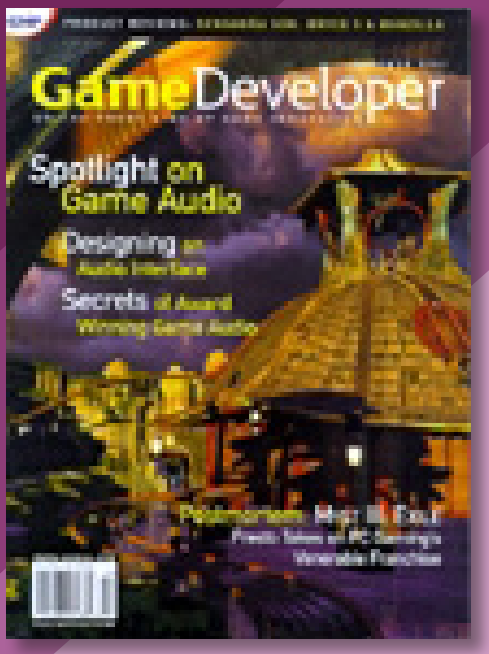
gd

GAME DEVELOPER MAGAZINE

OCTOBER 2001



UBM Tech

GAME PLAN

LETTER FROM THE EDITOR

Insert Ad Here

In the beginning, there were mainframes. And SPACE WAR. Then came arcade games, and COMPUTER SPACE. TV game systems, and PONG. Soon thereafter came home game consoles, electronic toys, watches, personal computers, key chain fobs, pedometers, PDAs, cell phones, and so on, all of which you can play games on.

Nowadays, everywhere that you find a microprocessor, you'll find that someone has created a game to play on it. Over time, the range of capabilities of these microprocessors has grown. And so has the range of games that you can play. You can buy cheap key chain fobs that play TETRIS, cell phones that play SNAKE, PDAs that play MINESWEEPER, and personal computers that play UNREAL TOURNAMENT.

But what will people actually pay for? Can we survive as an industry when we have such a broad variety of target platforms? Can you reasonably expect to recoup your development costs when making original titles for low-end devices?

These are certainly increasingly challenging questions. With all of these potential platforms, it's difficult to choose your targets. One possibility for low-end devices is to repurpose older titles, as Nintendo has done with MARIO KART for Game Boy Advance, and Maxis has done with SIM CITY for Palm. On cell phones we see the perennial favorites, SNAKE and TETRIS.

For high-end games, you could team up with a prominent license in order to have a better chance of scoring a hit. Major publishers these days invest two to three years and multiple millions of dollars to create an original title. It's a huge risk. If a game flops, it could bring the whole company down.

So how do you ensure that you're going to make money on your game?

One possibility that publishers are beginning to turn to is incorporating advertising. If you get advertising dollars from Company X to include its product somewhere in your title, you have at least some guaranteed source of revenue. Or you could make games that are purely advertising vehicles, like Shockwave or WildTangent titles which go up on a movie web site as part of the pre-launch hype.

But how much advertising is too much?

This magazine is largely paid for through advertising. That's true for most media. You can receive your television signal for free via your antenna, because the broadcaster sells advertising based on who they expect will be watching. There is *big* money involved there.

How much are you willing to sacrifice control over your creative vision in order to guarantee some revenue? Will you let your game be sponsored by Coke? Will you put Absolut vodka on the shelf in the bar that your character walks past? Will your non-player character prefer Mountain Dew?

The use of advertising in games is a complicated issue. Too much advertising in anything is a real turn-off. In a world increasingly dominated by advertising, your game will soon stand out more if it *doesn't* include ads. But will you be willing to take that risk?

Andy Warhol painted Campbell's soup cans. Will you soon have your artist modeling them?

Task Switching

I joined the *Game Developer* magazine team just over a year ago, excited to have the opportunity to contribute to our community. It's been a very interesting year, and I'm happy to have had the pleasure of introducing you to exciting new technology and innovative developers in our industry. But ultimately, I'm a game developer at heart. So I'm going back out into game development. This certainly won't be the last you hear from me, though.

Thanks for reading my words over this past year, and thanks for continuing to read *Game Developer*. We hope to provide you with information on the cutting edge that helps you in your day-to-day development needs. It's been a pleasure for me to be at the helm of this vessel.

For me, it's not Game Over. Just a context switch.

GameDeveloper
600 Harrison Street, San Francisco, CA 94107
t: 415.947.6000 f: 415.947.6090 w: www.gdmag.com

Publisher
Jennifer Pahlka *jpahlka@cmp.com*

EDITORIAL

Editor-In-Chief
Mark DeLoura *mdeloura@cmp.com*
Senior Editor
Jennifer Olsen *jolsen@cmp.com*
Managing Editor
Laura Huber *lhuber@cmp.com*
Production Editor
Olga Zundel *ozundel@cmp.com*
Editor-At-Large
Chris Hecker *checker@d6.com*
Contributing Editors
Daniel Huebner *dan@gamasutra.com*
Jeff Lander *jeffl@darwin3d.com*
Tito Pagán *tpagan@w-link.net*
Advisory Board
Hal Barwood LucasArts
Ellen Guon Beeman Beemania
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 & Marketing
Greg Kerwin e: *gkerwin@cmp.com* t: 415.947.6218
National Sales Manager
Jennifer Orvik e: *jorvik@cmp.com* t: 415.947.6217
Senior Account Manager, Eastern Region & Europe
Afton Thatcher e: *athatcher@cmp.com* t: 415.947.6224
Account Manager, Recruitment
Morgan Browning e: *mbrowning@cmp.com* t: 415.947.6225
Account Manager, Northern California
Susan Kirby e: *skirby@cmp.com* t: 415.947.6226
Account Manager, Western Region, Silicon Valley & Asia
Craig Perreault e: *cperreault@cmp.com* t: 415.947.6223
Sales Associate
Aaron Murawski e: *amurawski@cmp.com* t: 415.947.6227

ADVERTISING PRODUCTION

Vice President, Manufacturing Bill Amstutz
Advertising Production Coordinator Kevin Chanel
Reprints Stella Valdez t: 916.983.6971

GAMA NETWORK MARKETING

Senior MarCom Manager Jennifer McLean
Marketing Coordinator Scott Lyon
Audience Development Coordinator Jessica Shultz

CIRCULATION

BPA INTERNATIONAL
Game Developer is BPA approved

Group Circulation Director Catherine Flynn
Director of Audience Development Henry Fung
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
President, Business Technology Group Adam K. Marder
President, Specialized Technologies Group Regina Starr Ridley
President, Technology Solutions Group Robert Faletra
President, Electronics Group Steve Weitzner
Senior Vice President, Human Resources & Communications Leah Landro
Senior Vice President, Global Sales & Marketing Bill Howard
Senior Vice President, Business Development Vittoria Borazio
Vice President & General Counsel Sandra Grayson
Vice President, Creative Technologies Philip Chapnick

GamaNetwork
A DIVISION OF CMP MEDIA LLC

WWW.GAMANETWORK.COM

INDUSTRY WATCH

daniel huebner | THE BUZZ ABOUT THE GAME BIZ

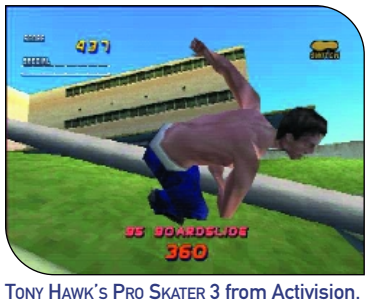
Improving financials. Though fiscal first-quarter losses increased from last year, Electronics Arts still managed to beat analyst predictions. The company reported a first-quarter net loss of $45.3 million, up from a net loss of $42.3 million in the same quarter last year. That loss is equal to 34 cents per share, slightly lower than the 37 cent per share loss that had been expected. EA was also able to point to increased revenues, up 18 percent to $182 million from $154.8 million in the same period last year, as another bright spot.

Activision also managed to beat analysts' expectations by posting a modest profit instead of an anticipated loss. Activision reported earnings of $29,000, after taxes and before charges. The company posted a loss of $5.2 million in the same period last year. Revenues for the quarter jumped 31 percent to $110.6 million from $84.6 million in the same period last year. Much of the improved performance was attributed to strong sales of Game Boy Advance titles. Based on strong sales so far this year, Activision has raised its projected full-year revenue from $605 million to $620 million.

TANG TANG, one of Take-Two Interactive's latest releases for Game Boy Advance.

