gd

GAME DEVELOPER MAGAZINE

DECEMBER 2002

Game Developer
Exit, Stage Left
A Postmortem Of Presto Studios
Saving Face
Choosing The Right Facial Animation System

It's Just the Way Things Are

It's been over five years since the first Postmortem appeared in *Game Developer*. Combining equal parts vicarious wisdom and Schadenfreude, the brainchild of former editor Alex Dunne hit on the same magic formula that has made ABC's *Wide World of Sports* a favorite for more than 40 years: the thrill of victory, and the agony of defeat.

Since that first Postmortem of DARK SUN ONLINE: CRIMSON SANDS in October 1997, the 61 Postmortems that have appeared in every regular issue of *Game Developer* have had one thing in common (besides scheduling gaffes): they were about the development of a game. This month we break with *Game Developer*'s longest-standing tradition by bringing you the first Postmortem of a recently defunct game studio.

Oh sure, game studios come and go all the time, but the end of Presto Studios in particular kept me thinking about the current and future state of game development for small independents. Just as several years ago we saw some developers fail to make the transition from one-to-three-person teams to teams of 20 or more, increasingly I worry about the fate of many of today's boutique development houses as the industry moves toward consolidation and teams continue to grow and change their dynamics. Multiple projects and team sizes of 50-plus require a whole new level of management savvy above and beyond what's been passable in years past. (And the past 61 Postmortems suggest that "passable" is sometimes too generous a description for game development management practices.)

The model of a game studio started up by a committed group of friends or family members isn't just a familiar story in game development, it's one of the oldest models of enterprise. Game developers being the precocious lot they are often start up studios before they're out of school. With any measure of success, a few years later these entrepreneurs find themselves leading companies of dozens, even hundreds of people. How can you learn to manage so many people when you've never actually worked for anyone else? How do you transition smoothly from a startup where everyone puts in long hours because their fortunes are inextricably intertwined with the company's, to an organization where employees are primarily concerned with supporting their families at home and have little regard for the fortunes of the owners?

I am optimistic that more experienced managers, broad economic success, ever-improving tools, and a growing pool of diverse talent will conspire to make game development processes less about reinventing the wheel and more about replicating past success. The craft is a work in progress, which is why game development continues to be such a captivating field to those who commit themselves to it. The trick is for development managers to be able to discern opportunities for change and then implement them successfully across a team.

My hat's off to Presto for recognizing what they felt was a good time to make a graceful exit. Don't give in to the lazy rationale that a studio that made its name with CD-ROM graphic adventures had its days numbered. If you read Michael Saladino's article starting on page 44 and don't see something of your own company in there, you're either not looking hard enough or you have the good fortune to be working for the best-run game studio out there (or perhaps you're managing it yourself!). The industry's continued success raises the stakes every year, making even the small mistakes more costly, sometimes irrevocably so. Those who explain away their company's chronically broken processes by saying, "That's just the way things are," may eventually find themselves, like Presto, saying, "That's the way things were."

Jennifer

Jennifer Olsen
Editor-In-Chief

GameDeveloper

600 Harrison Street, San Francisco, CA 94107 t: 415.947.6000 f: 415.947.6090

Publisher
Jennifer Pahlka *jpahlka@cmp.com*

EDITORIAL

Editor-In-Chief
Jennifer Olsen *jolsen@cmp.com*

Managing Editor
Everard Strong *estrong@cmp.com*

Production Editor
Olga Zundel *ozundel@cmp.com*

Product Review Editor
Daniel Huebner *dan@gamasutra.com*

Art Director
Audrey Welch *awelch@cmp.com*

Editor-At-Large
Chris Hecker *checker@d6.com*

Contributing Editors
Jonathan Blow *jon@number-none.com*
Hayden Duvall *hayden@confounding-factor.com*
Noah Falstein *noah@theinspiracy.com*

Advisory Board
Hal Barwood LucasArts
Ellen Guon Beeman Monolith
Andy Gavin Naughty Dog
Joby Otero Luxoflux
Dave Pottinger Ensemble Studios
George Sanger Big Fat Inc.
Harvey Smith Ion Storm
Paul Steed WildTangent

ADVERTISING SALES

Director of Sales/Associate Publisher
Michele Sweeney e: *msweeney@cmp.com* t: 415.947.6217

Senior Account Manager, Eastern Region & Europe
Afton Thatcher e: *athatcher@cmp.com* t: 415.947.6224

Account Manager, Northern California & Southeast
Susan Kirby e: *skirby@cmp.com* t: 415.947.6226

Account Manager, Recruitment
Raelene Maiben e: *rmaiben@cmp.com* t: 415.947.6225

Account Manager, Western Region & Asia
Craig Perreault e: *cperreault@cmp.com* t: 415.947.6223

Account Representative
Aaron Murawski e: *amurawski@cmp.com* t: 415.947.6227

ADVERTISING PRODUCTION

Vice President, Manufacturing Bill Amstutz
Advertising Production Coordinator Kevin Chanel
Reprints Cindy Zauss t: 909.698.1780

GAMA NETWORK MARKETING

Director of Marketing Greg Kerwin
Senior MarCom Manager Jennifer McLean
Marketing Coordinator Scott Lyon

CIRCULATION

BPA INTERNATIONAL
Game Developer is BPA approved

Group Circulation Director Catherine Flynn
Circulation Manager Ron Escobar
Circulation Assistant Ian Hay
Newsstand Analyst Pam Santoro

SUBSCRIPTION SERVICES

For information, order questions, and address changes
t: 800.250.2429 or 847.647.5928 f: 847.647.5972
e: *gamedeveloper@halldata.com*

INTERNATIONAL LICENSING INFORMATION

Mario Salinas
t: 650.513.4234 f: 650.513.4482 e: *msalinas@cmp.com*

CMP MEDIA MANAGEMENT

President & CEO Gary Marshall
Executive Vice President & CFO John Day
Chief Operating Officer Steve Weitzner
Chief Information Officer Mike Mikos
President, Technology Solutions Group Robert Faletra
President, Healthcare Group Vicki Masseria
President, Electronics Group Jeff Patterson
President, Specialized Technologies Group Regina Starr Ridley
Senior Vice President, Global Sales & Marketing Bill Howard
Senior Vice President, HR & Communications Leah Landro
Vice President & General Counsel Sandra Grayson
Vice President, Creative Technologies Philip Chapnick

CMP United Business Media

GamaNetwork

WWW.GAMANETWORK.COM

Return

INDUSTRY WATCH

KEEPING AN EYE ON THE GAME BIZ | *everard strong*

Mr. Brown takes on consoles and PCs. Black Label Games, which Vivendi Universal started up in August, announced they will be working with film director Quentin Tarantino in bringing an interactive version of *Kill Bill*, his soon-to-be-released movie, to consoles and the PC. The movie, starring Uma Thurman and Lucy Liu, is the tale of a former assassin betrayed by her boss. This will be Tarantino's first foray into games. The game is scheduled for a year-end 2004 release.

Vivendi keeps growing. Vivendi Universal Games has acquired Swedish interactive development studio Massive Entertainment, makers of GROUND CONTROL.

VR1 and Jaleco become one. Pacific Century CyberWorks Japan recently acquired both game developer VR1 and publisher Jaleco USA, merging the two units into a new company, Jaleco Entertainment. The company, headquartered in Buffalo, N.Y., includes 11 internal teams based in Europe and the United States. Targeting the console market, the company plans to publish nine titles by the end of this year, including TRAILER PARK TYCOON and FIGHTER ACE 3.5, on six different platforms.

Uma Thurman's *Kill Bill* character will soon be slicing and dicing her way across consoles and PC monitors.

Move over Tivo. Sony Computer Entertainment has revealed that it is close to bringing a kit to market that will enable the PS2 to record television on a hard disk, similar to a Tivo recorder. The timing and details of the product launch have not been decided, but the product will target the Japanese market first.

What a nickel buys these days. Interplay was delisted from Nasdaq's SmallCap Market in October and was trading at 5 cents a share at press time. The company's overcast skies have been darkening since Nasdaq demoted the company from Nasdaq's market to their SmallCap Market last May.

Michael Crichton and Sega team up. Known for science fiction best-sellers *Jurassic Park* and *The Andromeda Strain*, author Michael Crichton announced he will be working with Sega in developing an original videogame, not based on any of his novels or characters. Crichton's previous efforts in game development include founding Timeline Studios and then closing it after releasing a single title.

CEO bails out 3DO (again). Trip Hawkins, 3DO's CEO, once again reached for his checkbook, this time offering $3 million of his money to help the company secure a $15 million loan from GE Capital.

The silver screen is doomed. According to id Software, Warner Brothers Pictures is in final negotiatons to bring the first-person shooter DOOM to the big screen within the next 15 months. The movie's story line will most likely follow the third installment of the DOOM series, set in the future at a paramilitary base on Mars.

Send all industry and product news to news@gdmag.com.

THE TOOLBOX

DEVELOPMENT SOFTWARE, HARDWARE, AND OTHER STUFF

Intel releases free light-field mapping toolkit. Intel recently released OpenLF, an open-source light-field mapping (LFM) toolkit. The tool lets developers build 3D applications that model the way light reflects off of real objects and surfaces. Developers can use the code freely as-is or modify it for use in their applications. www.intel.com

NXN launches Alienbrain 6. With a revamped interface and pricing scheme, Alienbrain 6.0 is a thorough overhaul of NXN's asset and configuration management system. New tools include full branching, branch merging, sharing, and full integration with Microsoft Visual Studio and Metrowerks' CodeWarrior, plus a license for Araxis Merge Professional. www.alienbrain.com

Factor 5 teams up with DivX. In conjunction with DivX, game developer Factor 5 recently released a DivX for Gamecube SDK, offering DivX video compression with Gamecube-optimized assembler code, the ability to map video on any surface and run in parallel with other applications, integration with Factor 5's MusyX and AX sound libraries, and more. www.factor5.com

UPCOMING EVENTS CALENDAR

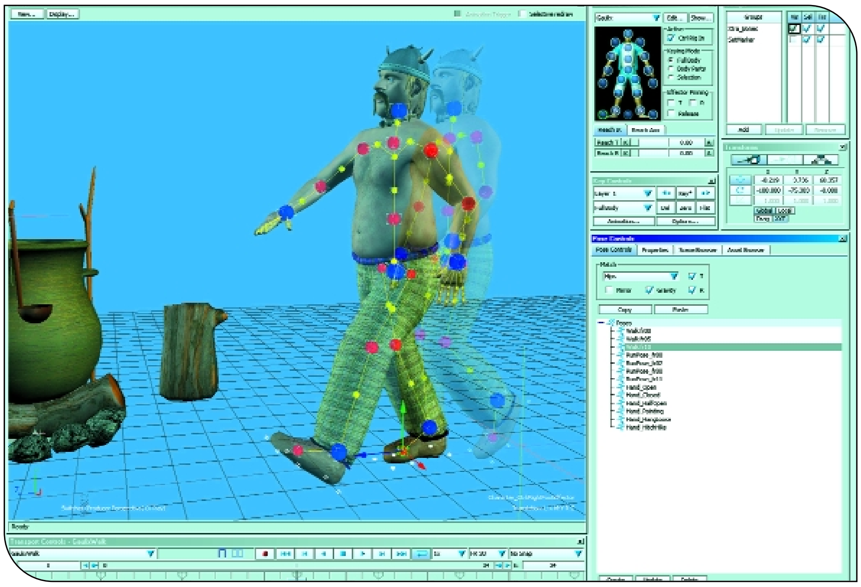
SECOND INTERNATIONAL CONFERENCE ON APPLICATION AND DEVELOPMENT OF COMPUTER GAMES
HKSAR
Hong Kong, China
January 6–7, 2003
Cost: variable
www.adcog.org

MILIA
PALAIS DES FESTIVALS
Cannes, France
February 4–6, 2003
Cost: variable
www.milia.com

Kaydara's Motionbuilder 4.0

by tom carroll

The first version of Kaydara's Motionbuilder (4.0, that is) could be considered heir apparent to the popular Filmbox 3.5 package — but should it? Only FBI recruits, atomic scientists, and Navy SEALs require more training than Filmbox users. After taking Motionbuilder 4.0 for a small test run, it's safe to say that it is much more intuitive. The package's user interface, workflow, and feature set are all different from previous versions of Filmbox. In fact, Motionbuilder 4.0 incorporates drag-and-drop functionality that would have probably been anathema to Kaydara's earlier offerings.


With new actor and character controls, Motionbuilder 4.0 offers animators a powerful program.

What it is

Motionbuilder 4.0 is a real-time animation system that lets anyone create character animation quickly, without having to wait for rendering to see the results. The package contains dedicated tools for creating cameras, lights, shading, cel shading, textures, shadows, and various scene constraints. At its heart, though, Motionbuilder 4.0 is all about 3D animation: keyframing, organizing and preparing motion capture sequences, and blending animation sequences of various types together.

It features an intuitive drag-and-drop approach to content management and importers for many file formats. Leading 3D content creation packages have embraced Kaydara's own .FBX file standard, making it even easier to transport content and data from package to package.

For this reason, Motionbuilder is a viable choice for game developers who need a single package to accomplish all their 3D animation needs, both character and facial; using a single package will help improve and streamline pipeline issues.

Some of Motionbuilder 4.0's new features (and benefits over Filmbox 3.5) include floating windows, an asset browser, a navigator window, a scene browser, and new actor and character controls.

While floating windows are nothing new, they are new to Motionbuilder, increasing the package's functionality by making it more acceptable to a wider variety of users. There are four predefined layouts for creating, animating, editing, and previewing, but the user can customize the package to his or her own tastes or to conform to the requirements of a specific project.

The asset browser and scene browser are very powerful tools. The asset browser lets you select assets, such as lights and cameras, from a pool of templates or from your own added shortcuts. Assets display in a tree hierarchy, and you can use different layouts to view the content

TOM CARROLL | *Tom broke into videogame graphics in 1983 at Cinematronics (yes, the warehouse was overflowing with dead laserdisc players and SPACE ACE cabinets). Currently, he is lead level designer at Vision Scape Interactive. You can reach him at tcarroll@vision-scape.com.*

★★★★★ excellent
★★★★ very good
★★★ average
★★ disappointing
★ don't bother

within the folders. The scene browser lists every asset in the scene, showing their composition and any ongoing relationships between them; it also enables the user to control display of assets by using filters.

The actor and character controls window displays options relative to a selected actor or character. Some features that were previously in the actor settings are now in the actor controls window, letting you manipulate and adjust your actor. The character controls window includes a new character animation system that lets you work with control rigs on characters. You can create your own character animation from scratch or modify your motion capture data using control rigs.

KAYDARA MOTIONBUILDER 4.0 ★★★★

STATS
KAYDARA
Montréal, Québec, Canada
(514) 842-8446
www.kaydara.com
PRICE
$ 3,495 (MSRP)
SYSTEM REQUIREMENTS
Intel Pentium processor with Windows 2000 SP2 or Windows XP, Red Hat Linux 7.2 or higher, Mac OS X 10.1 or higher; 256MB RAM; 300MB disk space; OpenGL graphics card (16MB RAM)

PROS
1. Complete package lets users go as far as they want.
2. Widespread acceptance of Kaydara's .FBX file format.
3. Any Filmbox user on active maintenance gets free upgrade.

CONS
1. Newbies may be overwhelmed by program's depth and complexity.
2. Occassional problems in importing data from other 3D packages.
3. Known release bugs (though workarounds are often provided).

Working with Motion

Motionbuilder 4.0 begins with a relatively simple and recognizable interface: windows, listers, pull-down menus — nothing new or different. The overall look is reminiscent of various popular 3D packages, most strongly Lightwave, with a clean layout that allows users to work without too much clutter. The Motionbuilder menu bar lets you select commands to perform various operations, including File, Edit, Model, Animation, Media, Settings, Window, Layout, and Help. You can also use keyboard shortcuts instead of menu commands.

Because of the flexibility of the .FBX file standard and the package's new drag-and-drop functionality, workflow with Motionbuilder was relatively simple. I used Maya to create and bone a simple character. Using Motionbuilder's drag-and-drop menus, I imported the character into a scene and used the package's character-rigging tools to rig my character quickly. Soon I was animating with it, and I'm convinced this happened much more quickly than if I had stayed within Maya to do so. I was pleased, to say the least.

Unfortunately, I wasn't as universally happy with Motionbuilder's ability to import every animation without some flaws. While most of my scenes came through acceptably, there were a couple of occasions where some data was munged, as happened with a 3DS Max file from one of my previous games. The fault may lie with my own inexperience with Motionbuilder (and I've never known a 3D package that was completely accurate in importing and exporting to every other package — it's 3D animation's Achilles heel). In fact, Kaydara's engineers readily admitted that there might be times when the package's limitations here might need to be smoothed by a fix or workaround, especially when establishing the asset pipeline for a new project. Kaydara maintains a staff of problem solvers that stands by to assist development teams with similar problems (and even ones that might make my little glitches seem insignificant).

