hands off the graphics entities to the GSG for rendering. A game or application only needs to interact with the scene graph, which means the only part of the code that the system must port and optimize for a particular hardware platform is the local version of the GSG class itself.

Panda currently runs on a wide array of graphics cards for all flavors of Windows PCs, Linux PCs, and Silicon Graphics workstations running Irix. The engine offers rendering support for DirectX, OpenGL, and Pixar's Renderman (S. Upsill, *The RenderMan Companion*, Addison-Wesley, 1990). In addition, our long-term plans include providing support for game consoles and Macs running OS X.

NETWORKED

We designed Panda to support a range of networked applications including multiuser design scenarios, multiplayer games, and multiple synchronized displays powered by multiple PCs. In addition, developers can use Panda to deliver downloadable experiences over the Internet.

Designers thoroughly leverage Panda's networked features in the development of massively multiplayer online games. In *Toontown Online*, for example, Panda's small memory footprint makes it possible to support the online delivery of large content. As Figure 1 shows, an application can begin running and then continue downloading content in the background as needed. In addition, the in-game requirements for a massively multiplayer game include built-in concepts of distributed objects, visibility, and efficient network communications.

DISPLAY ARCHITECTURE

Panda can drive a range of displays from low-end PCs up to high-end multiscreen immersive theater environments. Configuring applications that require multiple windows, even if they require synchronization across hardware platforms, is relatively easy.

The ability to specify the state of nodes internal to the scene graph means that an object's state depends on an ordered accumulation of all the states in that object's ancestor nodes, up to the scene graph's root. To avoid having to aggregate this state every

rendering frame, Panda includes an efficient caching mechanism. The system initially determines an object's properties by doing a full traversal of the scene graph, caching the results, and storing the cumulative results at each node. After doing this, the engine only recomputes a node's properties when an intervening state changes.

Panda supports several methods of dynamically scaling content to take advantage of different hardware capabilities. The system implements several visibility algorithms to eliminate unnecessary rendering, including cell portal visibility (D.P. Luebke and C. Georges, "Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets," *Proc. 1995 Symp. Interactive 3-D Graphics*, ACM Press, 1995, pp. 105-106) and another cell visibility algorithm based on adjacency. Panda also supports fading between levels of detail. As a result, the same Panda application can look good on a high-end graphics workstation and yet still run at high frame rates on an older PC.

Development is also underway on a distributed rendering system for Panda, Pandamonium, which developers will use for multidisplay rendering on PC graphics clusters. Similar in spirit to WireGL (G. Humphreys et al., "WireGL: A Scalable Graphics System for Clusters," *Proc. ACM Siggraph 2001*, ACM Press, pp. 129-140), Chromium (G. Humphreys et al., "Chromium: A Stream Processing Framework for Interactive Graphics on Clusters," *Proc. ACM Siggraph 2002*, ACM Press, pp. 693-712), and Blue-c (M. Gross et al., "Blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence," *Proc. ACM Siggraph 2003*, ACM Press, 2003, pp. 819-827), Pandamonium distributes low-level scene graph changes across a network.

KEY FEATURES

In addition to the attributes that Panda's name represents, several other prominent features deserve mention.

***Figure 1. Online content delivery. Panda's small memory footprint enables background downloading of content for graphics-intensive applications such as Disney's* Toontown Online.**