Acclaim reported a profit of $0.2 million on net revenues of $38.6 million in its fiscal third quarter, marking the company's third straight profitable quarter in fiscal 2001. Net revenues of $38.6 million for the quarter were up from revenues of $33.8 million for the same period last year.

Adding capital. A number of game publishers and developers are using the momentum gained from better-than-expected quarterly results to fatten their coffers in advance of an expected surge in the coming year. Activision has filed with the Securities and Exchange Commission to issue five million new shares of common stock. Based on Activision's share price at the time of the filing, the sale could raise as much as $177 million, funds Activision hopes to use for product development, capital expenditures, joint ventures, and strategic acquisitions.



TONY HAWK'S PRO SKATER 3 from Activision.

Acclaim has already completed its stock sale, a private placement of 9.4 million shares of common stock to institutional investors that earned Acclaim $33.6 million in new funds. The company plans to use the money to pay down debt, finance product development and marketing, and make acquisitions.

Take-Two Interactive has also completed its fund-raising efforts, generating gross proceeds of approximately $22.2 million through a private placement of 1.3 million shares of the company's common stock. Much of this money, however, is earmarked for reducing the company's outstanding debt.

Finally, Crave Entertainment has lined up $35 million in loans as the privately held company prepares to seek its first round of equity financing. Crave hopes to raise between $20 million and $30 million this time around, and is considering an eventual public offering sometime in the future.

Console updates. Japan's Fair Trade Commission announced that Sony violated fair trade rules in hardware and software sales. The commission contends that Sony Corporation and Sony Computer Entertainment violated trade rules by pressuring retailers to sell Playstation 2 games at set prices and through predetermined channels, and hampered free distribution by directing wholesalers to sell only to retail outlets. Sony was ordered to halt the practice immediately.

Microsoft is hoping the Xbox will make some waves in Japan, but it might not arrive before the end of the year. The company had earlier set a date of November 8 for the North American launch, but didn't set dates for other markets. As Microsoft prepared to announce a release date and pricing information for the launch of Xbox in Japan, it wasn't promising that the console would arrive in time for the Japanese holiday season. Yoshio Hongo, senior manager of public relations for Microsoft in Japan, characterized the eventual launch date as, "around the same time of, or not too far behind, the U.S. launch," but wouldn't say whether the launch would come before the end of the year.

Amn, the main city in Interplay Entertainment's BALDUR'S GATE II.

Konami buys into Hudson Soft. Konami is spending $40 million to acquire a 38 percent stake in Hudson Soft. The transaction, which will make Konami Hudson Soft's single largest shareholder, is designed to help offset increasing development costs. Konami also hopes to use its stake in Hudson to diversify its product offerings and increase its market share. Konami will increase its holdings in Hudson to 45 percent in December.

Interplay cuts jobs. Interplay Entertainment saw its stock recover slightly on reports that the company has cut some of its staff. Interplay, still looking for a rebound after putting negotiations with a potential buyer on hold earlier in July, has confirmed that it laid off between 55 and 65 employees. The job cuts were described as a cost-cutting move, possibly signaling Interplay's intentions to improve its financial situation without the help of a strategic partner.

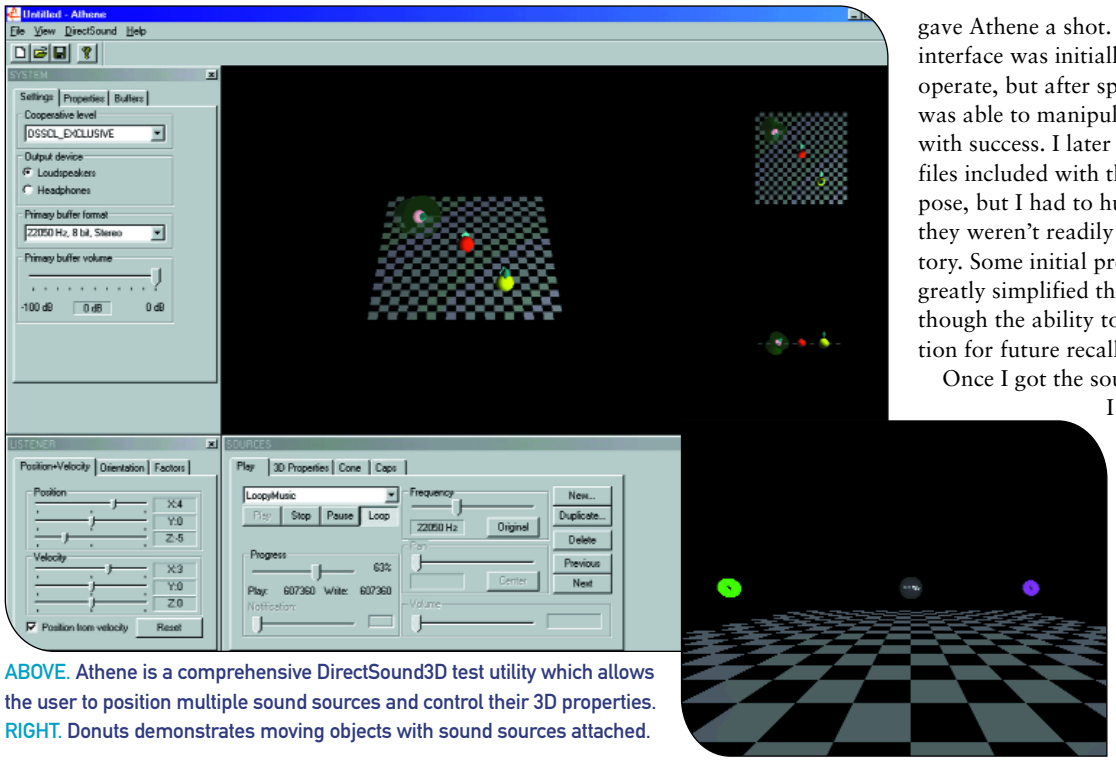
UPCOMING EVENTS CALENDAR

COMDEX FALL

LAS VEGAS CONVENTION CENTER
MGM GRAND CONFERENCE CENTER
LAS VEGAS HILTON
SANDS EXPO AND CONVENTION CENTER
Las Vegas, Nev.
November 12–16, 2001
Cost: variable
www.key3media.com/comdex/fall2001

Sensaura SDK

by aaron marks



ABOVE. Athene is a comprehensive DirectSound3D test utility which allows the user to position multiple sound sources and control their 3D properties. RIGHT. Donuts demonstrates moving objects with sound sources attached.

Sensaura is a high-quality three-dimensional sound recording technology that emphasizes spatial accuracy by employing natural hearing cues that the brain uses to determine the direction of sound. This 3D positional audio technology, in the form of patented mathematical algorithms, is licensed to audio chip vendors that supply many of the leading sound card, motherboard, and desktop and notebook PC manufacturers. The Sensaura SDK provides software developers with the information they need to use this technology in order to create more believable soundscapes.

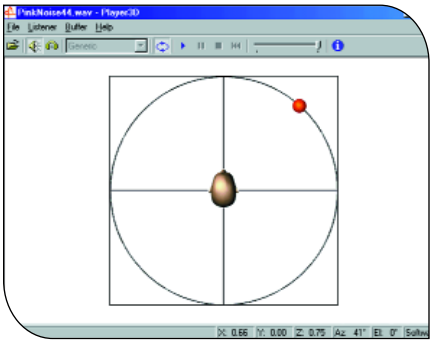
Previously, Sensaura's open APIs had been incorporated in the DirectX and Creative Labs SDKs. After the introduction of its ZoomFX API, the company felt it necessary to produce its own SDK in order to showcase its capabilities fully. While the interface of this SDK release is rudimentary and graphically lacking, the audio demonstrations and information included are highly enlightening. Its primary contents include many utilities and built-in sample versions, full documentation and relevant technical papers on 3D audio, useful libraries, source code examples, and information on other property sets supported by Sensaura drivers. Most questions can be answered easily by delving into the package.

The first of the 3D audio utilities and test programs included in the SDK is called Athene. Sensaura touts this particular program as a comprehensive DirectSound3D test utility that gives users the ability to position multiple sound sources and control their 3D properties, including full support for sound cones. I gave Athene a shot. The demonstration interface was initially cumbersome to operate, but after spending a few minutes I was able to manipulate my own sound file with success. I later discovered short .WAV files included with the SDK for this purpose, but I had to hunt for them, since they weren't readily apparent in the directory. Some initial presets would have greatly simplified this testing process, though the ability to save the demonstration for future recall did help.