Motionbuilder 's flexible nature doesn't negate the need for various kinds of help. While the demonstration package evaluated for this review came without documentation, the online help system seemed extremely complete and relatively easy to understand and use. Also, the company provides complete release notes and valuable online tutorials from its web site, www.kaydara.com.

Bottom Line

Motionbuilder 4.0 is a terrific package, especially for users importing assets into the package and then marrying them up with animation data. It delivered everything I asked of it: importing characters, creating scenes with lights and cameras, automatic character rigging, motion blending, and retargeting animation data from one character to another. The functionality I tested leads me to believe Kaydara isn't trying to fudge the consumer in other areas. This includes facial animation, something that can be tough to get started with in other packages.

There's another, much smaller problem that Motionbuilder has, and it's not about performance but rather economics. Motionbuilder creates a requirement for animators who are skilled in its use, and $3,495 is steep for even established animators to fork out on their own. While this isn't the problem for Kaydara that Filmbox was — again, Motionbuilder is considerably more powerful and intuitive — it's a challenge that Kaydara is confronting with its new offering.

CEBAS'S GHOSTPAINTER

by sergio rosas

GhostPainter is a great tool for game developers that use Photoshop and 3DS Max. The program links a material's texture map in Max with the current file in Photoshop and becomes a simple but powerful 3D paint program.

Once installed, GhostPainter becomes an option in any 3DS Max map slot including bump, displacement, or opaci-

★★★★★	excellent
★★★★	very good
★★★	average
★★	disappointing
★	don't bother

ty. Setting up a 3D painting environment is easy: run Max and Photoshop side by side and apply GhostPainter to a material. That's all there is to it. With GhostPainter's straightforward interface, there is no learning curve.

In paint mode, GhostPainter projects a real-time line onto your 3D object representing the stroke to be applied in Photoshop. As soon as you release the mouse button or stylus, GhostPainter applies the stroke in Photoshop and updates the texture on your 3D model. Since all image operations are done right in Photoshop, you have all the power of Photoshop's tools: brushes, layers, actions, and so on. You can also draw on the map in Photoshop and see your changes in Max.

Although it's not quite a full-fledged, real-time 3D paint program, GhostPainter's simple concept of combining two very powerful graphics packages makes it a great alternative.

GhostPainter works in 3DS Max versions 3, 4, or 5, and 3DS VIZ in conjunction with Photoshop 5 or later. The program retails for $150 and is distributed only through an American partner, Trinity 3D (www.trinity3d.com).

★★★½ | GhostPainter
CEBAS | www.trinity3d.com

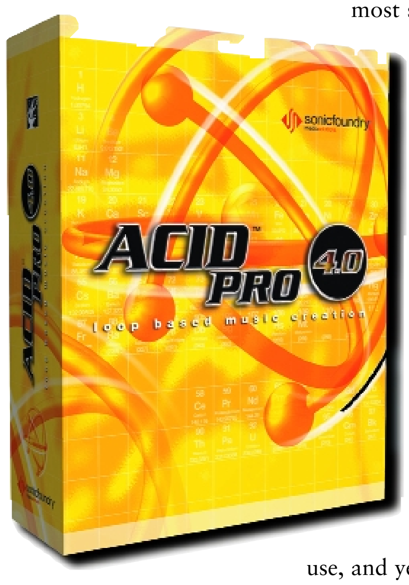
Sergio Rosas is an art director at Ion Storm in Austin, Tex.

SONIC FOUNDRY'S ACID PRO 4.0

by gene porfido

This is not your momma's Acid Pro. The last time I took a good look at Acid was version 2.x, and while the looping creativity was inspiring, I felt it lacked some of the strong points that full MIDI/audio sequencers have. Version 4.0, however, is so packed with features that it's hard to ignore. The incredible loop flexibility that has made Acid famous is back with a vengeance, and there are a host of new features that make Version 4.0 exciting and intense as an audio tool, regardless of what project you're working on.

Acid Pro 4.0 comes packed with hundreds of loops and material to use as a starting point or an addition to your own projects. The backbone of version 4.0 includes an easy-to-learn interface, real-time tempo and pitch matching, unlimited audio and MIDI tracks (limited only by your computer's capabilities), video scoring, 5.1 surround mixing, DirectX effects, support for VST instruments, tempo and key mapping, locking envelopes for effects volume and pan, a wide range of file format support, 16- and 24-bit audio, and the ability to read/generate MIDI time code. But wait, there's more.



Important new features include plug-in automation, ASIO support, MIDI piano roll and step recording, loop cloning, Master, Aux, and Effects "bus" tracks, autosave for crash recovery (this has proved itself useful already), and Windows Media import. Stack all of these atop Acid's already capable feature set and you have a program with more bells and whistles than nearly anything out there. However, all the bells and whistles in the world do not a perfect program make. With all of its good points there's still room to improve.

Installing Acid Pro was a snap, as was loading and playing a project file. Loops rule in Acid, and I'm still impressed with the speed with which the program changes tempo while maintaining pitch across multiple tracks of sampled audio. This feature alone may be worth the price of admission. While the Piano Roll editor is a welcome addition and shows that Acid is getting serious about MIDI, MIDI is where Acid needs work. After half an hour of serious digging around, I still couldn't figure out a simple task like switching a MIDI patch (using the built-in Sonic Foundry soft synth). Even scouring the help file and following instructions to the letter left me hanging. Trying to add some of the cool VST instruments, like NI's great B3 and DX7's emulations, also proved to be more difficult than I'd like. Acid handles its MIDI tracks well once preferences and settings are together, but getting them there proved less intuitive than most sequencers.

For those loop masters out there who use Acid daily, 4.0 is an extensive upgrade that leaves little to be desired when working with audio and loops. There are a million ways to chop, slice, dice, and mix those samples. Playback response and overall program speed was good even on my elderly PII 400. The interface, while different from mainstream sequencers, is easy to use, and yet it masks the power of Acid in its simplicity.

There are a million parameters, plugs, and envelopes in 4.0, but strangely enough, getting to them all is the hard part. The program's menus are almost barren. Where are all those cool plug-ins? You'll have to dig through the channels, mixer, and other areas to find them. And while MIDI is not Acid's strong point yet, I'm sure future versions will improve. The level of creativity Acid Pro offers for remixers, sound designers, and composers who do loops is unequaled, and when MIDI catches up, the whole package will be an indispensible tool.

★★★★½ | Acid Pro 4.0
Sonic Foundry | www.sonicfoundry.com

Gene Porfido runs his own company, Smilin' Pig Productions. E-mail smilinpig@earthlink.net.

Return

Interactive Profiling, Part 1

Working with different development houses, I've seen that it's often extremely difficult to get anything done. Most developers don't work efficiently. Our programming languages, compilers, debuggers, source control systems, and profiling tools are all flawed in one way or another. These flaws cost us huge amounts of time and money. This month, I'd like to attack one corner of this problem space: the inadequacy of our profiling tools.

For the practicing game developer, there are a handful of commercial profiling products to choose from. Two representative examples are Intel's VTune and Metrowerks' CodeWarrior Hierarchical Profiler (HiProf). Despite being profilers, these products are inappropriate in several ways, and using them in a production environment can be painful. Usually a profiler will slow down the game tremendously, preventing us from reproducing the situations we want to profile. VTune is not slow if you do not collect any information other than the program counter at each sample, but that information is only useful in limited circumstances.

Because most game developers are not in the business of selling tools, the profilers we write ourselves are usually minimal.

Because these products don't satisfy our needs, a lot of games get rigged with a homemade profiler. We manually mark sections of the code with telltale names like "`render_world`" or "`simulate_physics`," and a heads-up display shows, in real time, how much CPU we're spending in each area.

How Interactive Profiling Is Different from Batch Profiling

There are some ways in which even the simplest in-game profiler is more useful than a tool like VTune or HiProf. With a real-time heads-up display, you can see correlations between the frame rate and the activity of each game subsystem and use human pattern-matching and intelligence to figure out what's happening.

Suppose your game's frame rate stutters annoyingly in a map with lots of objects and busy activity, so that roughly one frame out of every 30 takes three times as long as it should. If we try to diagnose the problem with VTune or HiProf, we'll meet with some difficulty. Since the long frames are intermixed with normal frames, the code causing the problem is only taking one and two thirds as much CPU as it should, when averaged over time. This amount probably won't be sufficient for that code to stand out as the culprit in the profiling report.

We might be able to track down the problem in VTune by enabling the "Collect Chronology Data" option. We'll suffer through annoying slowdowns, and if the problem is timestep-dependent, it may no longer occur. To solve this problem robustly, we can build our game engine to support full journaling and playback. That's useful in its own right, but it's a big engineering task that we'd rather not require just to obtain a profile. After all this, we may discover that the problem is caused by our AI pathfinding code. We may even know that a specific loop is eating up a lot of time. But the profiler can't tell you why.

On the other hand, if we use an in-engine interactive profiler, we'll see the problem instantly because we're watching a live report that changes from frame to frame. Every time the frame stutter happens, we'll see the program section "AI Pathfind" jump to the top of the chart. And because we're running around inside the game world, we may be able to easily correlate the spikes with specific problems. For example, now that we know the problem is pathfinding, we can see that it happens whenever a monster walks near a certain kind of lamp. So we go investigate the geometry of the lamp to get to the heart of the matter. Because we can turn on the interactive profiler at any time, we don't have to strain to reproduce a test case. When it happens, we just hit the function key that turns on the profiler and it shows us what's happening.

Batch profilers try to help you visualize data with bar charts

JONATHAN BLOW I *Jonathan is a game technology consultant hanging out in Austin. He can't find anything worth stealing under 15 feet of white snow. Send him e-mail at jon@number-none.com.*

and pie charts and hierarchical diagrams. But our game renderer is already a rich visualization of the game's variables, a far more complex and meaningful one than any batch profiler will give us. Augmenting the world rendering with some supplementary data is usually the best visualization approach.

There are two things batch profilers are good for: giving us a broad idea of a program's behavior averaged over time and giving us precise, instruction-level performance data for small sections of code. We can obtain the former just as well through a simple, in-engine profiler, and the latter represents a small fraction of the profiling use cases of a modern game.

As game developers, we frequently write our own in-engine profilers, but it's seldom articulated that interactive profiling is a different paradigm from batch profiling. The batch profilers available to us now are useful, but most of the time we really want a good interactive profiler.

Because most game developers are not in the business of selling tools, the profilers we write ourselves are minimal. I think it would be worthwhile to spend some up-front effort to build a high-quality interactive profiler that people can just plug into their games. This month's column and the next will take steps toward that goal.

Goals of the Interactive Profiler

I want this profiler to be effortless to use. It should be so fast that you can always leave it enabled in every build of a game, removing it only when the time comes to ship the final product. Disabling profilers with `#ifdefs` in various builds is a bad idea. Suppose you see a problem and want to check it out, but your current build doesn't have profiling turned on. You have to stop, change the `#define`, do a full rebuild (which may take a while), and finally run the game and try to reproduce the case (which can be hard). That's exactly the kind of process impediment that stifles game development. By itself it doesn't seem like much, but these situations build up to where many impediments affect our daily progress, and it becomes a chore to do anything.

I'd like it to be easy to detect the kinds of performance problems we care about with the profiler. I also want the profiler's behavior to be reliable; for example, it should not change in character based on the frame rate. And I think it would be helpful if it provided some tools that help us jump to a higher level and see the different kinds of behaviors our games exhibit and how common each behavior is within the overall gamut.

Profiler Overview

I'll create a small header file that I'll include in all code I want to profile. The header contains macros for declaring different zones of the program (rendering, input, physics, and so on) and for stating that code execution is entering or leaving a given zone.

When we enter a zone, we'll ask a high-resolution timer what time it is and record that time. Upon leaving the zone, we subtract that entry time from the current time, which tells us how long we were inside. We add this to a total-time accumulator for that zone. Once per frame, we draw the accumulated results and reset them.

With a little bit of extra accounting, we can separately track what HiProf refers to as the "hierarchical time," which is the time spent inside a zone and all zones that it calls, and "self time," which only counts time within a specific zone body and stops when a new zone is entered.

Zone Declaration

I chose to declare zones by instantiating structs in the global namespace. Each struct contains the variables used to store the raw timing data for that zone. For convenience, I created a macro called `Define_Zone` that declares the appropriate instance (Listing 1). When entering or leaving a zone, we refer to the name of that global variable; all references to zones are resolved at link time, so no CPU is spent at run time matching zone IDs. Also, you get a compile-time error if you try to use a zone that isn't declared, which is good for catching spelling mistakes.

There are two alternatives for declaring zones that people often use, but they both have big disadvantages. One method is to create a global header file with enumerated constants for each zone; these constants index into an array. With this approach, adding a new zone requires a full rebuild of the project, and full rebuilds are horrible for many code bases. Also, we often want to create temporary zones incrementally, which is painful with this scheme. The struct-based scheme I'm presenting here is confined to the files being worked on, requiring minimal recompilation.

The other alternative is to use strings to designate zone names, but this option incurs a lot of run-time overhead, and

LISTING 1. The `Define_Zone` macro helps the profiler store timing data efficiently for each zone.

```
struct Program_Zone {
    Program_Zone(char *name);

    char *name;
    int index;

    Profiling_Int64 total_self_ticks;
    Profiling_Int64 t_self_start;

    ....  // Other members deleted for brevity
};

#define Define_Zone(name) Program_Zone PZONE_ ## name(#name);
```

spelling errors can go unnoticed for a long time.

Entering and Leaving Zones

It's easy to define macros to indicate that you are entering or leaving a given zone. When entering, the zone is pushed onto a stack; when leaving, it's popped. However, I don't like using these macros directly, as it's too easy to forget an exit declaration in functions that have a lot of exit points (Listing 2). I prefer to make a C++ dummy class that contains no local data, but that calls `Enter_Zone` on construction and `Leave_Zone` on destruction. Then I can instantiate that class whenever I want to enter a zone, and leaving will occur automatically.

Getting the Time

We want to acquire an accurate timestamp as quickly as possible. We could use the Windows API function `QueryPerformanceCounter` (QPC) for this, but QPC is very slow. In Windows 2000/XP it's a wrapper function in a DLL, which calls a different wrapper function in a different DLL, which triggers a kernel trap to a routine that reads the PCI bus clock. If you want to be saved from the agony of including windows.h in every file of your game, you'll probably wrap QPC yet again. Add all that up, and it's just not lightweight.

Instead, I wrote some inline assembly language that uses the `rdtsc` instruction, which has been available on all new x86 chips for years now. This gives me integer timestamps in units of some arbitrary chip frequency, but I want time measured in seconds. For maximum speed, I perform all the timestamp accounting directly on these integer timestamps and only convert to seconds once per frame, when it's time to process and display the profiling data. To perform the conversion, I use QPC at the beginning and end of the frame to get a frame duration in seconds. By measuring how many `rdtsc` ticks have passed during the same interval, I can convert from rdtsc ticks to seconds.

Unfortunately, newer mobile processors have a feature called SpeedStep, wherein they dynamically adjust their clock speed based on demand. Even though `rdtsc`'s name ("Read Time Stamp Counter") indicates that it was intended as a timer, someone at Intel decided that it's O.K. for `rdtsc` to report clock cycles without compensating for SpeedStep. As a result, if you're running on a laptop and your application has bottlenecks that involve blocking on I/O or accidentally sleeping, `rdtsc` is unreliable as a measurement of time. Fortunately, the profiler can detect this situation and report that SpeedStep is causing a problem (it just looks for fluctuation in the number of `rdtsc` ticks per QPC second). However, there is no way to fix the profiling data itself. If this happens to you, I recommend using a desktop machine for profiling.

Displaying the Data

We draw the profiling data with one zone per line, sorting by CPU usage so that the most expensive zones are at the top. But since timings in a game fluctuate a lot, if we do this naively, lines of output will move around haphazardly and the display will be hard to read.

This is a good application for the frame-rate-independent IIR filters I introduced two months ago ("Toward Better Scripting, Part 1," October 2002). We apply these once per frame to the timing values and find the resulting numbers are more stable and the display is easier to read. By varying the filter coefficients, we can decide whether we want to see long-term averages or very recent values.

But we don't just want to know about frame-time hogs. Consistency of frame rate is very important, so we should also seek out routines that are inconsistent in the amount of time they take. In that column on scripting, I showed how to compute the variance of a filtered variable. We square-root this number to get the standard deviation (`stdev`), which is measured in seconds. Now the profiler can display and sort by self-`stdev` and hierarchical `stdev`. In addition, we color-code each line of output depending on the predictability of the zone's timing. Steady zones are drawn in a cool color, and vibrating zones are drawn in a warm color.