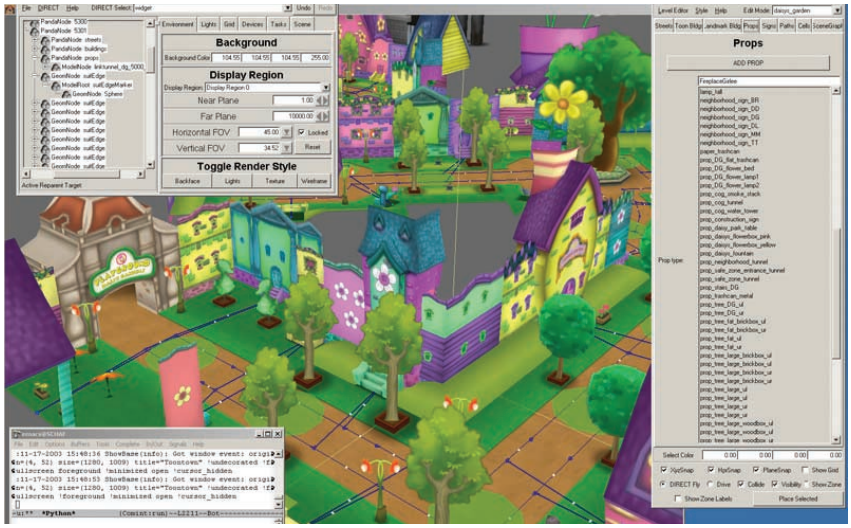


Figure 2. Interface generation. Panda can generate 2D GUIs using Python's Tkinter/Pmw wrappers around the Tcl/Tk API.

Standard tools

Wherever possible, Panda uses standard programming tools and techniques. This decreases the learning curve for new Panda programmers and makes the engine more adaptable to advances in compilers, graphics, and other related components. We wrote much of Panda in C++ and ensured that it takes advantage of the Standard Template Library. As Figure 2 shows, developers can use Python's Tkinter/Pmw wrappers around the Tcl/Tk API to generate graphical user interfaces. They can use the Cg high-level shader language to create advanced rendering effects.

To integrate with the variety of input devices that virtual reality applications require, Panda supports the Virtual Reality Peripheral Network API (www.cs.unc.edu/research/vrpn). Developed at the University of North Carolina at Chapel Hill, VRPN has become a de facto standard for the tracker and input device drivers common to interactive simulations.

Panda also facilitates flexible content creation. It can import models and animations from common programs such as Maya, MultiGen, 3Dstudio Max, Softimage, and Lightwave. The engine also supports most common image and sound file formats.

Open source

Panda benefits from having a thriving open source community for documentation, testing, and development of new features. Its performance characteristics and free availability as an open source software project position Panda as a potential graphics engine standard for simulation and game development. Since 2002, Carnegie Mellon University's Entertainment and Technology Center has become a central repository for information related to Panda and the Panda community (http://etc.cmu.edu/panda3d).

Rapid prototyping

Panda uses the Python scripting language (www.python.org) as an interface to C++ libraries. Developers can use Python, an interpreted language, to make changes to their code without recompiling. This allows for rapid prototyping, which can shorten development time and improve quality—making Panda particularly useful for research projects and mockups during game development. Because Python also resembles C, most programmers will recognize familiar rules and structures and can get started relatively quickly.

With the built-in Interrogate system, Panda can be used with scripting languages other than Python. This system works with any interpreted language that has a foreign function interface that allows calls to C libraries. Interrogate works like a compiler, scanning and parsing C++ code. Instead of creating object libraries, however, it creates a database of entities such as objects, methods, and global variables that are contained in the corresponding C++ library.

The system can then query this database to discover the functional elements and their interfaces. To use the database, the developer creates an automatic code generator in the scripting language. The generator scans the Interrogate database and generates scripting-language wrapper calls that execute the C library calls via the scripting language's foreign function interface. Interrogate imposes no restrictions on exactly how a scripting language interfaces with the C libraries; the libraries can interface in whatever way best fits their language's natural environment.

Developers have already used Panda to release and support multiple professional, production-quality applications (http://sourceforge.net/projects/panda3d/). They have used the engine to design both theme parks and theme park attractions, as the platform for a variety of research and development projects, and as the engine for the successful massively multiplayer online Toontown game. Panda is currently being used for several research and game projects, both inside and outside Disney. ■

Mike Goslin and Mark R. Mine are developers at the Walt Disney Internet Group, VR Studio. Contact Goslin at Mike.Goslin@disney.com and Mine at Mark.Mine@disney.com.

Editor: Michael Macedonia, Georgia Tech Research Institute, Atlanta; macedonia@computer.org