Once I got the sounds started, however, I was in for another puzzler. It wasn't clear which object on the interface was the sound source and which was the listener position. I moved an object to the left, and the sound moved to the right speaker. For a minute, I thought I had my speaker wires crossed. I soon managed to regain composure and make some sense of the interface. Once I got going, I began to appreciate the complexity of manipulating multiple sounds around the listening position.

The listener can be fixed or continuously moving along the X, Y, or Z axis; sound sources can also remain fixed, pointed in a specific direction, or be moving. Further parameters can be adjusted such as relative distance, amount of Doppler effect, roll-off factor, and the orientation of the sound (pointing toward or away from the listener and focused in either a small or wide area via a parabolic arc adjustment, or sound cone). I set up a complex demonstration using a wide variety of settings and could easily hear the results, despite the graphical depictions moving in every direction.

The next demonstration in the SDK is called Donuts. The included documenta-



Player3D demonstrates use of a single sound source in the horizontal and vertical planes.

tion describes the Donuts demo as "moving objects with sound sources attached," including support for turning MacroFX on or off and applying reverb. The program opens in a "room" with a sphere in the center, aurally depicted as a music loop. Two colored "donuts" are rotating around the sphere, each triggering its own sound — one chirps, the other honks. The user is initially at a distance from the sphere, watching and listening to the orbiting donuts. Only accidentally did I find that using the arrow keys changed the listening position in the horizontal plane around the room, which enabled me to experience the sounds from all angles. Additional manipulation of the Page Up and Page Down keys allowed movement vertically, increasing the capabilities of the demonstration. Properties of the room can also be adjusted using EAX reverb settings to simulate various environments. Overall, this demonstration gives a good indication of 3D audio positioning using both fixed and moving sound sources, much like in first-person shooter applications.

The DirectSound3D demo borrows from both Donuts and the well-known arcade game ASTEROIDS. Its purpose is to show how easily one can add 3D positional audio to a game using Microsoft's DirectSound3D API. The full source code for this application is even included in the SDK, providing a good starting point for programming. Instead of a room, as in the previous demo, the sphere hangs in the center of space with a flying saucer and a rock orbiting it at different speeds. As before, the listener can be positioned anywhere in the XYZ axes using the arrow and Page Up/Down keys; the added bonus to this demonstration is the ability to shoot the objects. This allows the listener to experience both the laser shots and explosions relative to a specific position, even while the reference point is moving.

Another efficient demo, Player3D, is a simple but very effective application that allows users to position a single sound source in 3D space while allowing full horizontal and vertical positioning. This utility also enables reverb and includes the ability to turn MacroFX off and on via the registry and set ZoomFX properties. For software buffers, it is also possible to select different 3D rendering algorithms. The display features a head in the center of the screen; the head represents the listener. Sound travels in either a horizontal or vertical circle around the fixed head. Users can set the sound to fly by from far left to far right, ping-pong around the room, or be manually controlled for exact positioning. Doppler and reverb effects are also available for varied environmental representations.

The first listen I had at these demos — on my laptop — was not very impressive. However, after loading the SDK onto my main audio production rig, the results were spectacular. I found that by using the included white and pink noise files instead of the music loops, the perception of the location of a sound was incredibly accurate, even when using only two speakers. With my eyes closed, I could easily determine whether the sound was above or below me, and I could judge distance very well. I was quite impressed and could immediately see the possibilities of this technology in gaming applications. Users should definitely check this out with decent speakers for the full effect.

The final demo is the second of the source code examples, ZoomFX, which demonstrates the use of the complete property set. This utility program allows creation of multiple 3D buffers and the setting of their properties in addition to the ZoomFX bounding box and orientation. Because this program requires the use of a recent Sensaura driver, I didn't actually get to see any of it work, greeted instead by an error message. Fortunately, the SDK has since been updated with the working ZoomFX drivers, although a working release would have been nice.

Overall, the Sensaura SDK fulfills its mission of demonstrating the ZoomFX API and their 3D positional audio technology. The extras add some serious value as well. After listening to the audio examples, I spent some time with the documentation and was not disappointed with the depth of the additional 12 technical papers. The included source code and library files should fit most programming needs. While the interface was nothing much to look at, nothing at all like Sensaura's Showcase CD they pass out at trade shows, 3D sound is, after all, the purpose behind this SDK, and they hit the nail right on the head.

Aaron Marks is a game composer, sound designer and owner of On Your Mark Music Productions (www.onyourmarkmusic.com). He is also a frequent contributor to Game Developer *and Gamasutra. Contact Aaron at aaron@onyourmarkmusic.com.*

SENSAURA SDK ★★★★

STATS

SENSAURA LTD
Dawley Road, Hayes
Middlesex, UB3 1HH
United Kingdom
+44 (0)20 8848 9779
www.sensaura.com

PRICE
SDK is free to download after registering with Sensaura's developer program.

SYSTEM REQUIREMENTS
233MHz Pentium or higher, Windows 95/98/NT

PROS

1. Audio demonstrations were excellent aural displays of Sensaura's technology.
2. Included 3D audio technical papers made understanding the 3D audio concept very easy.
3. Source code and libraries included provide a solid foundation for 3D audio programming.

CONS

1. Interface was initially confusing and required concentration to set up properly.
2. Graphics were uninteresting and detracted from the audio demonstration.
3. No updated Sensaura drivers available in SDK.

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

COREL'S BRYCE 5

by mark peasley

Bryce 5 is the first full revision of the classic landscape software to come out of Corel Corporation since they purchased it from Metacreations. For those of us who have followed Bryce for some years, there has been a certain amount of concern as to whether Bryce would improve or degrade under the control of Corel.

Overall, some major improvements have been made under the hood of this latest release. There are some solid enhancements to the package, including network rendering and a couple of new interfaces such as Tree Lab and Light Lab. In addition, there is now a metaball object as well as improvements in the Terrain Editor and more in-depth rendering options.

Out of the box, you will find the Macintosh/Windows application disc and a support disc containing additional presets, tutorials, and some eye candy to help inspire the creative juices. One interesting side note is that all of the scene files are still in Bryce 4 format, which loads into Bryce 5 just fine, but implies that not as much time was devoted to this portion of the upgrade. The documentation has been downgraded to a single color manual and no quick reference card. I had an invalid serial number supplied with my product, which required a 30-minute call to technical support to rectify. Twenty minutes of the call was wading through the phone system at Corel Corporation and being on hold until I talked to a real human. Fortunately, once I was connected, I was supplied with a new, valid number very quickly.

Once installed, the product appears relatively unchanged from version 4, with the notable exception of a questionable upgrade to the overall user interface. A more monochromatic and less polished revision replaces the older one. With the addition of some new menus, which follow their own color scheme, the entire UI feels a bit haphazard and much less cohesive in its approach. The Terrain Editor has been upgraded to floating windows, which allows for user-customizable layouts, but oddly enough none of the other editors was similarly upgraded. On the plus side, some of the interfaces have been redesigned with less clutter. As an example, the Create menu is simplified and BryceTalk is now gone.

One of the new icons on the Create menu is a tree, which generates a fully modeled 3D tree. Editing the new object will take you to the Tree Lab, which is a totally new interface in Bryce 5. It allows you to create trees from scratch or select from 60 preset types of tree trunks and corresponding leaf shapes. As with most interfaces within Bryce, it takes some getting used to. A new Minimum/Maximum slider is available for many of the selections. Unfortunately, the preset function isn't implemented fully, so even if a specific tree trunk and foliage type are chosen, it's still up to the user to determine many of the parameters. Several of the selections are not intuitive enough to make this an easy process. However, even experimentation in the preview mode brought my system (a 433MHz Pentium III with 128MB of RAM and a 32MB Oxygen VX1 video card) down to a crawl. Be aware that it is very easy to create some graphically intense, high-polygon-count trees in no time at all.

The new Light Lab expands users' control over the lights. Sliders are provided to control intensity, edge softness, cast shadows, shadow ambience, soft shadows, and falloff range. A timeline slider and keying capabilities are also available for animations that involve lights specifically. Still missing in Bryce 5 is true radiosity, but with the new controls it will be much easier to fake.

The new Tree Lab allows the user control over branching, trunk, and foliage characteristics.

In the Terrain Editor, the maximum grid for a terrain has been bumped up by several orders. You can now select a "gigantic" (2048×2048) or "planetary" (4096×4096) resolution terrain. As the planetary resolution mesh weighs in at 33.5 million polygons, a bit of restraint might be in order to keep renderings within reason.

Overall, Bryce 5 is a solid upgrade, but it is a bit rough around the edges. For the professional-level modeler and animator, the interface and unique way Bryce approaches 3D editing and rendering may seem a bit foreign. Bryce still lags behind the other 3D applications in the way it handles materials.