Sample Code and Next Time

This month's sample code (which you can download from www.gdmag.com) is a toy game system with the profiler built in. There's a simple world full of crates. Besides rendering, the game has dummy functions for AI, audio, and so on. To ensure that the different systems run slowly enough for the profile to be interesting across a variety of machines, they artificially slow themselves down. Some functions, such as the AI code, fluctuate their CPU usage so that the profiler can detect that variance.

The profiler itself is very modular, so you should be able to use it in your own game with minimal effort. I've only just gotten started exploring what we can do with an interactive profiler, and next month I'll look at some data processing and visualization enhancements.

LISTING 2. Multiple zone exit points can cause problems.

```
bool foo(char *s) {
    Enter_Zone(foo);

    if (s == NULL) {
        Exit_Zone(foo);
        return false;
    }

    char *t;
    // Do something complicated with s,
    // putting the result into t...
    t = result;

    if (!well_formed(t)) {
        Exit_Zone(foo);
        return false;
    }

    do_something_complicated_with(t);

    Exit_Zone(foo);
    return true;
}
```

Return

How to Play the Interview Game

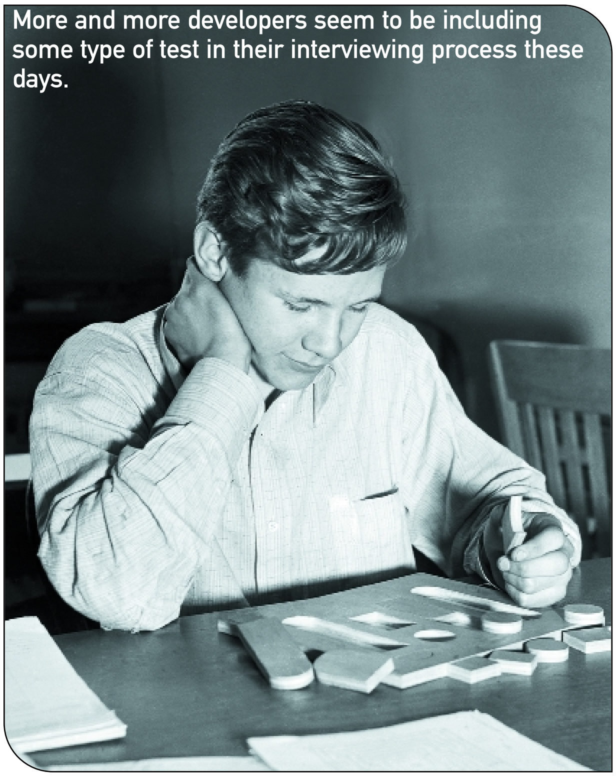
I should have gotten up and left before the (seemingly) 14-year-old assistant manager came into the makeshift waiting room and stopped chewing her gum just long enough to say, "Hayden Deeval? Deval? Durvall? Is that you?"

Thirty-five minutes earlier, I had entered the large electrical goods retail outlet and had been guided to sit next to the other smartly dressed twentysomething waiting to be interviewed for the position of store assistant. If I remember correctly, my fellow interviewee was called Henry, and as we waited to be summoned, we discussed, shamefacedly, how appalling it was that two intelligent young men with degrees in history, psychology, and film could be competing for such an underpaid, mind-numbing job, advising people on the relative merits of cylinder versus upright vacuum cleaners. But still, neither of us left.

Choosing not to dispute the pronunciation of my name with the girl who might shortly end up being my boss, I indicated that I was the person in question, and she motioned for me to follow her from one starkly lit room into another, where a tired-looking man wearing a tie that could have stopped a charging rhino at 25 paces smiled the smile of the weary and offered me a seat.

The interview as a whole can be summed up with the opening question: "So, Mr. Duvall, why do you want to get into retail?" To be honest, as obvious as the question now seems, I genuinely wasn't expecting it. Sitting there, as my heart, still riding the caffeine bicycle from breakfast, pounded in my ears, the only thing I could think of to say was: "Well, it seems like a career with a lot of potential." Potential? Potential what? Potential downward spiral into a chasm of despair and hopelessness?

What I should have said was, "Actually, I would rather eat my own foot than work here, but it's a job, and I have two kids to feed," but whoever said that honesty is the best policy was a bit naïve. Not surprisingly, I got a letter of rejection within the week, but the experience taught me two things: First, if you are not serious about a job, chances are that this will show up in the interview. Second, if you are serious, you'd better be prepared.


More and more developers seem to be including some type of test in their interviewing process these days.

Game Industry Interviews

Is an interview for a job in the game industry really different from the majority of job interviews? Some elements are common to any area of business (like trying not to sneeze on the interviewer), but as the game industry is in many ways peculiar (just look around the office), it needs to be treated as such.

While it goes without saying that each studio has a different approach to the whole recruitment process, after I did some research across a range of developers, I found it is possible to get a reasonable picture of what to expect and how to maximize your chances for success.

The purpose of the interview will vary between developers, but in most cases, the main thrust of the time they spend

HAYDEN DUVALL | *Hayden started work in 1987, creating airbrushed artwork for the games industry. Over the next eight years, Hayden continued as a freelance artist and lectured in psychology at Perth College in Scotland. Hayden now lives in Bristol, England, with his wife, Leah, and their four children, where he is lead artist at Confounding Factor.*

with you will be assessing you as a person, with regard to how you might fit in with the team and project for which you are applying.

Getting Yourself Ready

Preparation is always touted as being the key to success, and there are a few areas where some research and advance consideration can definitely help.

Study up. Don't go for an interview without any knowledge of a developer's past games. If it is a new studio, it's not necessarily important to establish what the founders did previously, but it's very difficult to be convincingly enthusiastic about working for a developer if you don't have much of an idea about their past games. The best preparation is to have played several of the games the studio has released. If you want to be written off as someone who's just looking for any old job, and not this specific job, then when they ask what you thought of their last game, just stare blankly at them (or perhaps say it had a lot of "potential").

Rest up. Try to get some sleep the night before. It sounds trite, but the burned-out look triggers instant alarm bells most of the time. Ironic as it is, most developers expect that you will come to work for them as fresh as a springtime daisy, and that they will have the pleasure of burning you out.

Gussy up. Iron your clothes and have a shower. This advice may seem facetious, but too many experienced industry veterans forget that applying for work somewhere else requires a certain level of superficial grooming. Ours is not an industry of high fashion and excessive hairspray, but the difference between working on a team and going for a job interview is that in the latter you come into contact with management. They spend less time on the shop floor and more time in front of a mirror. Until you have earned a place on the team, you need to give the 1987 Iron Maiden World Tour T-shirt a few days off.

Build up. On average, between 80 and 95 percent of all portfolios and demo reel submissions don't result in an interview. So remember that they already like your work, and that this is a solid base from which to build once you are talking to them.

Dealing with Pressure

Of all the exams I have ever taken, my driving test made me the most anxious. There's nothing quite as bad as having to perform live, and being nervous in an interview is perfectly normal. A classic mistake is to overcompensate and end up with what those familiar with the film *Trainspotting* will recognize as "Spud Syndrome."

In the same way that most developers I surveyed expressed dissatisfaction with interviewees who knew nothing about their previous work, many also highlighted the negative impression left by those who raved on obsessively about their last games. Sufferers of Spud Syndrome (whether amphetamine-induced or not) tend to overwhelm the interviewer with a tidal wave of enthusiasm, which ends up having the same effect as if they stripped naked and ran around the office singing, "I love you, give me a job."

Questions

Most hiring managers rely on a portfolio or demo reel as proof of your artistic talent. They then use the interview process to establish:

• That the work they have seen is actually yours.
• That you have represented yourself accurately in your résumé.
• That you are the kind of person who will fit well with the team.
• That you are the right person for the specific position that they need to fill.

The exact form that an interview can take varies a great deal. Some companies have a single, long interview after a thorough telephone vetting procedure, and some use a short, preliminary interview followed by something less formal once you have impressed them sufficiently. Whatever the format, the following questions are likely to feature at some point:

What kinds of games do you like? Remember that there is no correct answer to this question. It is often tempting to present yourself as an RTS nut if the company you are applying for works in the RTS genre, but this fact is usually of no real consequence; the most important thing is that you come across as someone who enjoys playing games.

There are people in the industry who have little real interest in games themselves, and quite often an artist who has moved from another industry or another area of art will not be into games. If his or her work is good enough, many studios won't mind, but the general feeling among employers is that avid game players are equipped with a better understanding of the context in which their work will appear.

What kind of games have you worked on, and what was your role in their production? Once again, this is a standard interview question. The information will be available on your résumé, but as everyone who has ever been involved in recruitment (or has ever applied for a job) will know, a résumé is often about as reliable as a chocolate surfboard.

Talking to artists about their role in the production of past games is the best way to get a feel for what they actually did or didn't do. The number of "lead artists" in our industry is in many ways similar to the number of people who claim they attended Woodstock, largely fictitious.

Employers see it as a major black mark when candidates represent themselves as much more senior than they actually are. Hiring someone who needs to have real experience of leading a team, when it turns out that the only leading they really did was in the lunchtime sprint to the sandwich shop, usually leads to a speedy and unceremonious parting of ways.

What Are Employers Looking For?

When asked who would most likely be doing the interviewing, more than half of the developers I questioned suggested that each candidate would be interviewed by a panel of some kind, or at least two people, usually the lead artist and either some additional senior member of the art team or the project lead.

Outside of your artistic talent, there are three areas where it is vital to make a solid impression at the interview.

Understanding the process. Showing that you have a grasp of the game development process illustrates that you are not simply an artist but a capable game artist, which includes an understanding of the context in which you will be producing your art. Each game and team are structured differently, but demonstrating awareness of level structure, prototyping, alpha and beta phases, and so on, will help underline your competence.

Understanding the techniques. If you've produced a quality demo reel or portfolio, employers will see that you can deliver a high standard of work. Beyond that, a discussion about the merits of texture baking or vertex coloring, for example, can establish whether you have a broad range of techniques at your disposal. As most games have their own particular limitations, showing that you are able to get maximum impact within these boundaries is a great advantage.

Understanding the technology. As a game artist, you need to be as technically well-versed as possible. Texture memory, bump mapping, multi-texturing, and so on, across the different platforms, all influence the art produced for each game. Most developers I talked to indicated that an artist who can work closely with a programmer to help get the maximum performance from their technology is a rare and precious thing.

Fitting In

As I mentioned previously, the whole interview experience is largely a forum in which your future employer (or their representatives) can establish whether they want to work with you, and if they do, how much you are likely to contribute to the team.

The only possible advice I can give (even though it is painfully lame) is to be yourself. To be fair, to some extent you will be the interview version of yourself, but the more you try to put across a personality that isn't your own, the more potential trauma ends up lying in wait to ambush you in the future.

Your interviewers will remember most clearly the first and last contact they had with you.

The bottom line is that if they are put off by the way you came across, and that is in fact the real you, chances are you wouldn't have enjoyed working there anyway.

The Test

More and more developers are including some type of test in their interviewing process these days. Each is different, but they ask you to perform a set task in a given time limit, using a specific package or medium. Those tests that I have come across have ranged from a one-hour pencil and paper exercise to a full level-building scenario complete with texture limit and style guide.

It's important to remember that any test of this kind is unlikely to represent your best work, and that it is also unlikely that you will be thrilled with what you produce. From the hiring manager's point of view, they want to see firsthand how you approach a problem and whether the work you produce fits to some degree with the art in your portfolio.

Problems arise when a candidate doesn't clearly understand the instructions for the test (and I have seen some pretty awful sets of instructions). Don't feel too intimidated to ask for clarification.

Final Thoughts

Primacy and recency. Your interviewers will remember most clearly the first and last contact they had with you. First impressions always count, but last impressions do, too. Try to leave on a positive and enthusiastic note. If you feel yourself drifting off in an interview (it's not hard to zone out for a couple of minutes), snap yourself back into full alertness before they pick up on your glassy stare.

Asking questions. Most developers I spoke to confirmed that they generally end interviews with the classic "So, do you have any questions for us?" Anticipate this, and have a couple of intelligent questions in mind. What employers usually do not appreciate is an immediate plunge into questions about working hours, pay, benefits, and royalties. If you ask about these things, employers will see you as focused on your paycheck and not the project.

Follow up. Unfortunately, not all developers are the highly oiled machines of efficiency that they think they are. This means that after the interview, you may be left hanging for an unnecessarily cruel length of time. Before you leave, you need to establish when you can expect to receive word on how it went, and once this deadline has passed by two or three days, it is entirely appropriate to check up on things. Do not, however, turn into a pest; if one or two e-mails or phone calls haven't elicited a response, wait a while longer and try again, but after that, you should always assume that they aren't going to take you on.

In the end, interviews are almost wholly subjective, and they can be affected profoundly by the ravages of nerves or even the aftermath of a bad burrito from the night before. Try to relax and avoid putting on an interview "performance" to maximize your chances. And someday when you've made the leap from interviewee to interviewer, remember the path that led you there, and please, be gentle.

Return

The Live Orchestra Recording: A Producer's Awakening

The recent production of live orchestra soundtracks for games like MAFIA and MYST III provokes a stunning question: If the time spent massaging synthesizers were allocated towards a live orchestra, could the value of that time cover the cost of a live orchestra?

Given the right circumstances, it most certainly can.

Time is money, and a significant part of a composer's fee is based upon the time it takes the composer to program synthesizers. By dismissing the synthesizers, a producer is significantly reducing man-hours.

I recently finished work on the score for *Merregnon 2*, recording the entire soundtrack with the Prague Symphony Orchestra. A typical cue from this score would have taken even the most experienced synth programmer 35 to 40 hours to create an almost-real sounding MIDI recording. Instead, I spent under an hour recording that very same cue with 54 live musicians. If you do the math, you'll see that the producer has a choice: pay the composer for the time it takes to massage the synths, or pay an orchestra for the time it takes to record live.

Let's consider the cost effectiveness of a live orchestra by examining a sample budget for a 50-minute soundtrack recorded by a 50-piece orchestra. According to Aaron Marks' *The Complete Guide to Game Audio* (CMP Books), "an established composer can charge $1,500 per finished minute of music." As such, a $50,000 budget for a 50-minute orchestral score is a conservative estimate.

A competent 50-piece European orchestra will cost $10 to $20 per player per hour, including studio costs. Armed with well-conceived orchestrations, flawless parts and a knack for conducting, an equally competent composer can record 50 minutes of music in 10 hours:

50 players @ $20 per hour × 10 hours = $10,000

During an orchestral recording session we need the assistance of a support staff:

4 support staff @ $30 per hour × 10 hours = $1,200

Based on my trip from New York to Prague, when I was accompanied by a senior producer:

Travel and accommodation for 2 people = $1,600

Finally, let's give ourselves a safety net:

Miscellaneous expenses = $1,200

Adding up all these costs, we get:

Total cost of production = $14,000

With a $50,000 budget, the producer is now left with $36,000 for the composer's creative fee. Is it possible to convince a composer to give up 28 percent of his or her fee for a live orchestra recording? I suspect a skilled composer would do it in a heartbeat.

On the composer's end, by dismissing the synthesizers, you have eliminated hours and hours of extremely labor-intensive (and hence costly) work. In my earlier example, I suggested that it would take four or five days to program a four-minute cue in a manner that would yield an almost-real-sounding MIDI orchestra. That's approximately one minute per day. In our budget example, we have eliminated 50 minutes, or 50 days' work. Given an eight-hour day, we have thus eliminated 400 hours of work. Provided you have a composer that can make the transition from MIDI to live orchestra, it is well worth $14,000 both to eliminate 400 hours of work and to have a live orchestra recording at the end of the day.

In order to record a live orchestra successfully within our budget confines, your composer must be able to make the transition from MIDI to live orchestra. To do so, the composer must have the skills necessary to write the music, appropriately orchestrate it, create proper sheet music, and conduct the music during the recording sessions. Make sure before you sign a contract with a composer that he or she can do all these things and has a proven track record in all these areas. There is a Catch-22 that confronts the game music industry: in order for orchestral game soundtracks to rise to the next level, we must move to live orchestral recordings. But as the quality of sample libraries increases, the skills needed to handle a live orchestra decrease.

Armed with both a solid understanding of orchestral writing and an in-depth knowledge of the available software tools, it makes economic sense for both the composer and the producer to accept the budget I just outlined. It's a win-win situation when, by reallocating resources and finding the right composer, a producer can take an all-MIDI music budget and create a live symphony orchestra soundtrack.

Andy Brick conducts the Prague Symphony Orchestra during the *Merregnon 2* recording sessions.

ANDY BRICK | *Andy is a New York–based composer whose game credits include* SHADOAN, THE FAR REACHES, *Merregnon 2,* KID PIX DELUXE, *and* TESSELMANIA. *Andy's feature film credits include* Little Mermaid II, Lady and the Tramp II, *and* Chatham County. *You can learn more about Andy and listen to cues from many of his soundtracks by visiting www.andybrick.com.*

Fun for the Player

This month's rule is one I learned from one of my favorite game designers, Sid Meier, at a conference years ago. In the same lecture he acknowledged that many of his best ideas have been borrowed from other sources, and in that spirit I pass it on to you.

The Rule: Make the game fun for the player, not the designer or computer.

This may seem like an obvious concept, but often game designers forget that the player is the final audience. It's hard enough to make a game fun for the player — in fact, that's what most of the craft of game design is about — but it's even harder when you lose sight of your audience.

The Rule's domain. This rule applies to the process of game design and is meant as a warning for designers to avoid a common pitfall.

Rules that it trumps. "Make only games that you really enjoy playing." I've heard this rule stated many times, most often by people who have never actually designed a game. It's not a bad rule, but it can lead to the mistake of making a game that is fun for the designer. An aspiring designer may reason, "I'm representative of the audience I have in mind, so if I enjoy playing the game it will be popular." But there are a couple of fallacies there. First of all, a person with that much interest in games, the willpower to follow through and actually make one, and the skills to do it well is not representative of the general game-playing public. Second, once you've been working on a game for many months, you perceive it differently from someone coming to it for the first time. If you continue to tune and tweak the game to be fun for you at that moment, you're almost certain to make it inaccessible to new players.

How much fun did Sid Meier have designing ALPHA CENTAURI? It doesn't matter.

Rules it is trumped by. I can't think of any rules that should trump this — can you?

Examples and counterexamples. Since this is a rule of negation, let's consider what happens when it is broken. How do designers make games that are fun for themselves? There are many ways besides the tuning example I just mentioned. There's another that I call the Egotistical Dungeon Master trap, where a designer considers the point of the game is to show that he is smarter than the players. This can take the form of a combat-oriented game where the player is encouraged to follow a series of easy victories, only to be ambushed at the end by vastly superior forces and learn later through trial and error that the only way to win is to take the least logical step. Or it might be an RPG or adventure game where the only way to get past a monster is to possess an arcane and obscure piece of real-world trivia that happens to be a favorite topic of the designer. The point is to always remember that we designers succeed when the players have fun, not when they curse in frustration.

Making a game that is fun for the computer is less common, but still a danger. Let's take an illustration from a war game. The player clicks on a German artillery piece and then on its target, an American tank. Then the computer goes to work. Here's what it is thinking:

"Let's see, the gun is an 88, model G, from 1943. It's been in service for two months and has seen heavy use. The barrel was last replaced 14 rounds ago, and it's still in fair shape. The shell fires successfully, muzzle velocity 178.4 meters per second, the wind is from the northeast at 8 km/h, temperature and humidity will slow the shell by an additional 0.7 percent — it's a hit! But the range is 2,743 meters, on frontal armor, at an angle of 76.23 degrees off of perpendicular — the shell does not penetrate and detonates harmlessly."

Several milliseconds later, the player sees the message, "Attack ineffective." The computer has all the fun. Of course, if the designer had made most of that information accessible to the player, the player would have had more fun (at least the sort of player that likes WWII war games), and the designer probably had a good time taking all those factors into account. Making a game fun for the computer is a special case of making the game fun for the designer, when the designer is probably also a programmer and so knows what is going on beneath the surface.

I'm tempted to end this column with an obscure joke that I'm sure only a few of you would understand, but that would be missing the point, wouldn't it?

NOAH FALSTEIN | *Noah is a 22-year veteran of the game industry. You can find a list of his credits and other information at www.theinspiracy.com. If you're an experienced game designer interested in contributing to The 400 Project, please e-mail Noah at noah@theinspiracy.com (include your game design background) for more information about how to submit rules.*

Animation for the Masses

Choosing the Right Facial Animation Technique

Thanks to ever-increasing graphics horsepower on contemporary platforms, animators working with real-time 3D (RT3D) facial animation have access to tools and technology previously available only to those in the film and video industries. This flexibility can be a double-edged sword, since choosing the specific technique appropriate to your project is a big responsibility. Facial animation can be expensive and time consuming, and it isn't any fun to find out — too late — that you've started down the wrong path.

Like most development choices, the decision to implement a particular technology is a balancing act between time, money, user features, and in-house expertise. If you have all the information in place, deciding which system to implement is often straightforward. This article will shed some light on how to choose the right facial animation technique.

In general, all the various facial animation methods available to RT3D animators can be classified in one of two categories, each characterized by how the 3D surface is manipulated. For discussion purposes, we'll group these techniques into skeletal networks and morphing systems.

Skeletal Networks

The skeletal network technique includes any system that uses a set of nodes or channels to influence the 3D surface indirectly, either by affecting clusters of vertices through weighted envelopes or through volume effects such as free-form deformation matrices. The most common implementation is a simple skinned surface weighted to a network of bones, as shown in Figure 1.

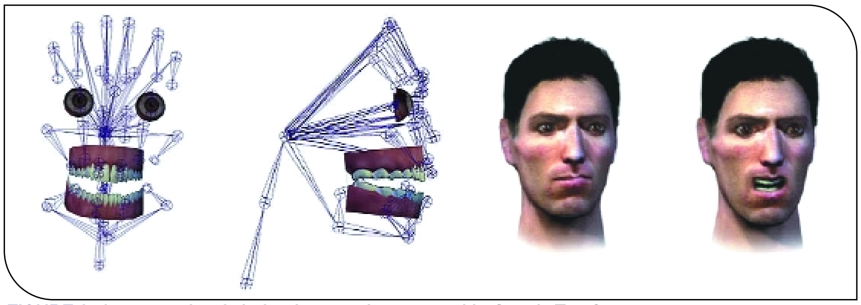


FIGURE 1. An example of skeletal networks, as used in Sony's THE GETAWAY.

There are two key advantages to using skeletal networks for facial animation. The first is that because the animator is only indirectly affecting the 3D surface, he or she need not have access to the actual "face" being animated. This means that the facial animations can be generated in parallel with or even ahead of other development, without requiring access to the exact character mesh. Also, because there is only a loose dependence on the specific 3D surface being animated, the animations generated using a skeletal network are fairly easy to port between different facial surfaces.

The second advantage is that skeletal animation of skinned surfaces is probably the most widely supported in animation packages and run-time engines. The tools and techniques for skeletal animation are proven and reliable, and there is a wide variety of choices individual animators have at their disposal.

However, a side effect of the indirect method is that the facial animator does not generally have direct access to the facial surface being animated, implying that some amount of finesse is being for-

MEL GUYMON | *Mel has been working and consulting in the real-time 3D community for several years and is currently working on a yet-to-be-announced project in Silicon Valley. He can be reached at mel@infinexus.com.*

feited. The inability to introduce extremely fine-grained nuances into facial animation is the primary disadvantage to skeletal networks.

The second disadvantage is related to performance: given a particular character being animated in RT3D, the increase in the number of bones may become problematic. At run time, part of the cost of putting a character on-screen is the transform cost associated with all of the bones in a skeletal network. Typical bone counts for average RT3D characters are around 20 to 30 bones for the body, arms, and legs; yet in order to get any real degree of finesse, a skeletal network may require some 20 to 30 additional bones for the face alone. And although advances in hardware support for skeletal transforms and aggressive use of skeletal level of detail (SLOD) can alleviate this problem, you need to plan for it in advance.

Morphing Systems

The morphing system technique includes any approach involving direct surface manipulation: morph targets, BlendShape, and direct vertex or CV animation. As shown in Figure 2, the most common version of this involves the use of a single primary surface, with which are associated any number of morph targets (the Morpher Modifier in 3DS Max and the BlendShape feature of Maya are good examples), where a morph target is a duplicate of the primary surface stretched into a different shape.

Morphing systems come with two primary advantages. The first is that direct access to the 3D surface offers the animator an incredible amount of finesse and nuance in facial animation. For this reason alone, many animators choose morphing systems outright in lieu of any other implementation.

The second advantage is the speed at which changes to the primary surface can be integrated into the facial animation. This is a subtle point that I'll cover in more detail in the case studies that follow, but in essence it means that you can modify the shape of the primary surface directly, and the facial animations previously built for it will generally still work.

The major disadvantage of morphing systems is their dependence on parity between the primary surface and its associated morph targets. The vertices in any 3D surface each have an assigned number, or index. Morphing animation works by generating deltas off of each numbered vertex. This means that in order for there not to be any artifacts, the vertices of the primary surface and all its dependent morph targets have to match exactly — both the same number of vertices and the same vertex indices. For example, if vertex #57 is part of the eyebrow in the primary surface, but in one of your morph targets vertex #57 has somehow been assigned as part of the ear, you'll end up animating the ear when you meant to animate the eyebrow. To avoid this, structural changes to the number and ordering of vertices in a facial surface need to be done very carefully.

The other major disadvantage is that for complex surfaces the number of vertices being morphed can become a limiting factor to performance. And while the specific limitations will vary from engine to engine, it's important to determine early on the exact performance limitations on how many vertices you can manipulate and still maintain frame rate.

Facial Animation Planning Do's and Don'ts

DO USE BUILDING BLOCKS. Whether you're using morph targets or nodes, deconstruct your character's face into regions. Common partitioning uses eyebrows (left and right), eyelids (left and right), eyes (together or left and right), upper lips, lower lips, and tongue. By creating animations for these channels first, you can combine them individually and in sets to create more complex expressions (see Gary Faigin's book in For More Information).

DON'T RELY ON SYMMETRY. Although all six of the primary emotions (happiness, sadness, surprise, fear, anger, and disgust) are commonly represented as bilaterally symmetric, subtle differences between the left and right sides of the face can add a huge amount of spice to any given expression.

DON'T SHORTCHANGE THE EYES AND MOUTH. With morph target animation, when executing changes on the primary surface for variation (as in case study 3), be especially careful around the eyes and mouth. Because the animations of these areas are fairly complex, changes to the initial surface can sometimes have unpredictable results. In general, uniform scaling of the mouth and eyes is O.K., but stretching along one or more axes or discretely deforming the surface can be problematic. Be sure to test your results against as many combinations as is reasonable.

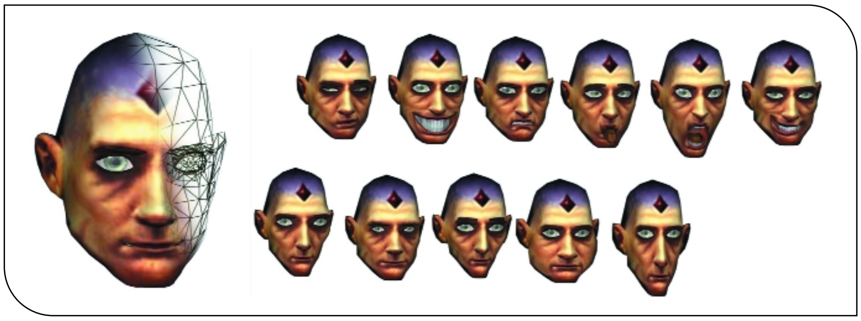


FIGURE 2. Morphing systems entail direct surface manipulation.

General Principles

In the next section I'll examine several case studies, looking at them from the perspective of the individual development teams. In each case, I'll demonstrate how the answers to the following questions helped determine the proper course of action.

First, how important is facial animation to the product? Is it a primary user feature or is it eye candy? How much of your development budget and timeline, and what percentage of the cost of the retail product, are you willing to devote to this feature?

Second, what does the art direction dictate? Are the characters realistic or highly stylized? Is there a need for subtlety to convey mood and emotion? How much diversity is there in the character population? Also, how complex are your characters? Are you dealing with many low-resolution faces or very few rendered at high resolution?

Finally, what expertise do you have in-house to deal with the art path and run-time implementations for each type of system?

Case Studies

Each of the following hypothetical case studies has been set up to represent one or more of the challenges today's development teams face. Developers in each situation take into account different considerations to arrive at the best results for the team and the project.

Case study 1: A simple web-based game. Alice, an animator on a small third-party development team, has been given the task of determining how to animate a small number of medium-resolution characters for a simple web-based game. The art direction for the product has an anime flair, it's very stylized, and the character size on-screen is small (Figure 3). For the product, facial animation will be used to convey basic mood and emotion, as well as some dynamic lip-synching. The publisher, who will have final say over the aesthetic, is providing the initial character models to the team. The tools and engine being used are off-the-shelf technology, and while support for skeletal animation is well-documented, support and tools for morphing are not. Facial animation is not the key feature of the product, and the development budget and production timelines are both fairly small. This is a quick, hit-and-run project.

Based on this information, we could predict that Alice will sit down with the engineering lead, and together they'll decide to use a skeletal network for facial animation, with a network setup like Figure 4 shows. In our tailored example, the reasons for this are obvious. Because facial animation is not a key feature, the development team is wise in choosing the most expedient path to success. In this case, because the engine and tools are not proprietary, the first course of action should be to use what already works, in this case skeletal animation.

Because this is a quick project, leveraging a single set of facial animation on multiple face meshes is a big advantage and only requires some small modifications to the bones and weighting for each character. Because of the low requirements for subtlety in the characters (the emotions are simple and the characters are small on-screen), there doesn't appear to be a downside to using a skeletal system.

The fact that the publisher is providing the initial character models and may come back to request changes to them later on is significant — if the team generated morph targets for animation and were then told to make structural changes to the primary surface, they'd have a difficult, though not impossible, challenge ahead of them.

Case Study 2: A mass-market CD-ROM title. Bob is the lead animator at a large, self-publishing development house. Bob has just been handed the product design for a RT3D product based on interaction with famous historical figures. The art direction for the characters involves a slightly stylized rendering approach with shapes that are closely true-to-life and should be accessible to the mass market (Figure 5). The character meshes are fairly complex, and the faces will be viewed close-up and only a few at a time. Facial animation plays a fundamental role in the product, both for expressing subtle mood and emotion, as well as extensive

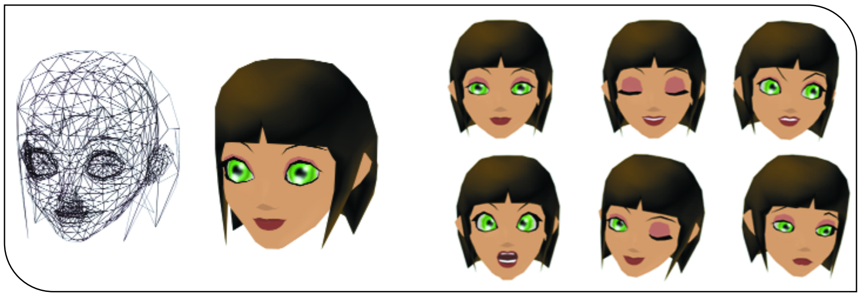


FIGURE 3. Case Study 1: Character profile for a simple web-based game. The characters have a simple, stylized look and will appear small on-screen.

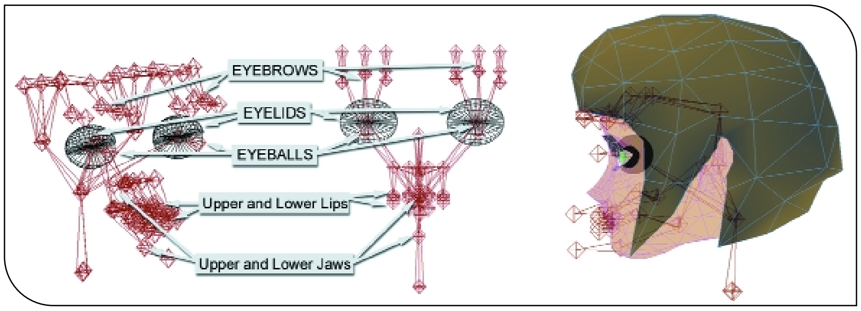


FIGURE 4. Case Study 1: The skeletal setup will lead to quick results and flexibility for later changes.