Some things I'd love to see in future versions include editable hotkeys for all commands and functions, camera and lighting controls that are more in line with other major 3D applications, and a more functional motion graph that allows easier editing of animations and camera motions.

Bryce 5 supports MacOS 8.6 and up as well as OS X, and Windows 98/2000/ME/NT 4 (SP 6). It retails for $299, but an upgrade from any previous version only costs $149.

★★★½ | Bryce 5 | Corel | www.corel.com

Mark Peasley is currently working on Xbox titles at Microsoft. Contact Mark at mp@pixelman.com.

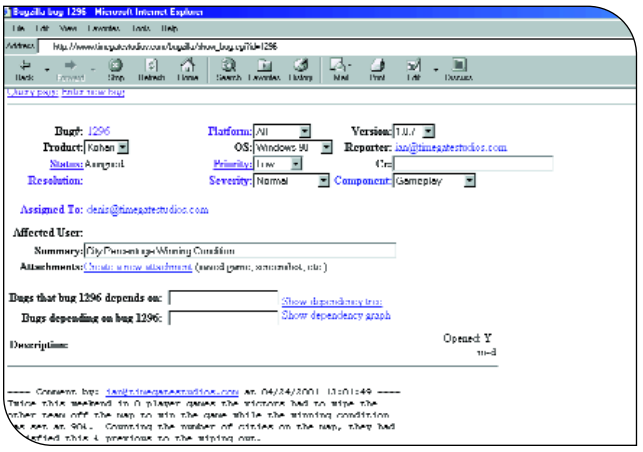
MOZILLA'S BUGZILLA

by denis papp

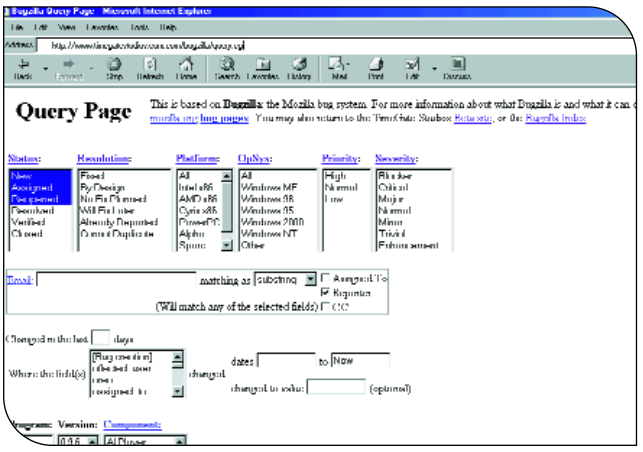
Bugzilla is a defect-tracking system created by Netscape Communications (listed as the initial developer in all the code), to track bugs in their projects. It is now open source and maintained by a group of developers via the Mozilla Organization. You can freely download, use, and modify the source under the MPL (Mozilla Public License). The fact that it is open source and free makes it a tempting solution for your defect-tracking needs. But is it right for you?

The disclaimer in the Readme for Bugzilla states, "This is not very well-packaged code. It's not packaged at all. Don't come here expecting something you plop in a directory, twiddle a few things, and you're off and using it." This statement is not an exaggeration. This package is really just a suite of Perl scripts that use HTML forms to interface with a MySQL database. Installation

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



Bugzilla's display/editing form.



The powerful Bugzilla query form.

requires that you first get MySQL and Perl running and have a basic understanding of both. Administration, and taking advantage of its flexibility, requires a much higher level of comfort with Perl. That said, the experience for the user is not so bad. The UI is entirely web-based, using standard HTML forms with cookies added for convenience. The default forms are a little cryptic. But the biggest benefit of this software is the flexibility you have in customizing it for your needs, since the code is moderately clean.

We started using Bugzilla at TimeGate Studios when our KOHAN project was near the start of the beta phase (the most recent version of Bugzilla is 2.14). We used a modified version called Fenris by Loki Software (see www.lokigames.com/development/fenris.php3). Fenris, based on an earlier version of Bugzilla, adds several convenient features. Since then, most of the major differences between Fenris and Bugzilla have been integrated into the main source tree.

Installation requires a moderate degree of technical competency. According to the Readme, installation is supposed to be "pretty straightforward" once you have MySQL and Perl running. However, Bugzilla was originally written for Unix, and our beta server was running on Windows 2000. I decided to be adventurous and try to get it running (any excuse to play with Perl). I found one guide describing some of the problems involved in the conversion, but it still took over a day of fixing code, debugging, and discovering quirks with the IIS web server (I eventually installed Apache for NT). Today, there are several guides detailing the steps required to install Bugzilla on Windows 2000. Matthew Barnson maintains "The Bugzilla Guide" at www.trilobyte.net/barnsons/html, which has a section on Win32 installation and several tips from other people.

In terms of features, Bugzilla gives you almost everything you might ask for from defect-tracking software. At TimeGate, we've even modified it for use as a more complex change-tracking/QA process. It features a complex query system, e-mail integration, file attachments, platform independence, and, provided you are comfortable with Perl, a high degree of flexibility and control. You can easily add fields and modify behavior.

On the other hand, there are negatives beyond the nontrivial installation process. The e-mail notification system is too verbose, it's not localized, and queries on bug descriptions are extremely slow (although this has been improved in recent versions). On the programming/customization side, there are also some minor problems. While the code is moderately clean, the UI and logic are not properly abstracted, and the entire system relies on a single database package, MySQL.

The UI can be daunting to the new user and is not well streamlined (it is primarily functional). However, any user comfortable with web forms can quickly figure out the basics. Furthermore, with a small amount of programming work, you can clean up the forms.

Should you use it? Bugzilla is primarily for small- to moderate-sized projects. The fact that it is web-based also makes it very convenient for dealing with public testers, and the number of users is scalable. The openness and flexibility of the system is a big benefit — you are not forced to a particular methodology. In the game industry, people tend to like to have the power to customize solutions to their needs. However, due to the nature of this flexibility, you must be prepared to put more work into installation and administration. Furthermore, if you are looking for a high-performance system, it could probably be better optimized or support a more powerful database back end. If you are looking for a secure system, it will take a lot of work.

Bugzilla worked well for our RTS game KOHAN, but will it work well for our massively multiplayer RPG? With a lot of work and a filtered interface for a much larger test audience, perhaps. We would then have a program designed exactly to our needs. But we may find a prepackaged plug-and-play program that meets all of our requirements, saving implementation time.

I would have given Bugzilla three stars due to the problems we encountered (and the work involved), but I'm going to add a star in favor of Bugzilla's flexibility and cost. The only thing you can waste with a free product is time.

★★★★ | Bugzilla
The Mozilla Organization | www.mozilla.org/projects/bugzilla

Denis Papp is the lead software engineer at TimeGate Studios. Contact Denis at denis@timegatestudios.com.

Hanging at the Arcade

with Elaine Hodgson

Elaine Hodgson, president and CEO of Incredible Technologies, has been bringing networked gaming into bars and taverns since 1996. We sat down recently and had a chat with her about her company and what opportunities she sees in the future for arcade game development.

Game Developer. Tell me about Incredible Technologies.

Elaine Hodgson. Incredible Technologies has been around for 16 years. We started in the game business doing stuff on the 8086 PC, the Apple, and the Commodore 64. My partner Richard Ditton did the programming for the original Commodore 64 WINTER GAMES, which at the time was done under contract to Action Graphics, for Epyx. Before we worked for Action Graphics, Richard and I worked for Martin Glass & Associates, which contracted with Bally/Midway to do coin-operated videogames. We did some games for them, which included games such as DOMINO MAN, TAPPER, and JOURNEY.

One of the first coin-operated games we put out as Incredible Technologies was GOLDEN TEE GOLF, which sold pretty well. After a few years, and a few other arcade titles, we decided to develop a new GOLDEN TEE GOLF and concentration on the bar market. We've had a lot of success there. The arcades have diminished quite a bit, since kids can now play comparable games in their homes. But the adult market still walks over to their local bar, and there's a certain age group that likes to go out to bars and socialize there.

GD. How have you been able to keep GOLDEN TEE GOLF (GTG) fresh for so long?

EH. We're currently on the third hardware revision of GTG. The earlier versions were pseudo-3D, but this one uses the 3dfx Voodoo 3 chip for very real 3D courses and fly-bys. The player drives and putts using a trackball, which is a very intuitive input device. It makes the player feel like they can golf well, even if they don't really know how to play.