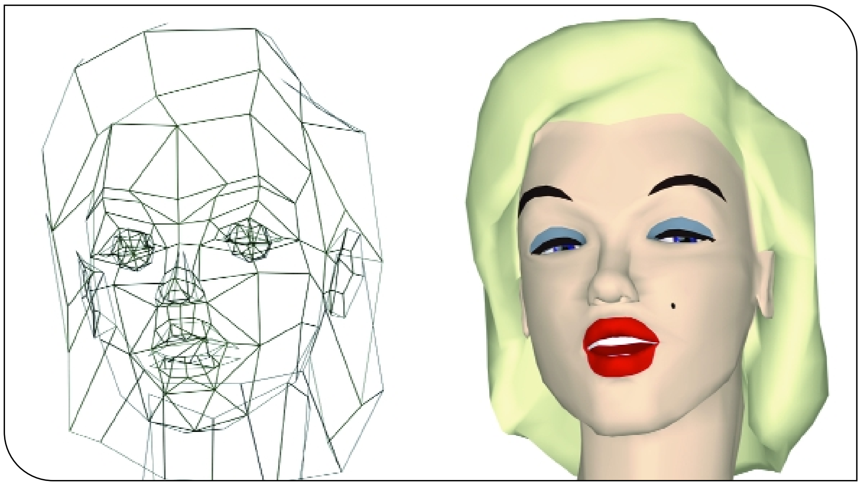


FIGURE 5. Case Study 2: A mass-market CD-ROM title with complex meshes and where facial animation is a focus.

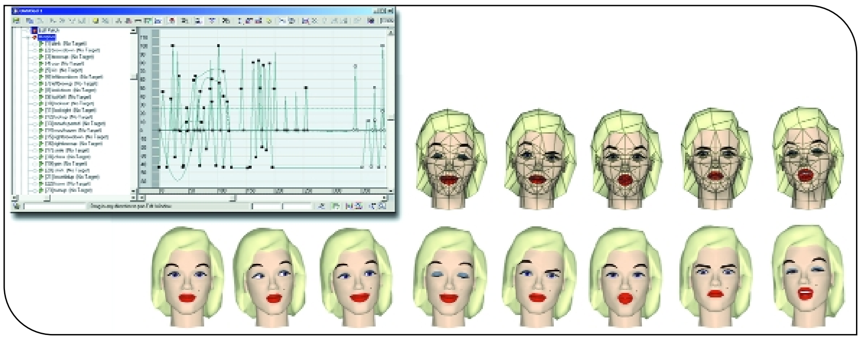


FIGURE 6. Case Study 2: Animated expressions, morph targets, and function curves.

support for lip-synch. The engine being used was developed in-house and has already been used on several successful projects. The tools and techniques for skeletal and morph-based animation are both supported. The engine has run-time support for subdivision surfaces and B-spline patch surfaces. The development house uses both 3DS Max and Maya.

Given the above information, lead animator Bob would do well to implement facial animation using a morphing system. The primary reason for this is the fidelity of expression and motion required by the product. The fact that the facial expressions will be viewed at close range, coupled with the highly recognizable figures represented, means that Bob will want access to as much of the surface as possible through direct vertex manipulation. Since the engine and tools are proprietary and support both systems, there is no development cost for creating a new process. Further, since there will only be a few characters onscreen at a time, a relatively large number of vertices and control points can be animated. Also, the in-house development probably means a more controlled development environment, so there is less chance of having to rebuild the character heads to meet someone else's aesthetic.

In Figure 6 we can see an example set of some of the dozens of morph targets that will be required for this project. Note that in both Figures 5 and 6, the surface that is actually being morphed is a Bézier patch mesh (at left in Figure 5 and top row in Figure 6) of approximately 300 vertices, whereas the actual in-world surface has been procedurally subdivided to be approximately 4,800 vertices (at right in Figure 5 and bottom row in Figure 6). This means that Bob will have much less data to work with while still being able to maintain a relatively high resolution result. This is a key factor in minimizing the amount of work involved but would not sway the decision either way by itself.

Case Study 3: Massively multiplayer character-based PC game. Adrian, the technical director, is working on a major massively multiplayer PC game. The game is a character-based pseudo-RPG and is highly dependent on giving users the ability to customize their individual appearance. As seen in Figure 7, the character art direction is highly stylized but primarily humanoid. Facial animation plays a key role in user-to-user communication and user-customization of their avatars. The theoretical space available to users for facial customization is broad and deep. The faces come in many shapes and sizes, and are cus-

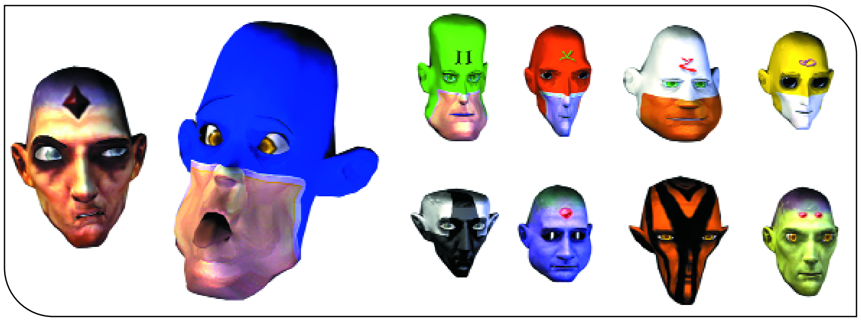


FIGURE 7. Case Study 3: Massively multiplayer character-based PC game.

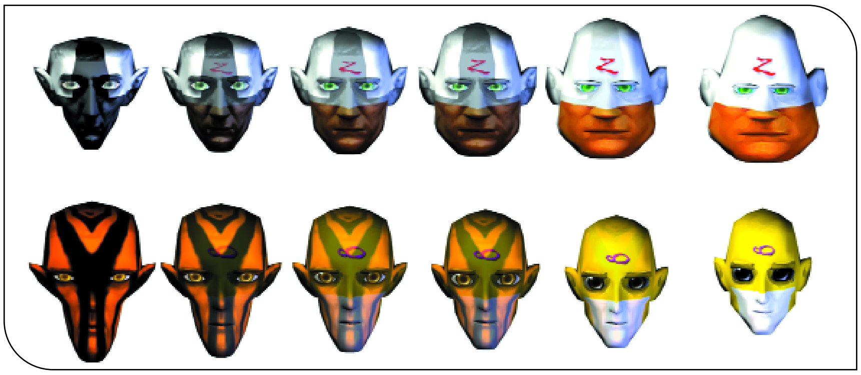


FIGURE 8. Case Study 3: Example morphing system for static head variation.

tomizable through color, texture, motion, and shape. In planning for this project, the system built must be able to accommodate thousands of user-tuned combinations. The minimum specification platform is a T&L-accelerated PC, and the engine is capable of displaying dozens of characters on-screen at any given time. For close-up viewing, the character head meshes will be subdivided to contain thousands of vertices.

On first glance, facial animation for this type of project may not seem like a straightforward problem to solve. The large number of character combinations may lead Adrian down the path of using skeletal networks, since their portable nature means they can be used on many varied facial structures. However, the key role played by facial animation in customization and communication means that Adrian will not be able to cut corners on doing the skeletal animation — it will require many bones and lots of finessing to keep the users happy.

With so many relatively high-resolution characters on screen, the cost of the additional nodes for facial animation starts adding up quickly. However, the real deciding factor in this case is the requirement to allow users to customize the appearance of their characters. The most expedient method for this is to generate a morphing system that blends between head choices. Since this system will already be in place for the static head variations, extending it to support run-time facial animation should be a small next step.

In Figure 8, we can see the power of a morph-based system coming to the fore. The art path and run-time implementation used by this system is identical to that used for the run-time facial animation. The single requirement to which Adrian will have to adhere is that blending will only be possible between heads constructed from the exact same vertex list. So, in Figures 7, 8, and 9, all of the heads shown have been created using the same initial primary surface. All of the differences are in the position and the texture — the actual vertex indices and mapping coordinates are identical for each.

In planning for this system to work, Adrian also came across another happy coincidence. Because all of the heads of a given type (if a type is a set of heads that shares the same vertex list) are essentially identical, Adrian only needs to create one set of expressions for all of them. Figure 9 shows the result of this: a single animation's keyframes and morph targets have been applied to both heads. Since the heads stem from the same primary surface, they both can be animated from the same expression. The big win is that this holds true for each of the thousands of user-created combinations of shape and texture. Put another way, it would be nigh impossible to create the same fidelity of motion and diversity of character without using a morph-based system.

Wrap Up

The correct decision of which piece of technology to use can make or break a development effort, and facial animation is no exception. By highlighting many of the most common challenges faced by teams using facial animation today, I hope this article will help you weigh some of your available options. Check your references, do you homework, and choose the technique most in line with your development goals. Happy hunting.

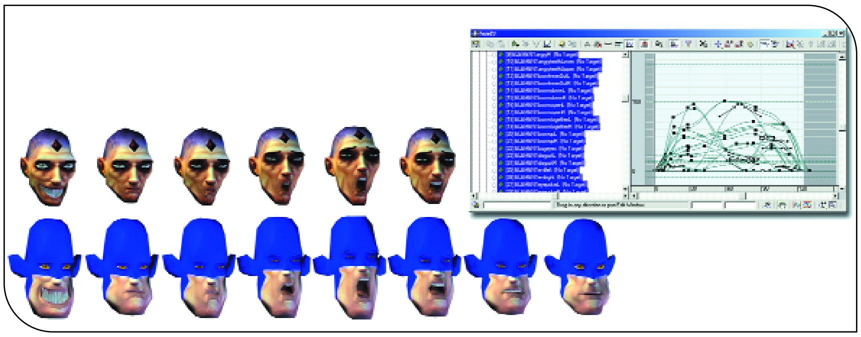


FIGURE 9. Case Study 3: Reuse of animation between disparate meshes.

FOR MORE INFORMATION

Faigin, Gary. *The Artist's Complete Guide to Facial Expression*. New York: Watson-Guptill, 1990.

Fleming, Bill, and Darris Dobbs. *Animating Facial Features and Expressions*. Rockland, Mass.: Charles River Media, 1999.

Hamm, Jack. *Cartooning the Head and Figure*. New York: Perigee Books, 1986.

Thomas, Frank, and Ollie Johnston. *The Illusion of Life: Disney Animation*. New York: Hyperion, 1995.

Williams, Richard. *The Animator's Survival Kit*. New York: Faber & Faber, 2001.

Return

RenderMan in Real Time: Staying on Top of Shaders

What a difference two years make. Two and a half years ago, I wrote in *Game Developer* ("All Aboard Hardware T&L," April 2000) about the upcoming graphics boards that supported hardware transformation and lighting. Nowadays we're all happily using hardware T&L, while searching for the next big advance that will set our games apart from the crowd visually.

What we've been waiting for has arrived. The current crop of performance video cards has upwards of over 120 million transistors — more than a Pentium 4. They can address 16GB of memory and can typically process four or more pixels simultaneously and render to two monitors simultaneously. Now the big challenge is overcoming the paradox that the more triangles we push, the harder it gets to give a game a unique visual character.

RON FOSNER | *Ron has been programming 3D applications since before there were floating point processors on PCs. The rapid change of pace in 3D PC graphics has provided an area rich for creating new effects and techniques, where Ron's been consulting since 1994. He spent the last two years crafting real-time animated digital human faces for a failed VC startup (anyone want some technology?). When not programming 3D graphics he's teaching it through magazine articles and conference courses. His most recent book is* Real-Time Shader Programming *(Morgan-Kaufmann, 2002), a how-to book for shader writers. He can be reached at ron@directx.com.*

Lately, one of the biggest complaints from artists and developers has been that once you've got all the curved surfaces and textures you ever wanted, it's too difficult to give the scene that drop-dead gorgeous, oh-my-god-how-did-they-do-that look. In order to achieve a unique look, we're forced to do some crazy stuff. Brian Hook reported that QUAKE III could use up to 10 passes to render a scene with all the eye candy enabled. It's rumored that DOOM III can use up to 20. It seems like a lot of effort just to get some pixels set to a particular intensity and hue.

Luckily, there's been a lot of focus on how to bring more creative freedom to developers and artists. The graphics hardware vendors have taken a page from the successful model of Pixar's venerable RenderMan software and brought programmability to consumer graphics boards through the use of small programs called shaders that allow you to

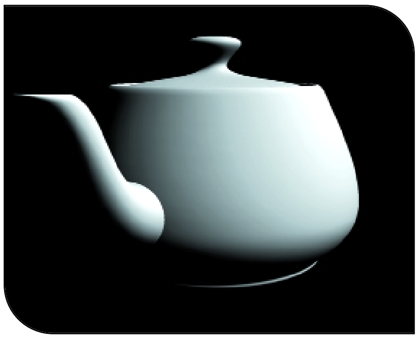


FIGURE 1. Traditional shading and its all-too-familiar diffuse-matte look.

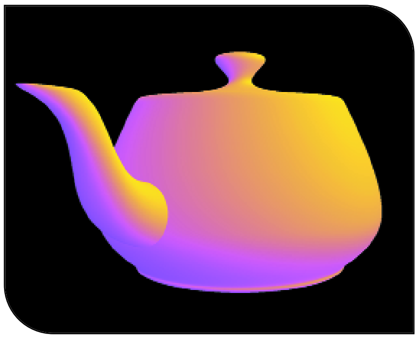


FIGURE 3. Gooch shading draws out details of shape and structure.

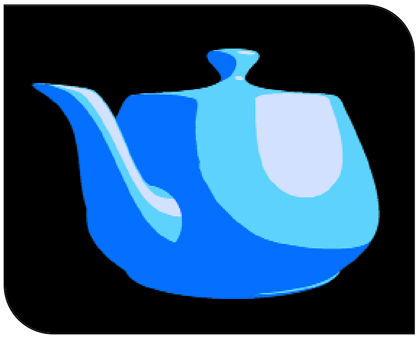


FIGURE 2. Cartoon shading using results from traditional lighting calculations.

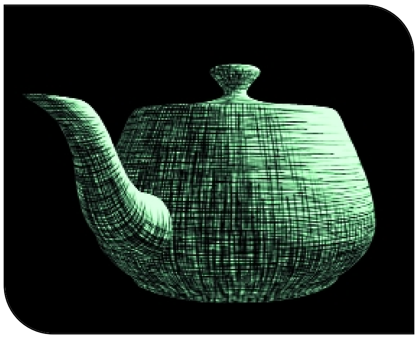
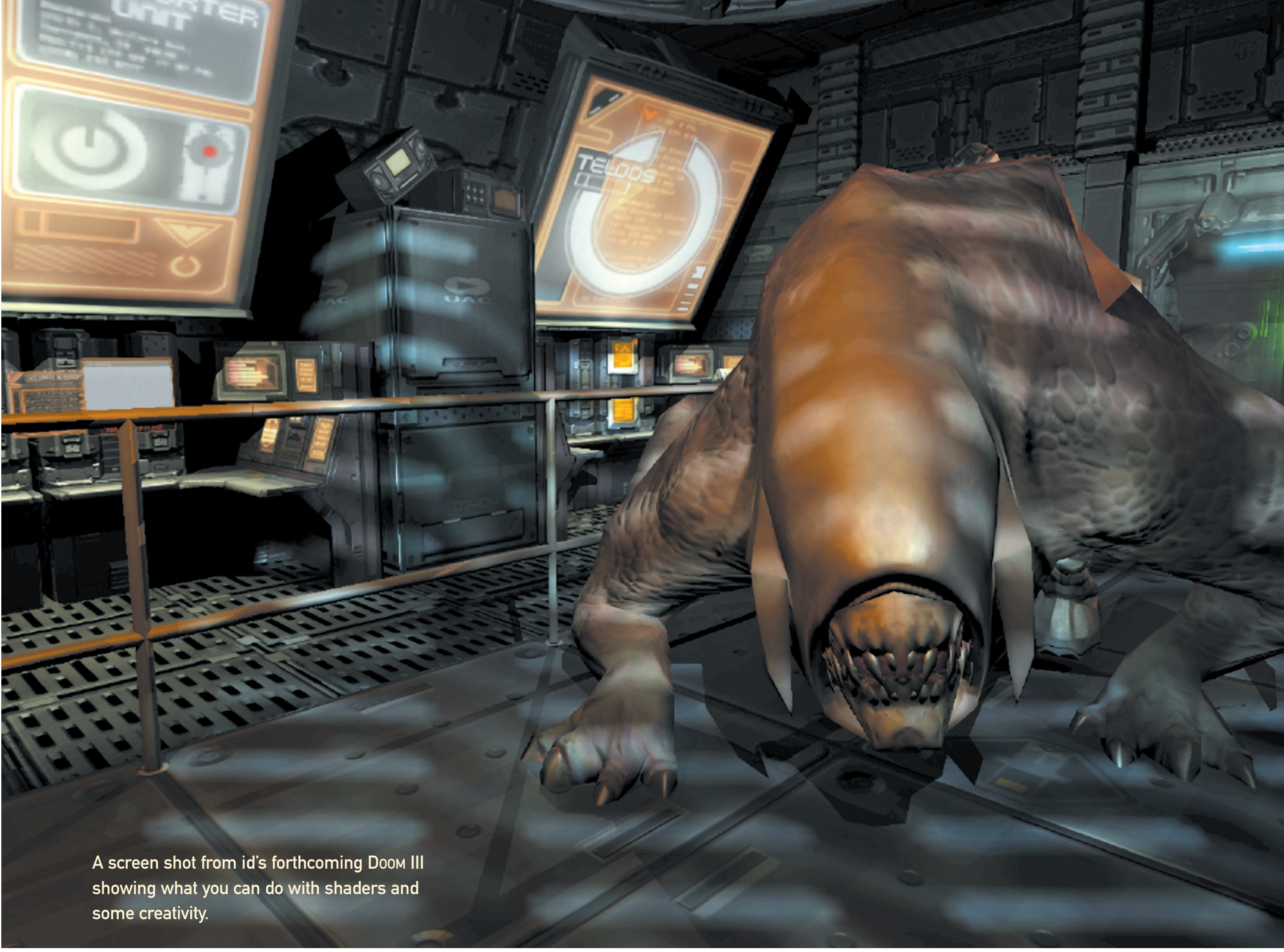


FIGURE 4. Hatching combines texture maps with pixel brightness for artistic effects.



A screen shot from id's forthcoming DOOM III showing what you can do with shaders and some creativity.

specify how vertices and pixels are actually output from the graphics processing unit (GPU). These were available last year with cards that supported DirectX 8.0 shaders, but it's finally reached critical mass this year with the latest crop of video cards. Unlike the transition to hardware T&L, using shaders involves a steeper learning curve.

My discussions with the various hardware vendors indicated that the effort of incorporating shaders into games is a strenuous one. Only a few of the cutting-edge developers are coming out with games that make use of shaders. The rest of them are busy getting ready to ship games for Christmas 2002, whose base technology predates that of the latest shader techniques. Many have discovered that implementing shaders is something more than your graphics guru can accomplish in a caffeine-fueled weekend. In fact, adding shader support to a game requires learning a new assembly-like language, learning how to check for various levels of shader support, treating shaders like any other resource file, and finally letting your artists have access to the shaders. It's no wonder that there's a dearth of shader-supporting games coming out this year.

If this sounds like a lot of work for a questionable return, you'll understand why the biggest changes in recent graphics hardware have focused on shaders. Shaders let you program the graphics pipeline using a low-level assembly language in DirectX 8, or a high-level language in DirectX 9's High-Level Shading Language (HLSL), Nvidia's Cg, and OpenGL 2.0's GL2.

In my previous article on hardware T&L, I talked about the speed increase that we'd gain from having the hardware and not the driver perform the texturing and lighting calculations for us. This meant that the graphics hardware took on some of the capabilities of the CPU. In fact, modern graphics hardware has the ability to perform complex math operations on multiple data in a SIMD fashion. You now have the ability to tell the hardware exactly how you want those calculations performed, and you no longer have to suffer with the matte-plastic-looking objects that the traditional pipeline gave you.

A Toon Teapot

Figure 1 shows the ubiquitous teapot model rendered by the traditional graphics pipeline. There's nothing surprising about it, it's got the diffuse-matte look that the traditional graphics pipeline is infamous for and that we're all thoroughly sick of. Shaders, by contrast, allow you to program the shading equations yourself.

Let's start off with a simple cartoon shader. The toon vertex shader computes the light intensity at each vertex (like the traditional graphics pipeline does), but it stores the light intensity as an encoded value. This value is then vertex-interpolated and passed to the pixel shader. The pixel shader then examines the intensity of the light and selects one of three color values representing the dark, illuminated, and brightly illuminated values. The colors can be part of the pixel shader code or can be variables. Programmability is part of the beauty of shaders. The cartoon teapot in Figure 2 uses traditional lighting calculations, but we took the results from those calculations to choose our cartoon-rendering scheme.

NPR

The real power of shaders is demonstrated by some of the non-photorealistic rendering (NPR) techniques with which developers are dabbling increasingly. There's a large number of these techniques (see Craig Reynolds' excellent listing of NPR links at www.red3d.com/cwr/npr), and we're starting to see many of these techniques show up as shader programs.

Gooch shading, shown in Figure 3, is designed for technical illustrations and is intended to clarify shape and structural details without cluttering up a rendering. It uses a warm and a cool color scheme to maintain edges and highlights.

Figure 4 shows an artistic hatching technique. The hatches are part of a carefully constructed set of texture maps that replace the brightness of a pixel with a selection of a texture representing that brightness. There are similar techniques for representing charcoal drawing, etching, dithering, and halftoning effects.

Procedural Shading

Some of the most creative shaders have been created by RenderMan users, and RenderMan has been the basis for much of the current crop of hardware shaders. However, the high-level languages that are currently being designed for real-time hardware shading have modifications based upon the limited interface with the hardware and, while they may look C-like, they aren't general-purpose programming languages. ATI has been busily replicating many of the shaders found in the RenderMan books, and you can find many examples on their web site (see For More Information).

The Tower of Babel: The Latest Shader Languages

Old-school developers who had to program in x86 assembly language may not find the low-level shader language that intimidating, but considering the dearth of games out there using DirectX 8 shaders, the low-level approach doesn't seem to be all that popular.

Now that the next batch of shader-capable hardware is making its way into the hands of consumers, it's time for the level of software support to catch up. All of the hardware manufacturers know that in order to differentiate themselves from the pack they'll need to stay in the lead. Their tried and true method for getting software to support their hardware features is to give game developers gobs of support to show them how easy it is to incorporate features into games that they may have been reluctant to put in before simply because they didn't have the resources.

Everyone realized that shader support would be a whole new ball game. Microsoft has always stated that their goal was to provide a high-level shading language that would make shader programming for PCs as easy as writing RenderMan shaders, namely C-like functions with a ton of built-in functionality.

Then Nvidia, a company never known for sitting back quietly when it could be doing something, preempted everyone by developing its own shading language/compiler called Cg. Some saw this as a strange move, since the Cg compiler currently only runs on Nvidia hardware (or hardware that supports Nvidia's extensions), and proprietary languages are not generally the path to market dominance these days. Nvidia told me that they saw developers' need for a higher-level language and wanted to provide a tool to fill that need.

Nvidia is still pushing Cg, even after Microsoft proposed its own language, HLSL (High-Level Shading Language). In addition, the OpenGL 2.0 draft specifies a shading language called GL2. Who wants to learn three shading languages? Could Nvidia be feeling some heat in the market? The press speculated that Nvidia's next-generation NV30 chipset missed some deadlines that were going to get it out in time for a significant presence this Christmas (at press time the NV30 was scheduled to be unveiled at Comdex in November), and in fact, the Cg presentations were the first public revelations of some of the NV30's capabilities, leading to a fair amount of early buzz.

I'm all for making life easier for programmers and artists, and Nvidia has been on the forefront of providing good tools to help game developers take advantage of the latest hardware, but historically their tools have been pretty agnostic. After the Cg announcement, though, Nvidia managed to embroil themselves in a small mêlée. ATI and 3DLabs jumped on Nvidia with both feet. Microsoft was a bit more politic, wondering why Nvidia didn't simply work with them.

The firestorm culminated with id Software's John Carmack weighing in just before the SIGGRAPH 2002 conference. 3Dlabs, who has been leading the OpenGL 2.0 spec, had sent Carmack a Wildcat VP board and an OpenGL 2.0 driver. It worked with minimal effort, and he was sold. Nvidia then sent John the Cg spec. There's really nothing too major that sets Cg apart from the other shading languages, since they are all in flux and still very similar at this stage. The Cg compiler was Nvidia specific, however, and hardware-specific code is a hobgoblin to most game programmers. Carmack stated that he wouldn't support Cg as it stands, since GL2 worked just fine for him.

In defense of Cg, it does provide the opportunity to write shaders that will work on both DirectX and OpenGL. Nvidia has open-sourced the parser and a back end to facilitate writing compilers for Cg, but there is a caveat: you'll need a compiler from each hardware manufacturer in order to produce optimized code for that hardware.

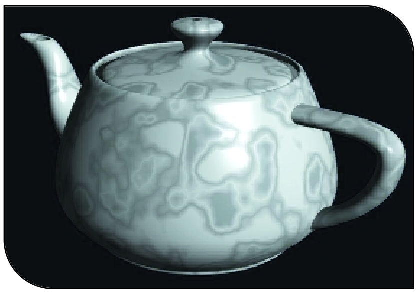


FIGURE 5. The RenderMan marble shader.

They've recently released a RenderMan shader compiler for RenderMonkey, their shader exploration tool, which lets you compile many RenderMan shaders (with some limitations). For example, Figure 5 shows the blue marble shader from Steve Upstill's *RenderMan Companion* (Addison-Wesley, 1990). Since there's no built-in noise function, they had to use a noise texture and perturb the texture coordinates, but it's basically the same shader. Listing 1 shows the actual code used in RenderMonkey to generate the image.

If you're familiar with traditional lighting calculations, you should have no problem figuring out this code; this listing should give you a good idea of what the higher-level shading languages look like. By the time you read this DirectX 9 should be out, and you will be able to download the SDK and try these out for yourself.

In conversations I had with people at ATI, they indicated they are happy with the code produced by the Microsoft HLSL compiler. After all, they said, Microsoft knows something about writing optimizing compilers. Their tests have shown that beta version of the compiler produces code for complex pixel shaders that's within three low-level shader instructions of the hand-optimized 35 instructions their engineers produced. That covers DirectX programs. The OpenGL solution for ATI is to use the GL2 shading language.

If you program for both OpenGL and DirectX, then a Cg implementation might be worth looking at. If you're monotheistic, it's still worth looking at Cg, if only for the educational value. There's a wealth of useful shader information on the Cg web site for programming in any shading language.

So what can you expect from a high-level shading language? For now, they all look similar. They are all C/C++-like and provide support for vertex and pixel/fragment shaders. The basic supported types are floats and float vectors, like you'd expect for SIMD architecture. High-level math functions such as dot products, vector normalization, and trig functions are built in. Swizzling and masking capabilities on variables are supported.

The one thing I haven't seen is the abstraction of the hardware dependence from the shader languages. It's possible to write a fairly complex shader (particularly using a high-level language) and discover that the compiled shader runs out of a particular resource (such as temporary registers). The only solution is to break the shader into pieces and render it as a multipass. This will force programmers either to have multiple versions for different hardware (the current awful situation), or to write to the lowest common denominator (an increasing likelihood as shaders become widespread). It would be much nicer to have the shader compiler figure out that it has to break up a shader and do it for you than to have to do this stuff by hand. In order for everyone to readily and easily take advantage of shaders in their games, this aspect of shader programming will have to be resolved.

The languages support shader files as external ASCII files, making interactive edit-load cycles possible for interactive shader editing. Microsoft's Visual Studio with the DirectX 9 SDK allows you to have a shader debugger, which is really handy when you're trying to figure out why a shader doesn't work right. DirectX 9 should be out just about the time this article hits the streets, so it will be interesting to see if Cg can find a home with Microsoft's HLSL and OpenGL's GL2.

— Ron Fosner

LISTING 1. The RenderMan marble shader used in RenderMonkey.

```
#define NNOISE 3
#define snoise(x)    (2*noise(x) - 1)

surface
bluemarble(
      float Ka = 0.1;
      float Kd = 0.8;
      float Ks = 0.3;
      float texturescale = 4.6;
      float ifreq = .106;
      float mfreq = 2.772;
      float scaleA = 0.148;
      float scaleB = .184;
      color ambientcolor = color(1);
      color specularcolor = color(1);
      float roughness = .01;
      color color0 = color (.93, .95, 1.0);
      color color1 = color (.98, 1.0, 1.0);
      color color2 = color (0.9, .9, 0.9);
      color color3 = color (0.85, .85, 0.85);
)
{

   color Ct;
   point NN;
   point PP;
   vector V;
   float i, f, marble;
   float scale;
   NN =  normalize(N);
   V  = -normalize(I);
   PP = P * texturescale;
   marble = 0;
   f = ifreq;

   for (i = 0; i < NNOISE; i += 1)
   {
      marble += snoise(PP * f);
      marble /= f;
      f      *= mfreq;
   }

   scale = scaleA * marble + scaleB;
   f  = clamp(scale, 0, 1);
   Ct = spline(f, color0, color1, color2, color3);
   Ci = Ct * (Ka * ambientcolor +
             Kd * diffuse(NN)  +
             Ks * specularcolor *
               specular(NN, V, roughness));
}
```

Roll-Your-Own Shading

Using shaders, it's possible to throw out the traditional model and craft something completely new. Figure 6 shows a detail of a car mirror and the different stages used to construct the final image. The base color is created from a combination of a diffuse color and two different specular colors, giving it the appearance of changing color at the middle of the specular highlight. A stage reproducing a metallic flake medium uses a highly specular term with a noisy texture map to perturb the light vector and give the appearance of embedded reflectors. Finally, a cubic environment map is applied to simulate a gloss finish. These three stages are applied in one shader pass, resulting in the finished image. The vertex and pixel shaders together constitute about 35 low-level instructions. As you can see, you get a lot out of those instructions.

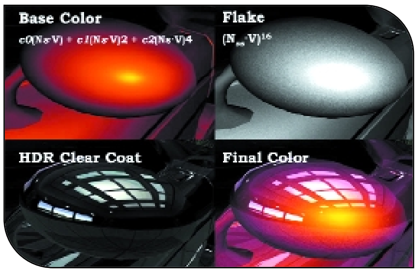


FIGURE 6. A three-color base coat, plus a flake specular, and a specular environment map, gives a two-tone metallic paint job.

The shader capabilities of the latest generation of hardware offer a great deal to developers, artists, and consumers alike. Nvidia is going to be pushing their NV30 architecture aggressively, pushing down the price point of their DirectX 9–capable cards to about $100 in early 2003. That's right, a card capable of playing DOOM III as it was meant to be seen for $100.

So if you've been sitting on the sidelines waiting for the time to be right to start implementing shaders for your game, now's the time to get out on the field. Download the SDK, brush up on how lighting equations work, and start thinking about how you can create your own unique shaders. There's a wealth of information out there to help you get started, so now that you've (hopefully) finished crunch mode and are ready to play, there's a whole new world of creativity waiting for you to discover it.

FOR MORE INFORMATION

Various hardware vendors have good resources on DirectX 9, OpenGL 2.0, and shaders:
www.microsoft.com/directx
http://developer.nvidia.com
http://mirror.ati.com/developer
www.3dlabs.com

There are also a couple of books:
Engel, Wolfgang. *Direct3D ShaderX.* Wordware Publishing, 2002.
www.shaderx.com
Fosner, Ron. *Real-Time Shader Programming.* Morgan-Kaufmann, 2002.
www.directx.com

Picking over the Current Hardware Crop

While ATI and Nvidia are the current leaders in the graphics hardware arena, 3Dlabs (given new life through its acquisition by Creative Technology) and Matrox are following about six months behind. Even on the low-end consumer level we've got cards like the SiS Xaber 400 that come with shader programmability, so it's a mix of players right now. The manufacturers listed in the table at right represent more than 90 percent of the current market. All the leading cards from these manufacturers come with at least DirectX 8 capabilities, and by Christmas 2003 many games should be taking advantage of shaders.

The two leading video cards come from Nvidia and ATI. As I write this Nvidia's NV30 hasn't shipped yet, but its current specs make it more impressive than ATI's already shipping Radeon 9700 line. Both fully support DirectX 9 shaders. 3Dlabs' Wildcat VP and Matrox's Parhelia are hybrids, supporting DirectX 9 vertex shaders but only DirectX 8.1 pixel shaders.

Matrox has also gone off on a bit of a tangent from the rest of the graphics chipset manufacturers with support for three simultaneous monitors, compared to two for the rest of the herd. While this might seem strange at first, it makes a weird bit of sense when you consider that all you need to support it is an adjustable field-of-view (FOV) parameter in your game, and realize that QUAKE III, UNREAL, and a few others already have an adjustable FOV. This produces some perspective problems at the edges, since it's not a mathematically correct thing to do, but no one seems to be complaining very loudly.

I've included SiS's Xabre as well, since it's representative of those sub-$100 cards that fully support DirectX 8 shaders.

— Ron Fosner

	Nvidia NV30	ATI Radeon 9700	3Dlabs Wildcat VP	Matrox Parhelia	SiS Xabre 400
AGP Bandwidth	8x	8x	4x	4x	8x
Memory	NDA*	128/256MB	64/128MB	128MB	64MB
Bus Width	DDR2	256-bit DDR	256-bit DDR	256-bit DDR	128-bit DDR
DirectX Support	DirectX 9	DirectX 9	DirectX 8.1+	DirectX 8.1+	DirectX 8.1
Pixel Pipes	NDA*	8	8	4	4
Pixel Shader Version	2.0+	2.0	1.2 with partial 2.0	1.3	1.3
Vertex Shader Version	2.0	2.0	1.1 with partial 2.0	2.0	1.1
Simultaneous Textures	NDA*	8	8	4	4
Floating Point Pixels	Yes	Yes	No	No	No

*Still under NDA at press time.