We do periodic software upgrades via CD to install new courses, and we can also install patches and upgrades remotely. For example, we just added a patch to allow players to play either the back nine holes of a course or the front nine. We can do this because the majority of GTG machines phone home every night.

GD. GTG calls home?

EH. Every night the GTG machines call back to the server and download national statistics, compare scores for tournaments, and so on. We run national-level tournaments with multiple divisions to keep people at different levels challenged. Players can go to a web site and look at their statistics.

The network also allows us to support advertisers. We have two at this point, Michelob and Rumple Minz. Michelob, for example, sponsors what they call Michelob Thursdays. The GTG machines know when it is Thursday and add Michelob advertisements which are integrated into the game, and more little prizes. We didn't want blatant print ads popping up on the screen, so we integrated the ads into the game on things like billboards and blimps. Michelob pays for that, and supports it with people on the street who give away tchotchkes for playing the game.

ABOVE. Incredible Technologies' Elaine Hodgson.

GD. It seems like ads are finding their way into more and more games. How do you feel about that?

EH. Everybody hopes for the great white hope of advertising to help them fund their game projects. We've found it to be useful, but the actual money that advertisers are willing to spend is not particularly overwhelming. It's still considered a new media, and they're really trying to figure out how it is valuable.

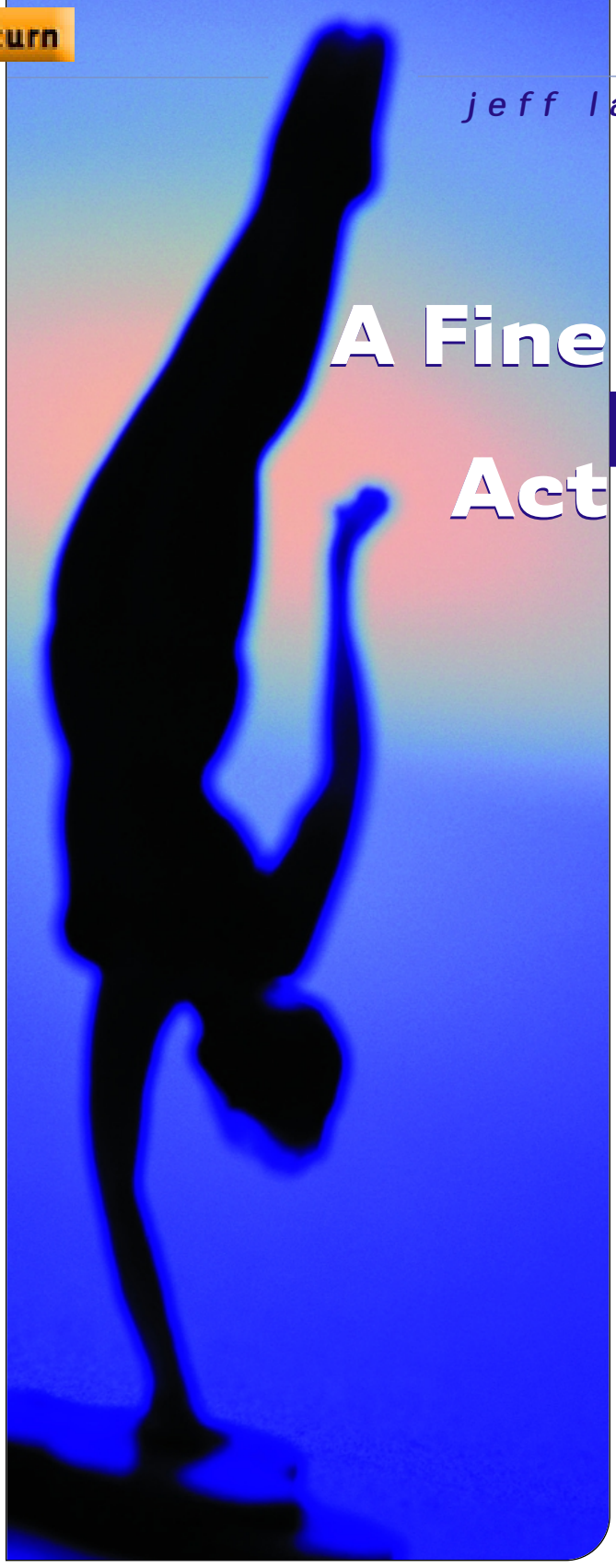
We're sensitive to the amount of advertising. We don't want to prostitute our game too much, because we want people to play the game, and we worry that too many ads will detract from the game. That's why we actually incorporate the ads into the game itself, like you would see at a real golf course.

GD. Aren't arcades dying off? How are you able to continue being successful?

EH. The death of coin-op is greatly exaggerated, as Mark Twain would say. It's going to exist because people want to get out of the house. We've been successful in the bar market, because that's where people already are, and we're there, ready for them to use in a social situation. We're looking for more games that will work there. But we're also hunting for that new, compelling thing that will bring people out of their homes to seek it out — some new gee-whiz wow technology. We're always on the lookout for that. People are pretty used to the great-looking graphics now, so it has to be more than just that.

GD. Why do you like working on arcade games?

EH. Designing arcade games is different than designing home games. In a PC or console title, the player buys it and the game needs to play for 40 hours or whatever for their money. In the coin-op environment, you have to catch them quickly and entice them into continuing. It's like, O.K., put in a dollar and see if you like it. O.K., you like it, now put in another dollar and play some more. You have to focus on getting that 500th credit just as much as you do the first one. And that's the really fun, challenging part of designing arcade games.



A Fine Balancing Act

I went for a ride in a sailplane the other day. I had always wanted to try it, but just never gotten around to doing it. For some family birthdays this year, we decided to check it out. I figured all those years of playing flight simulators would really pay off. Surely after the number of MIG kills on my record and the variety of jets I have been "checked out on," I was ready to tackle my SZD 50 Puchacz (nicknamed "The Pooch"). There was nothing to it. No afterburn, no flares or chaff to worry about. I knew how to execute an Immelmann to shake a bogey. I was ready.

PC gaming peripherals have become very sophisticated pieces of hardware. My PC joystick has integrated force feedback, 12 buttons on the stick, and foot pedals that attach for the rudder. This setup is much more sophisticated than the rugged simplicity of the controls in my Polish-made glider. Of course in the case of actually taking to the air, reality beats technology hands down. The real feeling of flying through the air, subjected to the most subtle changes of atmospheric pressure and the aggressiveness with which the plane responds to your most subtle gestures, is truly an inspiring level of immersion.

Once I got over the initial vertigo and got the hang of the controls ("You're going a bit too fast…" "O.K., now you are stalling…"), I sat back for a minute and really soaked it in. I was soaring with the eagles as Da Vinci had vividly imagined in his drawings, feeling fortunate to live in an age where such dreams are reality.

The gamer in me was struck by the view. The ground below me was quite surreal. The desert landscape became a patchwork of textures (thankfully quite tileable). I realized that we simulate landscapes from a bird's-eye view pretty well these days. The view below the Pooch could very well have been generated from a desert tile data set. The haze of the afternoon inversion layer allowed for a reasonable depth of view before fading to the far clipping plane.

However, the pit of my stomach knew this was no game. When I asked the instructor what it felt like to stall one of these gliders, he was only too happy to show me. We pitched the plane up and it began to slow. As we reached the stall speed, the air flow over the wings became more turbulent and the whole plane started to shake. We then hit the critical stall point and the plane was no longer moving forward, but falling. The plane then pitched forward and we began to speed up, gaining enough speed to once again level the plane. I have stalled planes in flight simulators many times but it doesn't come close to the real thing. The real forces of physics that we try so hard to simulate in a plausible manner were felt in every cell in my body.

JEFF LANDER | *Jeff has now opened a new division of interactive Feng Shui at Darwin 3D. For a small fee, he will move boxes and add an ammo clip or two around your 3D game to ensure the levels are in balance and harmony. For more information, contact him at jeffl@darwin3d.com.*

Tipping Point

The way the plane reacted after the stall was not simply luck and good flying by the instructor. The plane was designed to behave that way. The natural force of gravity enabled the plane to recover from the stall. Just like any other physical body, a plane has a center of mass where the force of gravity is applied. This is also the point about which an object will rotate when tumbling through space.

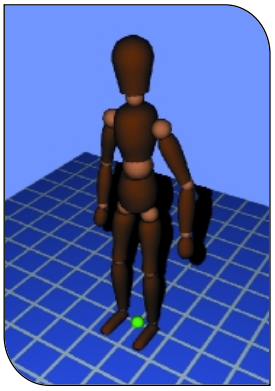


FIGURE 1. A simple standing mannequin.

In order for a plane to fly in a straight line, the forces being applied to the plane must cancel each other out or be in balance. The force of gravity, which is pressing down on the plane at the center of mass, must be cancelled out by some other force. In the case of the sailplane, the lift force generated by the wings counteracts the gravitational force caused by the combined mass of the plane, the instructor, and my own massive self.

When the plane stalls, the front wing is no longer generating as much lift, so gravity takes over and we start to fall. Thankfully, the designers of the plane were thoughtful enough to anticipate this situation and provide an easy solution. While the front wing is no longer providing me with lift, the rear wing has not stalled. It continues to generate lift, pushing up on the tail of the plane. However, the forces acting on the plane are no longer in balance. Gravity is pushing down on the center of mass, which is near the front wings of the plane, and lift force is being generated by the tail. This tail lift causes a torque about the center of mass, pitching the nose of the plane forward. As the plane pitches forward, it gains speed until the front wings once again begin to generate lift and the forces come back to equilibrium. The careful balance of forces acting on or about the center of mass of the plane is the power that enables us to take to the sky in flight.

Get the Balance Right

The center of mass is an important concept for everyone to understand. Balancing a spinning basketball on the end of a finger or a stack of books on your head is an exercise in control over the center of mass. Over the course of people's lives, they learn to use this fact almost implicitly. Pick up a book and try to balance it on your index finger. In order to accomplish this task, you need to find a point close to the center of the book and hold it up at that position.

You may not be aware of where your exact center of mass is located. However, in order to simply stand up straight, muscles all over your body are constantly making minor adjustments to ensure that your center of mass is supported, allowing you to remain standing balanced. If you have watched children learning to stand and walk, you realize this is a learned behavior. Most of you have probably also realized that this ability can be unlearned fairly easily with a couple of well-placed cocktails.

As grown humans, we are very sensitive to issues of balance. We can sense immediately when people look like they are about to fall. When something is not in balance, it looks wrong. Magicians use this fact to perform feats that look impossible. Mimes pretend to be leaning on walls that aren't there by subtly shifting their center of mass. Home decorators charge large fees to ensure that your home is in proper balance. Though, how they calculate and adjust the center of mass for a building by adding a plant or two is beyond me.

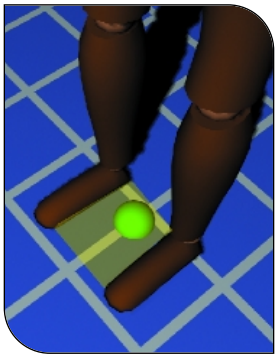


FIGURE 2. The support point and support polygon.

In computer games, however, the rules of balance are regularly broken. Dead aliens lie extending straight over stairways, defying gravity. Giant battle mechs never seem to fall over no matter how top-heavy they are. Human characters walking along look like they should fall flat on their face.

This is not always the fault of the computer artists, though it sometimes can be. Characters are very hard to animate. Making them move with plausible weight and balance is something that even the most budget-blasting visual effects studios find challenging. Even motion capture data will look wrong when you change the physical build of the characters or start blending motions together.

The problem is that the 3D tools are just not smart enough to help the artists out. However, if the tools were aware of the concepts of gravity and balance, it would be easier to create more plausible motions.

Where Is My Center of Mass?

The first step toward creating more balanced characters is finding their center of mass. It is pretty clear just looking at a human character that the center of mass is going to be somewhere around the hips. As a person moves, this center of mass moves around as well. In order to determine the actual center of mass at any given time, I need to make a few calculations.

When we animate characters, we tend to build them from a hierarchical skeleton. The form of the character can be approximated by attaching basic sphere and cylindrical shapes to the bones in this skeleton. You can see an example of this in the mannequin in Figure 1.

With any sphere, such as a basketball, the center of mass is in the center of the sphere. With a cylinder, the center of mass is on the center radius line, halfway up the cylinder. I can use this information to approximate the center of mass for each part of the mannequin's body.

Unfortunately, the average of the centers of mass of all of the body parts will usually not give me the center of mass of the object. That is because some of the body parts are larger (and therefore probably more massive) than other parts. For example, the torso is clearly larger than the hand. Also, the density of each body part could be different. A character may have an arm made of steel and legs of composite carbon fiber. This would mean size alone wouldn't be enough to determine the mass of the object.

Fortunately, the center of mass, *CM*, of a composite object like the mannequin can be calculated very easily using this formula:

$$CM_{Total} = \frac{\sum_i CM_i m_i}{\sum_i m_i}$$

The center of mass of the total character is equal to the sum of the center of each part times its mass divided by the sum of the masses.

I can make my life much easier for automatic calculations if I just assume that the density of the object is constant and the mass of each part is relative to the physical size of that part. The artist could always override the default mass to allow for lighter or heavier body parts. In practice, even rough calculations of the center and mass of each part is generally sufficient.

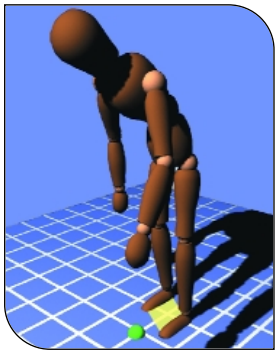


FIGURE 3. Leaning over.

I'm Still Standing

Once I know the center of mass of the character, I need to figure out if that character is currently balanced. This is done by looking at the support point and support polygon of the character. The support point is easy to calculate. It is just the center of mass of the character projected onto the ground. This is just as if you were to take the center of mass and drop it, letting gravity take effect. In the case of my standing mannequin, the center of mass is indeed inside the area of the hips, and the support point is between the feet, marked with a green hemisphere.

The support polygon is a little tougher to calculate. Each part of the character that is in contact with the ground forms a part of the support polygon. In this first case, both feet are touching the ground, so the outline of the feet makes up the points that are on the support polygon. The convex polygon which encloses these points is considered the support polygon.

There are two kinds of balancing, dynamic and static. Dynamic balancing occurs when an object is moving and allows for periods of unbalanced motion while support points are changing. For static balance, when the object is not moving, the support point must fall within the support polygon at all times. You can see this visualized in Figure 2.

Now that you see how this works you should understand a bit more about balance. It is tough to balance on one foot, because the support point must be contained within the outline of one foot. That means that the center of mass must stay directly within that outline as well. High heels just make the support polygon even smaller. This is also why creatures with many legs balance so easily. The support polygon in a creature with a lot of support points is much larger.

The problem with a character in motion is that the center of mass moves. If the feet don't move to compensate, the character should fall over. For example, look at the mannequin in Figure 3.

In this example, the mannequin is leaning over as if to pick something up. Notice that as the waist bends, the center of mass as well as the support point is moved forward. In this pose, the support point is outside of the support polygon and the mannequin is no longer in balance. The mannequin should fall forward. If it does not fall over or compensate for this imbalance in some way, it will not look realistic.

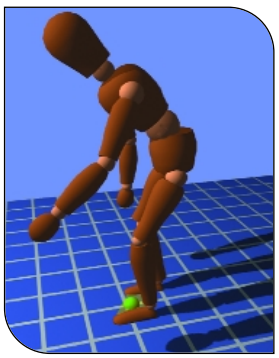


FIGURE 4. Leaning over while balanced.

When people really bend over to pick something up in this manner, what they do is move their body so that the center of mass stays within the support polygon. This can be done by taking a step forward or by pushing their rear end backward to compensate for the forward lean. You can see this adjustment in Figure 4.

Now most talented computer animators are well aware of this phenomenon, either intuitively or by studying physical motion. When they move a character, they will adjust the position of the body so that the center of mass is moved correctly.

As a programmer, I am often accused of messing up a perfectly good animation with technology. By making systems that are more flexible, I can sometimes screw things up. For example, I can take two perfectly weighted and balanced animations, like a bend and a reach, and blend them together to make something totally unique. However, the character may end up completely unbalanced and look very unrealistic.

Further complicating things is the fact that I may throw some algorithmic animation into the mix. I could use a real-time inverse kinematics algorithm to generate a pose that I want to blend into the mix. For example, I may want to enhance a generic grab by solving for the exact location for the object to be grabbed and mixing that in for the final solution. You can see how that could easily foul things up.

So, if the animation system itself is screwing up perfectly balanced animations, how do I fix it? My solution is to apply more technology.

Balancing Act

From the preceding information, it is possible and quite easy to calculate the center of mass, and therefore the support point, for any character. For each given pose, I can also determine the support polygon for the character.