And Presto... It's Gone!

The Final Days of Presto Studios: An Insider's View

COMPANY DATA

WHERE BASED: San Diego, CA

DATE FOUNDED: Incorporated in 1992

DATE CLOSED: Official final day October 15, 2002

HEADCOUNT AT FOUNDING: 8

HEADCOUNT AT CLOSING: 21

PEAK HEADCOUNT: 45 in 1999 during our attempt at three projects

SOFTOGRAPHY:
THE JOURNEYMAN PROJECT; completed in 1993
THE JOURNEYMAN PROJECT 2: BURIED IN TIME; completed in 1995
GUNDAM 0079: THE WAR FOR EARTH; completed in 1996
THE JOURNEYMAN PROJECT: PEGASUS PRIME; completed in 1997
THE JOURNEYMAN PROJECT 3: LEGACY OF TIME; completed in 1998
THE JOURNEYMAN PROJECT 4; cancelled at conceptual stage in 1998
BENEATH; cancelled in the final year of production, 1999
STAR TREK — HIDDEN EVIL; completed in 1999
STEPHEN KING'S F-13; completed in 2000
MYST III: EXILE; completed in 2001
WHACKED!; completed in 2002

Many people in the game industry grew up following the careers of those who led the first wave. Those mavericks from the late 1970s and early 1980s defined the punk rock, garage band atmosphere that fed our dreams of someday doing the same thing: working with a group of friends in a clubhouse for little money, pushing for that big payday when a man with a bag full of cash walks in the door and makes all your Ferrari-owning fantasies come true.

Now that I've been in the industry for nearly 10 years, I've developed a more realistic view of the business. Why do most small startups never make it? What mistakes are made? Is it inevitable that most small developers will eventually fail due to economic reasons beyond their control, even if they make all the right moves? Through the lens of Presto Studios' recent demise, I will look at the state of boutique development houses, examine the industry as it has evolved over the previous two decades and find how we all fit into it, now and in the future.

Presto Studios was started 11 years ago by a group of old friends from high school and college, along with their respective families. It started during the introduction of CD-ROMs which ushered a new era of photorealistic graphics. THE JOURNEYMAN PROJECT, Presto's first product, developed into a profitable intellectual property, supplying the bread and butter for the first half of this small company's existence. While often taking second chair to the hugely successful MYST/RIVEN license, Presto enjoyed a rabid fan base and became known as one of the premier art houses in the business, attracting an excellent collection of prerender artists.

Phase Two

A TOMB RAIDER–killer called BENEATH marked the company's push into real-time 3D and the beginning of the second stage for Presto. Along with that ambitious project, we scored a *Star Trek* adventure title under Activision that showcased our prerender talent. In addition to these two projects, a prototype team was assembled to explore even more opportunities, marking Presto's first promising foray into multiple projects. Unfortunately, the company stumbled at this point and only the STAR TREK game was released, winding us back to being a single-project house.

The remaining three years saw Presto returning to its roots to produce MYST III: EXILE. Released in May 2001, it was the sequel to Cyan's original MYST and

MICHAEL SALADINO | *Michael has been evolving over the last nine years from a low-level graphic guru into a technical director and game producer. With the closing of Presto, he is now spending his time searching for a new home for his purple velvet chaise lounge and wine bar. He can be reached at mikeyspeakeasy@yahoo.com.*

RIVEN, its follow-up. WHACKED, recently released for the Xbox, was our final attempt at real-time. However, these titles were essentially produced one at a time. On August 26 of this year, we announced to our employees and the world that WHACKED would be Presto Studios' last product.

I should point out that I joined the company during its second phase, in which it was developing its own internal real-time 3D engine. I am not a founder or an owner of the company; however, the details in this article from before my time come from the people that were there. My view from the trenches also excluded me from inside information about the financial dealings that kept Presto running for over a decade. Therefore, I won't attempt to expound too much on the financial state of Presto, but instead I'll focus on the internal development cycles and the publisher relationships I witnessed firsthand.

What Went Right

1. Focus on great art. Unlike most game companies, Presto Studios started as a storytelling enterprise. Because they are less about the game and more about the visual emersion, graphic adventures have a unique position in our industry. Despite the gameplay derived from their puzzles, they are very much digital coffee table books for users wanting to see something pretty. With this being the case, Presto began developing a name as an "artist's" game company. We were never at a loss for résumés from talented young artists around the world, and many of our team members have moved into film work at companies such as Pixar and Lucasfilm.

2. Unique, creative high concepts that pushed innovation. The halls of Presto were filled with awards and trophies. We delivered high-concept art that was at the top of the industry. But one of our biggest problems when trying to get contracts was the talent of which we were most proud. Financially conscious publishers tend to handle new or unusual ideas with skepticism. While I find this fact unfortunate, I understand why it happens.

The game industry is like all other businesses, here to make money for ourselves, our team members, and our financial partners. Experimental movie projects are often passed over for summer blockbuster action flicks starring everyone's favorite megastar, and the same holds true for games. It's not true that only bad, derivative works are published, and it's not true that only breakthrough games can be fun. If you want to stay in the business relying solely on risky ventures, however, you're in for a long road with your publishers. Therefore, we played both sides, building both license games and sure-thing sequels as a way to get permission to do unique work later.

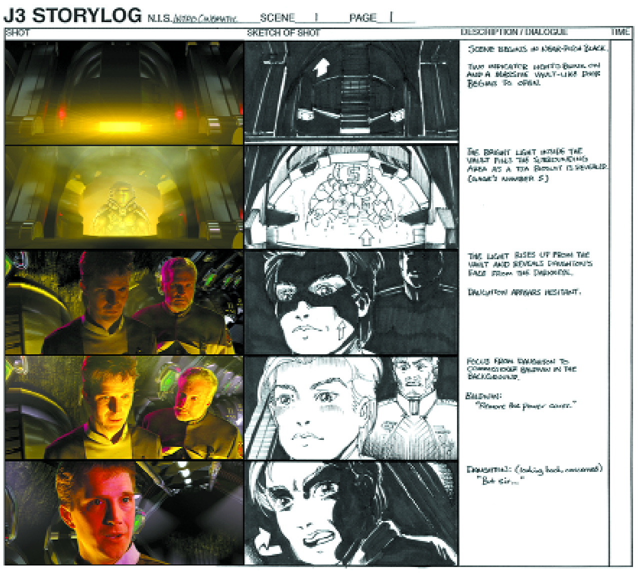
3. Built strong niche position with our own IP. Regardless of the industry, if you possess intellectual property of any value, it should be leveraged (or milked) for everything it's worth. Even if you don't own the IP, if you're the studio that created it, you should at least get first right of refusal to subsequent products. Presto Studios understood this with THE JOURNEYMAN PROJECT. Award-winning and financially successful (along with the original, two sequels and numerous ports were spawned across the Macintosh, PC, and Playstation platforms), it helped Presto establish its name as a premier studio for graphic adventure titles, bringing together people who were experts in this craft. Michel Kripilani, one of Presto's founders, honed his ability to schedule complex resource trees for both cutscenes and in-game flow. Phil Saunders, the creative director, held all his projects to an extremely high level of quality, a major reason for our development into such a strong art house. The foundation was thus laid for us to be the definitive graphic adventure house in the industry.

4. MYST III. When Cyan wanted to farm out the third installment of what was then computer gaming's best-selling franchise (before being surpassed by THE SIMS), Presto's reputation and experience gave us an incredible chance late in our history to develop MYST III: EXILE. Cyan's search for a company that could deliver to their level of quality brought them to Presto Studios in 1999.

One of my first responsibilities at Presto was creating the prototype for pitching the game. Using the Sprint engine and its hardware and software rendering capabilities, I programmed a mini-adventure in which you could smoothly look around inside 3D nodes. The nodes were cubes with prerendered images textured onto them with the camera located at the center. This created a powerful "you are there" feel that helped push the experience beyond what even Cyan was familiar with. The project turned into the biggest-selling game in Presto's history and in many ways represented the pinnacle of our craft.

5. Leapfrogging into real-time 3D at the right time. The first game created by Presto Studios was "programmed" in Director, so it was natural that for the first half of its existence no one saw technology as a primary focus. In the background of the Presto logo there are four words: animation, interactivity, video, and music. This mantra really defined the CD-ROM, multimedia mentality that spawned the company during the early 1990s, a mentality that did not include advanced game programming. So when the decision was made to move away from the dying graphic adventure market and into the mainstream arena of real-time 3D, a major retooling was required.

This rebirth came in 1997 in the form of Max Elliott and his Sprint engine. An agreement brought Max into Presto Studios as the CTO along with all the technology that he had developed over the previous two years at his own company, Sibling Interactive.


J3 STORYLOG N.I.S. INTRO CINEMATIC SCENE 1 PAGE 1
SHOT
SKETCH OF SHOT
DESCRIPTION / DIALOGUE
TIME
SCENE BEGINS IN NEAR-PITCH BLACK.
TWO INDICATOR LIGHTS BLINK ON AND A MASSIVE VAULT-LIKE DOOR BEGINS TO OPEN.
THE BRIGHT LIGHT INSIDE THE VAULT FILLS THE SURROUNDING AREA AS A TIA ROBOT IS REVEALED. (CAGE'S NUMBER 5)
THE LIGHT RISES UP FROM THE VAULT AND REVEALS DAUGHTON'S FACE FROM THE DARKNESS.
DAUGHTON APPEARS HESITANT.
FOCUS FROM DAUGHTON TO COMMISSIONER BALDWIN IN THE BACKGROUND.
BALDWIN: "Remove the power cover."
DAUGHTON: (looking back, concerned) "But sir..."

THE JOURNEYMAN 3 STORYLOG. THE JOURNEYMAN PROJECT 3 storyboards translating into the final product.

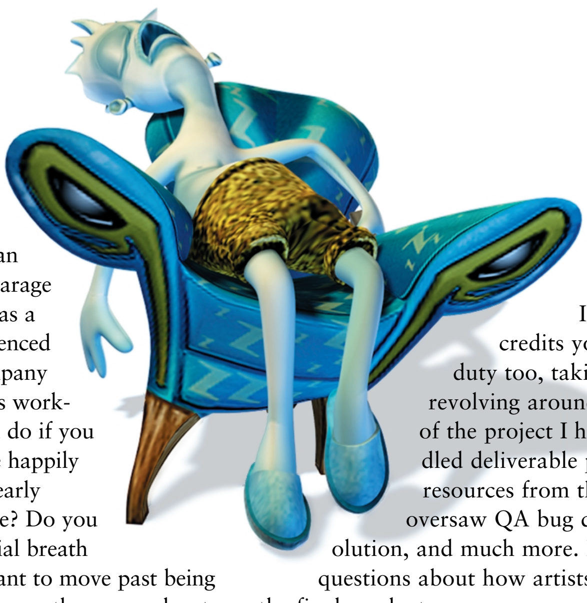
The use for this new technology was already decided: BENEATH, a third-person action-adventure done in an early 20th-century Jules Verne style. The Sprint engine came prepared with a full software renderer and a Glide hardware renderer. Other features included a sound system based on Miles, complete physics and collision systems, and plenty of general-purpose objects such as object emitters and switches. The package also included a tool path that allowed the construction of levels and characters inside 3DS Max. This suite of technology — along with the leadership of Max Elliott — transported Presto quickly from the stone age of game technology right up to the cutting edge.

What Went Wrong

We have a small startup, we have great art, we have a cutting-edge engine, and we have truly unique concepts. What could go wrong? Well, something did, because I'm sitting here writing Presto's postmortem.

1. Friends and family: Nice people. Maybe too nice. There is a critical problem with the "garage-band" game studio: these studios are usually started by a group of friends and family. This is a natural course; who else beyond such close associates is going to work that first year or two for next to nothing? Who else will moonlight (for free) for the chance to live the dream that they've had since they were 10 years old, playing BEACH-HEAD on their Commodore 64? This business model (for lack of a better term) is completely valid at the conception of a new company.

However, once the initial success is achieved, which for Presto came in the form of the JOURNEYMAN IP and several successful titles, the new company finds itself with money in its coffers for the first time. Ramping up the production staff is a necessity, and you're able to attract experienced people from the industry. People send their résumés to you in addition to Electronic Arts and id. The company evolves to a higher level of professional-


Otto, a contestant in Presto's WHACKED, plays the game from the confines of his fully articulated comfy chair.

ism, and the new staff reflects that in their skill levels and their salaries.

Unfortunately, these new people being hired might be more experienced than the founders. The modeler from the first garage project that helped start everything now has a team of modelers who are all more experienced and skilled at their craft. How does a company resolve these issues? Do you have founders working beneath new employees? What do you do if you discover your good friend, even though he happily pulled all-nighters on his own to support early efforts, can't manage a team to save his life? Do you let go those that gave the company its initial breath of life if they can't keep up? If you ever want to move past being a clubhouse and become a successful company, the answer has to be yes. At Presto, the answer was no.

2. Who wants to be the producer? Now, who wants to be the CEO? Presto suffered from its founders being placed in positions of authority and responsibility when they hadn't yet learned the skills that would allow them to perform the job well. Many of them wanted to be game designers, so they made themselves producers, which experienced people will realize is not the same job.

Most people entering the industry dream of being a game designer and don't even consider what a producer does, which can be described as tedious at best. Without any real production experience at the top of the company, Presto's projects were often lacking in direction and a fundamental understanding of the steps needed to create the game. These flaws were most evident in our forays into real-time 3D.

There was one exceptionally capable producer at Presto Studios, Michel Kripilani. Over the four years I was at Presto, I witnessed his skills improve with each project. Unfortunately, his business card didn't read "producer," it read "CEO." His purpose was to run the company day-to-day and procure new projects for future development. The topmost reason Presto Studios closed its doors was that we did not have a signed project when WHACKED was finishing up. The idea of going into debt to finance prototypes along with the inevitable layoffs that would accompany another lean time made closing feel like the better option.

Funneling his leadership skills into marshaling game production along kept Michel from focusing on his CEO role. He was constantly being brought back into the trenches to fill a hole at the top of any given project. On MYST III, he took on the responsibility of producing the theatrical trailer. On WHACKED, he was a hands-on executive producer through nearly the entire project, and at the end referred to himself as the "emergency online producer" for the Xbox Live portion of the game. While his work on the project was invaluable, he should have been able to hand these duties over to one of the numerous producers at the company, or as another possibility, return to producing full-time and hire a CEO to acquire new projects.

If you look in the WHACKED credits you'll find me pulling double duty too, taking on a producing role revolving around the technical understanding of the project I had as lead programmer. I handled deliverable preparation, managed resources from the art team into the game, oversaw QA bug distribution and early-out resolution, and much more. I was the point man for most questions about how artists' work would make it into the final product.

I thrived on producing during the day, while other people were around, and then programming on my own through the night. However, this was not a great way of handling the workload. Performing as a producer came at the expense of my primary role as lead programmer. In that way, Michel and I are cut from the same cloth. We both wanted to handle it all, which led to deficiencies on all sides.

3. Failure at multiple-project development due to gaps in management. Due to a lack of managerial skills, the company was never able to grow successfully into a multiple-project development house. Our one attempt saw three signed projects across three teams. The primary project, BENEATH, was the company's first real-time 3D game using the Sprint engine. Activision cancelled the project in its later stages, officially because milestones were slipping and similar products with close launch dates were adding to the risk of low sales. However, I believe that another reason was that the game's "fun" had never been fully designed. Our art department created vast stretches of beautiful levels, but the gameplay lagged months behind, with most of the levels never being set up for AI. It was enormous, it was beautiful, but it wasn't fun; an obvious breakdown at the producer level.

Meanwhile, the second team was working on another installment of THE JOURNEYMAN PROJECT, which was supposed to be the first done in real-time 3D. This project didn't make it past the initial conceptual production phase before it was cancelled. The third project, STAR TREK — HIDDEN EVIL, for which I was brought on as lead programmer halfway through development, was also plagued by runaway design from a team with an incomplete understanding of the tools at their disposal. Massive rewrites of the game flow were done in the last third of the project, leading to a schizophrenic gameplay experience.

I've met very few people in my career that possess a natural ability to manage large teams; it's something that is best learned by working with another great producer. But Presto Studios never hired that great producer. We had incredible mentoring in our art department, hiring promising rookies to work alongside our proven art leads. Under Max Elliott's leadership, I came up along a similar path in the company's engineering department. However, we had nothing like this for our producers. They were mostly founders or friends of founders, none of whom had ever worked in a game producer capacity before. The lesson was never learned.