Keeping the character in balance is just a matter of making sure that the support point stays within the support polygon. I can do this in a couple of ways. The decision of which method to use is really determined by what I want to happen with the feet.

I could detect where the support point is relative to the position of the character. This could trigger an animation change where the character would take a step in that direction. Since I also have an inverse kinematics solver running on the character, I could just determine where the feet need to be in order to balance the character. Then, using the IK solver, I could attempt to move the feet to that position.

However, it is possible that I do not want the feet to move. I simply want the

system to adjust the body posture such that the center of mass is supported if possible. This can be done with the IK solver as well. I can add a secondary optimization task for the IK solver that attempts to minimize the distance the character is out of balance. This secondary task is run on the torso and arms of the character. Much like people use their arms to steady themselves, this causes the character to move slightly to achieve balance.

One problem with this technique is the skeletal hierarchy. I usually design the skeletal systems for my characters with the base of the character at the hip. When dynamic balancing is enabled, this causes a bit of a problem. If the hip is the root and it needs to be adjusted in order to achieve balance, it can throw the entire animation off, since it will change every bone in the hierarchy. It will particularly mess up the leg positions, and I really don't want to change those if I want the character to stay grounded. I could use IK to get the feet back where they were, but this can be problematic.

When looking for research references on this issue I found a suggestion from a paper by Cary Phillips and Norman Badler (see For More Information). Their solution to this issue was to select a dominant foot to be the root of the skeleton. The hierarchy builds from that point. The second foot is then placed as an IK task off of the hip. In practice this works pretty well on the engine side, though it seems a bit tough to animate with this hierarchy in the 3D animation system. Perhaps I am just not used to it this way. People more accustomed to 3DS Max's Biped footsteps feature will probably be much more comfortable animating a character by moving the feet first.

Problems, Problems

I have just started really scratching the surface with this idea of dynamic rebalancing. However, it seems clear to me that this could be a plug-in feature of most animation packages. It is definitely helpful to have the system calculate the center of mass and show the balance issues at a keypress.

This simple system does not solve many problems still found in interactive situations. Sometimes given the needed motion, it is not possible to find a balanced pose automatically, and the character should still fall over. If I am willing to let the character fall, I can use the passive dynamic falling system I discussed last month ("The Life of a Silicon Stuntman," September 2001).

Until we are able to solve the problems with complete dynamic animation systems, minor improvements like this kinematic balancing system will have to do.

FOR MORE INFORMATION

Sailplane Rides in the Southern California Desert
www.greatwesternsoaring.com

Phillips, Cary, and Norman Badler. "Interactive Behaviors for Bipedal Articulated Figures," *Computer Graphics* (SIGGRAPH '91 Proceedings), pp. 359–362, July 1991.

Subdivision Surfaces

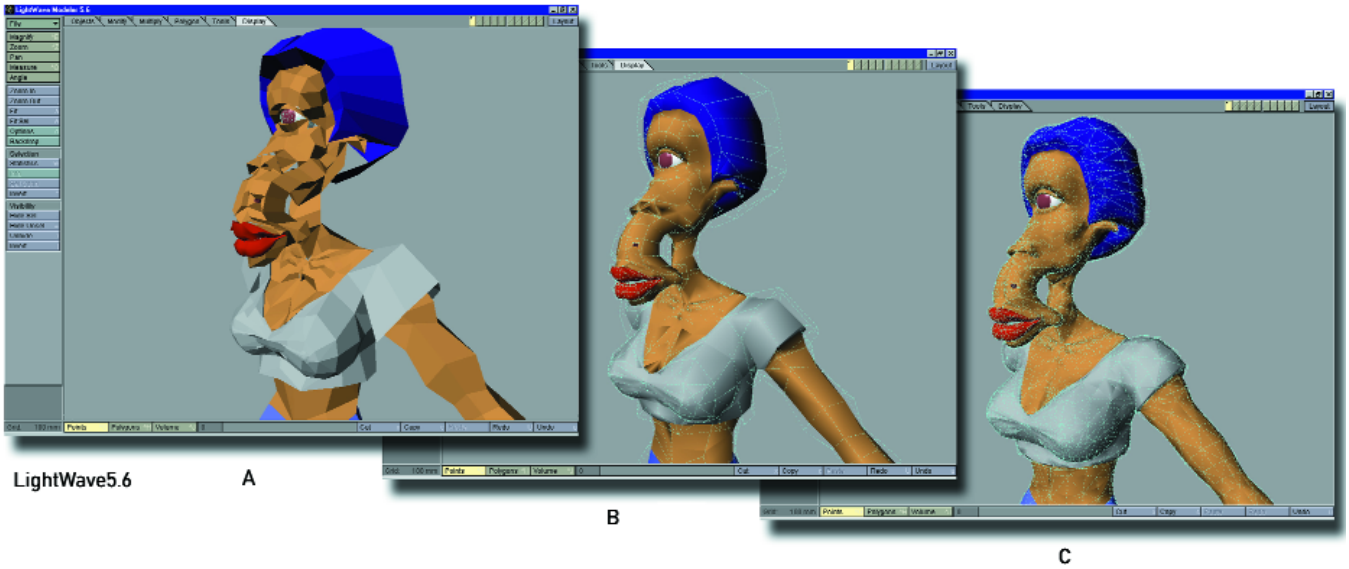
A Practical Alternative for Character Modeling

FIGURE 1. "Camelia" is the main character in an animation short. She was modeled in Lightwave using subdivision surfaces.

Recently a 3D artist friend of mine asked me about using NURBS curves and surfaces for a smooth-surface organic character he wanted to model in Maya. As I began to explain a few of the inherent problems of surface trimming and crack filling with this method, I realized that the challenges he would be facing with this approach reminded me too much of the struggles that my computer animation students at DigiPen had faced only months earlier. I was introducing the students to NURBS modeling in Maya 3. Prior to this, they had used 3DS Max 3 exclusively for over a year in the creation of polygonal models. Modeling complex characters using NURBS for the first time was a seemingly unattainable goal for these students who were facing a completely different modeling paradigm in a completely different 3D package. Similarly, although my very accomplished artist friend was just beginning to ramp up in Maya himself, he was no stranger to modeling low-polygon characters for real-time games using other 3D packages. What they all had in common was a desire for a better method that would have some resemblance to something they were already familiar with. Interestingly, they all had previously developed modeling skills that could transfer well to any other modeling platform.

Organic modeling using subdivision surfaces calls for such skills with polygonal modeling. Subdivision surfaces, simply put, are a way to describe a curved surface using a polygonal model. In fact, working with polygonal modeling tools is essential to becoming proficient in the use of subdivision surfaces. Like the polygonal model, the subdivision surface can be of any shape, size, or resolution. The best part yet is that the ability to subdivide or tessellate this geometry is a common surfacing method found in most 3D modeling programs today.

TITO PAGÁN | *Tito is a seasoned 3D artist/animator working at WildTangent and teaching at DigiPen in Seattle. His e-mail address is tpagan@w-link.net, and his web site is www.titopagan.com.*



FIGURES 2A–2C. A properly created control cage becomes the foundation for good form in the final model at output.

Change Can Be Good

As a game developer, there are many reasons to make the shift from the more traditional polygonal surface modeling method to another surfacing technique. Having a personal desire to produce higher-resolution organic models easily is always a compelling one. Another, more professional consideration is the fact that advanced 3D game engines support real-time cinematic sequences within the game. The recently released game MAX PAYNE uses such cinematic cutscenes extensively for this story-driven game. This approach naturally encourages game designers or directors to rely more heavily on traditional filmmaking ideas and techniques to create tension and drama that not only engages the player but also holds their attention for some time. A character-centric cinematic may even call for full shots to extreme close-ups of these real-time "actors," as in Figure 1.

As the creator of such character models, imagine seeing its head full-screen during run time. What will the character look like upon close inspection? Will your characters give the sophisticated and highly detailed characters of today's console games a run for their money? Or will they have a strong resemblance to the blockhead predecessors of years back? If so, hopefully it is because the game doesn't support the higher polygon count and not because the modeler lacked the skills to provide the richer surface detail.

Another consideration for finding a better solution to modeling is the huge advancements in hardware technology for processing and rendering real-time 3D content and the expectations they promote among consumers. The latest trends in software and hardware capabilities are sure to be supported and exploited by competing developers. This, along with a more sophisticated audience wanting better graphics and the competition for product attention and shelf space at stores, will demand plenty of richer content with higher detail. The pressure is definitely on for developers — both artist and programmers alike — to find effective ways to increase their ability to create more realistic and engrossing 3D content.