Presto's inability to transition successfully to multiple projects also incurred a less obvious expense that hampered the company financially: the cost of our internal Sprint engine. As a programmer and one of the chief architects of Sprint, I was a strong proponent of owning our own technology. Keeping control of your own technology and creative IP is one of the only ways a small boutique firm will ever survive. However, keeping Max and me on the payroll to develop the engine was expensive. Distributing that expense across multiple projects was an essential cost-saving measure that we never achieved. Instead, with us producing titles sequentially, licensing middleware solutions would have been a more appropriate solution.

4. Layoffs, walkouts, canned projects, and uncertain future led to low morale. After a dismal year in 1999 that included layoffs, a mass exodus, and two cancelled projects, we retreated to our roots with the successful MYST III. We also finally got real-time 3D right with WHACKED for the Xbox; however, these two projects were still essentially done sequentially, with another round of layoffs and departures separating the two. And on multiple occasions, we suffered mass walkouts as people lost faith in the company's ability to acquire our next project.

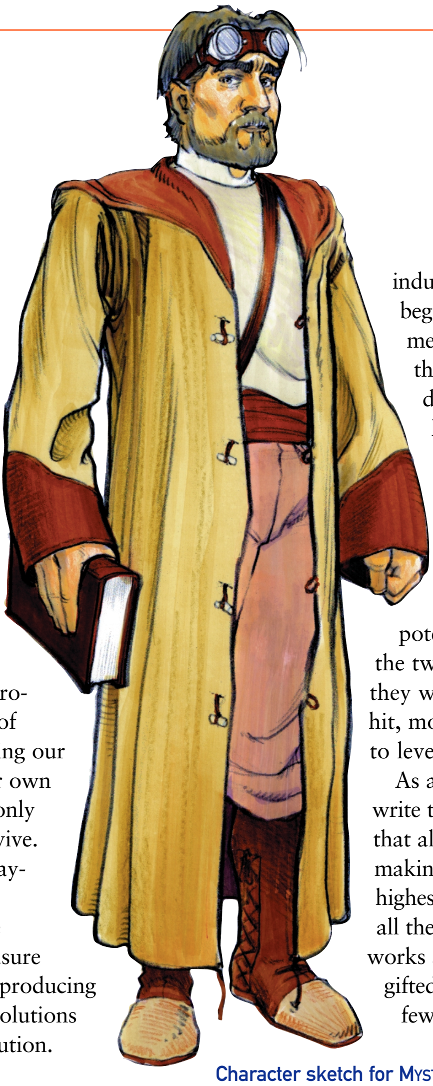
These high turnover rates then affected the morale of the people who remained, leaving them to wonder if they were next. At three different times during my four years at Presto, I found myself sending out résumés, not because I wanted to leave but because I was certain that this time the company wouldn't be able to pull out of the dive. I had sent out résumés one week before the closing of Presto was decided.

5. Presto's offbeat style got riskier as the industry evolved. Despite the internal problems, external changes in the industry also made Presto Studios a riskier venture for publishers. Witness the evolution of other industries such as movies and music. In their infancy, dozens or even hundreds of small startups littered the field, each struggling to carve out a share of the marketplace. Many in the beginning are too idealistic about the business surrounding their personal art, so industrial Darwinism takes hold as the large players begin to emerge. The small companies are then merged, purchased, or run out of business due to the economy of scale. This promotes the consolidation of the industry under a few major labels. It happened to the music and movie publishing industries, and it's currently happening in the game industry.

Reducing risk is where IP ownership makes a major play. If you can retain control of your IP and it becomes successful, you gain massive leverage when dealing with potential publishers. When a publisher is lining up the two dozen titles they want to release next year, they want to see sure things: sequels to last year's big hit, movie licenses, top-40 music tie-ins. They want to leverage what the public already knows and loves.

As a result, original content is on the decline. I don't write this to sound like a sensitive artist who believes that all great work comes from poverty and all money-making media are soulless. I believe that most of the highest-earning games really are that good and deserve all the success they achieve. However, truly inspired works are sometimes passed over because the people gifted at recognizing opportunity in this business are few and far between at most publishers.

As the industry becomes larger and more centralized around a couple of companies, breakout ideas will become rarer. As is happening in the music and movie industry, an independent subculture will hopefully continue to develop within the game industry and help pull the larger companies into the "next big thing."


Character sketch for MYST III's Altrus.

A Familiar Story with a Now Familiar Ending

The fall of Presto Studios is probably a familiar story for many of you. Working there was wonderfully creative, insanely demanding, and always fun, even at the worst moments.

Despite any of the negatives, trying a startup is something that I highly recommend for anyone that wants to test their own abilities. Most of us want that opportunity to execute on an idea that we've had for a long, long time. Just remember that paying your dues is a prerequisite to that opportunity. Maybe you will work on a few licensed products that offend your independent creativity because your publisher wants a sure thing. It's only by building this trust between your small boutique house and that big publisher that you'll ever get the chance to work on your dream project.

Working in this industry now and in the future will demand that each of us find our own comfortable balance between the financial realities of our growing industry and the artistic drive that brought us here from our childhood dreams.

Return

The Real USP

At some point while you are presenting your latest and greatest game idea, somebody (usually your publisher or product manager) will ask what your USPs are. It seems like a reasonable question at first, but it's not. Whether they are asking this because they want to assert their position, because they want to appear knowledgeable, or because they want something to pass word for word up the chain, they shouldn't expect an answer — the question is nonsense. Why? Good question.

Using acronyms can save you time in technical conversations, but more commonly people use them in nontechnical conversations to fast-talk others into accepting their views. This becomes obvious when such a person uses an acronym incorrectly. The publisher's use of the acronym "USP" in the situation I just described is a blatant example. GameDev.net's Game Dictionary neatly sums up this widely accepted (but mistaken) definition of the USP: "Unique Selling Points. Normally what will be put on the back of a box or an advertisement showing how a game is different and better than its competitors and predecessors."

How am I sure this definition is wrong? I did some research. USP stands for "unique selling proposition," a phrase first coined by Rosser Reeves, an adman and chairman of the board at Ted Bates & Company, where the technique was invented. He published a book in the early 1960s called *Reality in Advertising*, in which he explained the USP and many other principles developed over the previous 20 years at his company. He also published the research that backed up these techniques. His purpose was to evolve a body of theory based on collected evidence. Here is his definition of the USP from *Reality in Advertising*:

"Each advertisement must make a proposition to the consumer. Not just words, not just product puffery, not just show-window advertising. Each advertisement must say to each reader: 'Buy this product and you will get *this specific benefit*.'

"The proposition must be one that the competition either cannot, or does not, offer. It must be unique — either a uniqueness of the brand or a claim not otherwise made in that particular field of advertising.

"The proposition must be so strong that it can move the mass millions, i.e., pull over new customers to your product."

Now when you apply the real definition of USP to the question "What are your USPs?" you should notice two problems. First, you only make one proposition, so the question should be, "What is your USP?" And second, if you need to ask, then you don't have one. It's that simple.

Apart from exposing a bunch of publishers and product managers as the charlatans we knew they were, what does all this have to do with your game? Well, before it became a mere buzzword to beat game developers with, the USP was a powerful advertising technique. If it had not been, it's unlikely the term would still be in common (but mistaken) use half a century later. And the USP, like many things, can work for or against you.

To understand how a USP can increase or decrease your sales, you first have to measure the effectiveness of your advertising. You can't just measure the sales. Sales can go up and down for dozens of reasons, only one of which is your advertising.

There are two statistics we need to know to measure a marketing campaign's effectiveness. The first is penetration, the percentage of people who saw and can remember your campaign. The second is usage-pull, the difference between the percentage of people who can remember your campaign and bought your game, and the percentage of people who can't remember your campaign but still bought your game. Combining these two measurements gives you your campaign's overall effectiveness.

Although the penetration score cannot become negative (the worst you can do is 0 percent), the usage-pull can. If people who don't remember your campaign buy more copies than those who do, your penetration statistics will reverse, and you will sell more copies if fewer people can remember your campaign. A penetrative USP can actually decrease your sales. But even that is not the worst of your worries.

Worse comes if you can't distinguish your game in the marketplace. For example, if you have a driving simulation game, you cannot use "simulating the driving experience" as your USP because the GRAN TURISMO series already uses "the real driving simulator." If you try, you will actually spend money to lower your own sales and boost the sales of your direct competition.

Illustration by Steve Munday

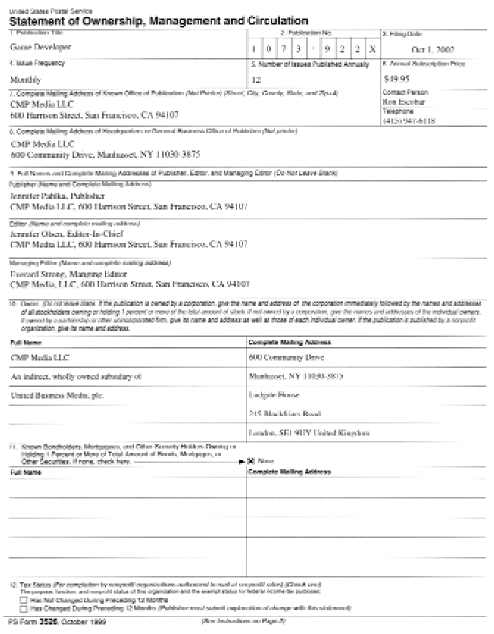
continued on page 55

continued from page 56

Having a real USP lessens this risk. By definition, a USP distinguishes your game in the marketplace. By contrast, listing the "unique" features of your game on the back of the box does the opposite. Only one game proposes "tactical espionage action" (METAL GEAR SOLID), but any number of games offer "unrivaled realism," "an epic tale," or "superior artificial intelligence."

The USP is a powerful selling technique, and there's a lot more to it than what's covered here. But I hope I have pointed the way for those who want to know more. You may not be in a position to sell your games direct to your customers, but you will have to pitch your ideas to your publisher or producer, and they will have to sell the same idea to their marketing teams, and so on. The next time somebody asks you what your USPs are, fight the battle against buzzwords. Getting it right will put you back on your front foot, and down the line it may save you some sales too.

PAUL SINNETT | *Paul is a game programmer at Coyote Developments Ltd. He has been writing games since 1996.*



United States Postal Service
Statement of Ownership, Management and Circulation

1. Publication Title: Game Developer	2. Publication No.: 1 0 7 3 - 9 2 2 X	3. Filing Date: Oct 1, 2002
4. Issue Frequency: Monthly	5. Number of Issues Published Annually: 12	6. Annual Subscription Price: $49.95
7. Complete Mailing Address of Known Office of Publication (Not Printer) (Street, City, County, State, and Zip+4): CMP Media LLC, 600 Harrison Street, San Francisco, CA 94107		Contact Person: Ron Escobar; Telephone: (415) 947-6118

8. Complete Mailing Address of Headquarters or General Business Office of Publisher (Not printer)
CMP Media LLC
600 Community Drive, Manhasset, NY 11030-3875

9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor (Do Not Leave Blank)
Publisher (Name and Complete Mailing Address)
Jennifer Pahlka, Publisher
CMP Media LLC, 600 Harrison Street, San Francisco, CA 94107
Editor (Name and complete mailing address)
Jennifer Olsen, Editor-In-Chief
CMP Media LLC, 600 Harrison Street, San Francisco, CA 94107
Managing Editor (Name and complete mailing address)
Everard Strong, Managing Editor
CMP Media, LLC, 600 Harrison Street, San Francisco, CA 94107

10. Owner (Do not leave blank. If the publication is owned by a corporation, give the name and address of the corporation immediately followed by the names and addresses of all stockholders owning or holding 1 percent or more of the total amount of stock. If not owned by a corporation, give the names and addresses of the individual owners. If owned by a partnership or other unincorporated firm, give its name and address as well as those of each individual owner. If the publication is published by a nonprofit organization, give its name and address.)

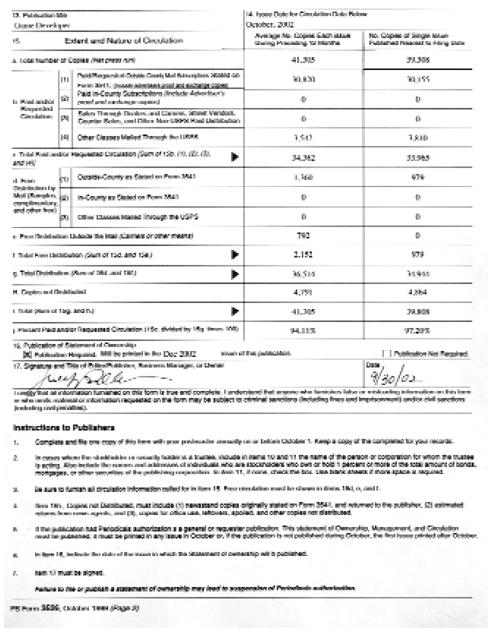
Full Name	Complete Mailing Address
CMP Media LLC	600 Community Drive
An indirect, wholly owned subsidiary of	Manhasset, NY 11030-3875
United Business Media, plc.	Ludgate House
	245 Blackfriars Road
	London, SE1 9UY United Kingdom

11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities. If none, check here. ► [✓ None]

Full Name	Complete Mailing Address

12. Tax Status (For completion by nonprofit organizations authorized to mail at nonprofit rates) (Check one)
The purpose, function, and nonprofit status of this organization and the exempt status for federal income tax purposes:
☐ Has Not Changed During Preceding 12 Months
☐ Has Changed During Preceding 12 Months (Publisher must submit explanation of change with this statement)

PS Form **3526**, October 1999 (See Instructions on Page 2)



13. Publication Title: Game Developer		14. Issue Date for Circulation Data Below: October, 2002	
15. Extent and Nature of Circulation		Average No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total Number of Copies (Net press run)		41,305	39,308
b. Paid and/or Requested Circulation	(1) Paid/Requested Outside-County Mail Subscriptions Stated on Form 3541. (Include advertiser's proof and exchange copies)	30,820	30,155
	(2) Paid In-County Subscriptions (Include Advertiser's proof and exchange copies)	0	0
	(3) Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Non-USPS Paid Distribution	0	0
	(4) Other Classes Mailed Through the USPS	3,542	3,810
c. Total Paid and/or Requested Circulation (Sum of 15b. (1), (2), (3), and (4)) ►		34,362	33,965
d. Free Distribution by Mail (Samples, complimentary, and other free)	(1) Outside-County as Stated on Form 3541	1,360	979
	(2) In-County as Stated on Form 3541	0	0
	(3) Other Classes Mailed Through the USPS	0	0
e. Free Distribution Outside the Mail (Carriers or other means)		792	0
f. Total Free Distribution (Sum of 15d. and 15e.) ►		2,152	979
g. Total Distribution (Sum of 15c. and 15f.) ►		36,514	34,944
h. Copies not Distributed		4,791	4,864
i. Total (Sum of 15g. and h.) ►		41,305	39,808
j. Percent Paid and/or Requested Circulation (15c. divided by 15g. times 100)		94.11%	97.20%

16. Publication of Statement of Ownership
[✓ Publication Required.] Will be printed in the Dec 2002 issue of this publication. ☐ Publication Not Required.

17. Signature and Title of Editor, Publisher, Business Manager, or Owner
[illegible]
Date 9/30/02

I certify that all information furnished on this form is true and complete. I understand that anyone who furnishes false or misleading information on this form or who omits material or information requested on the form may be subject to criminal sanctions (including fines and imprisonment) and/or civil sanctions (including civil penalties).

Instructions to Publishers

1. Complete and file one copy of this form with your postmaster annually on or before October 1. Keep a copy of the completed form for your records.
2. In cases where the stockholder or security holder is a trustee, include in items 10 and 11 the name of the person or corporation for whom the trustee is acting. Also include the names and addresses of individuals who are stockholders who own or hold 1 percent or more of the total amount of bonds, mortgages, or other securities of the publishing corporation. In item 11, if none, check the box. Use blank sheets if more space is required.
3. Be sure to furnish all circulation information called for in item 15. Free circulation must be shown in items 15d, e, and f.
4. Item 15h., Copies not Distributed, must include (1) newsstand copies originally stated on Form 3541, and returned to the publisher, (2) estimated returns from news agents, and (3), copies for office use, leftovers, spoiled, and other copies not distributed.
5. If the publication had Periodicals authorization as a general or requester publication, this Statement of Ownership, Management, and Circulation must be published; it must be printed in any issue in October or, if the publication is not published during October, the first issue printed after October.
6. In item 16, indicate the date of the issue in which this Statement of Ownership will be published.
7. Item 17 must be signed.

Failure to file or publish a statement of ownership may lead to suspension of Periodicals authorization.

PS Form **3526**, October 1999 *(Page 2)*