A successful and profitable team will find ways to accomplish all of this while keeping the team as small and manageable as possible with little down time learning new skills. We need to be free to increase the detail and diversity of our 3D characters using familiar production methods, and do it all without missing deadlines. Our art path for modeling must be scalable so that we are creating our unique assets one time only for a broad range of target systems while addressing the varying proximity to the in-game camera. As contributing artists and modelers, how do we possibly keep up with such high demands and continue to support such moving targets? How do we increase our development time for each character so that we can populate our games with even more of them in the same time frame?

For those of you facing a transition in 3D packages, or who simply want to expand your knowledge of organic modeling and surfacing methods, subdivision modeling is an excellent alternative. Unlike patch or NURBS modeling, subdivision surfaces are ideal for anyone who has invested much time in creating 3D models using standard polygon modeling and editing tools. The techniques you are accustomed to that resulted in coarser-looking low-polygon characters (700 to 1,200 polygons) will still serve you in creating higher-resolution and engrossing 3D content with several thousand polygons. These may be prerendered or real-time characters. The difference between the two is diminishing quickly. Just for the record, I haven't been using paint or glue in an unventilated area. These are real expectations of today. I'll discuss some examples of higher-resolution character meshes I have created that are intended for real-time animation in a browser-based 3D game engine. These were done in two different 3D authoring packages while employing their respective versions of subdivision technology.

The Workflow

If you are not already familiar with how subdivision works, here is a basic breakdown. You go about creating your

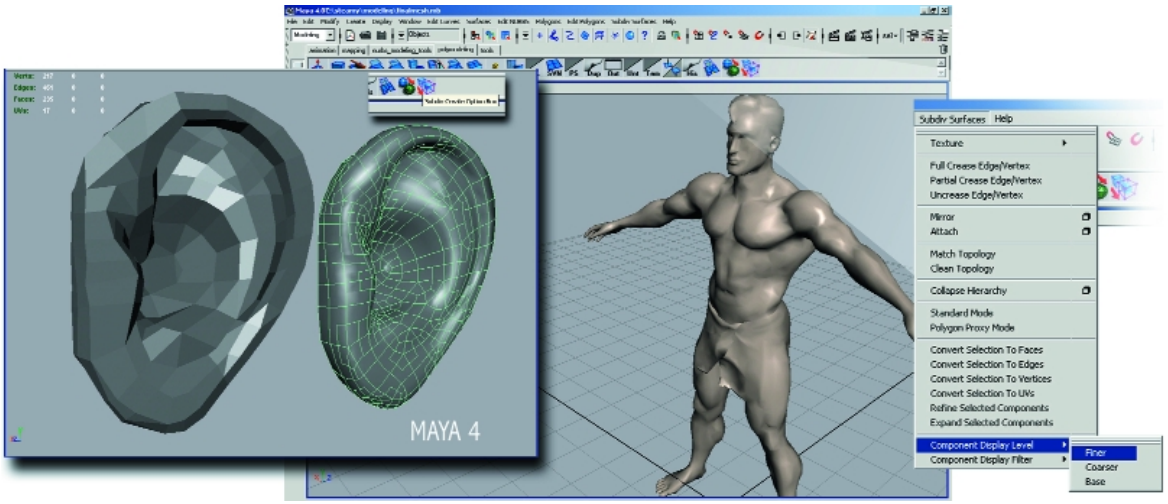


FIGURE 3. Build small isolated parts that can be easily assembled using both subdivision surface and polygon modeling tools. "Steamy Stud" is to be used in a real-time dancing music visualizer.

surface geometry as you would for a basic polygonal model. Using vertices and edges to create polygons, you can begin defining detailed areas of your subject in small, isolated parts. This is called the detail-out approach. Or, you can use standard polygon primitives such as cubes and spheres to begin defining the shape of your model, selectively adding more detail as needed. This is called the box approach or volume modeling.

However, organic modeling (such as modeling characters) is a process of creating objects which aren't easily constructed using just primitive shapes. I find that it's often easier to use both approaches. To be efficient, as with low-polygon modeling, you create only what you need as a coarse mesh representation of your object or character. The fewer vertices you use, the easier it is to make gross changes to your model's form throughout the process. You call upon vertex and edge manipulation tools to massage and refine your shape by pushing and pulling vertices, turning and splitting edges. You essentially create what is referred to as a control net, a control cage, or a control point lattice, depending on the 3D package you use. It is a simplified mesh, and the foundation for your subdivision surfaces (see Figure 2a).

Unlike the polygonal model and more like NURBS surfaces, a subdivision surface is smooth and can be shaped using relatively few control vertices. Subdivision surface schemes allow you to take the original polygonal model or coarse mesh and produce an approximation of the surface by adding vertices and subdividing existing polygons. This new model or approximation of your original can be as coarse or as detailed as you need. The relatively low-resolution control mesh is the framework for extracting the higher-resolution surface at the output (see Figure 2c). The output resolution or mesh density upon subdividing can be something you, the artist, control by simply defining a few values (like setting maximum edges per vertex) and clicking a button, or it can be programmatically determined by various schemes upon implementation into your game. Since the subdivisions are totally procedural, the smoothing algorithm can be handled in-game rather than by the artist in the authoring software. Either way, given one control mesh you created, a second and more complex mesh can be produced by a single subdivision step. This subdivision changes the surface of your model, making it smoother and more organic looking. This is the final result you get without the extra work of adding geometry yourself. The software takes care of that for you.

You Have the Tools

If you're currently working as an artist in the gaming industry, then chances are you already have a robust 3D package you're using to create art assets. Chances are also good that you have subdivision capabilities and features in your arsenal. These features are all essentially the same modeling tools and techniques that produce very similar results. Although subdivision surfaces are the easiest and most intuitive method for modeling free-flowing, organic subjects, like anything else in every 3D program, it also requires you to follow some simple basic rules to make the most of them. I will also review some of these in the following example models.

Lightwave 6 Modeler offers several methods for subdividing polygonal geometry. SubPatch surfaces (once called MetaNURBS in previous versions of Lightwave) are their version of subdivision surfaces and provide modelers many of the same features found in other 3D packages. Because of this powerful tool, I personally enjoy working in this program any time I need to create intricate organic models like the ones in Figures 1 and 2. I find Lightwave's set of polygon manipulation tools very intuitive and easily accessible.

To use SubPatch surfaces, once again you create a basic control mesh using standard polygon creation and translation tools. SubPatch will smooth an object dramatically with the original object acting as kind of a bounding box template for a slightly smaller, more rounded form. This is an adaptive process, meaning that locations in your object containing greater detail (more vertices and polygons) will have more detail in the smoothing process. The only stipulation is that you build your

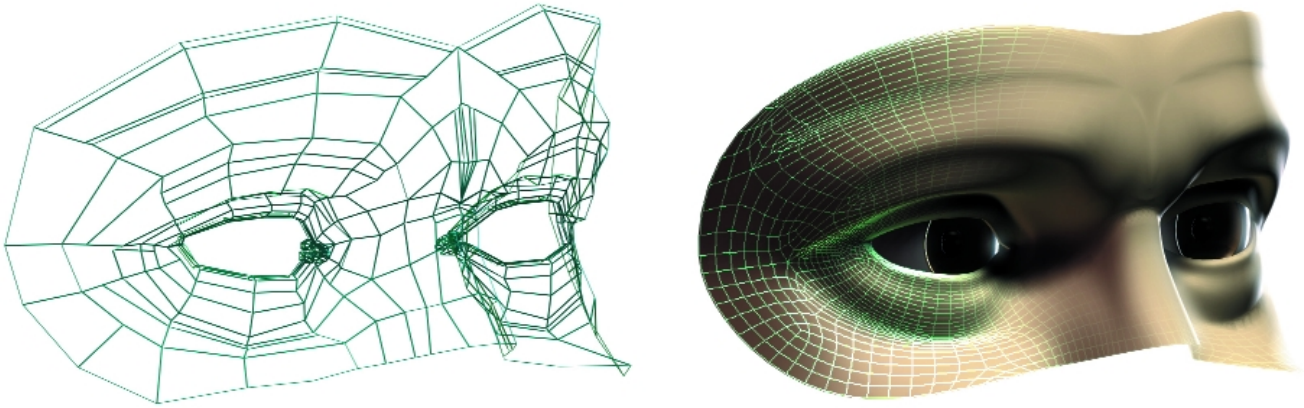


FIGURE 4. Modeled in 3D Studio Max 3 using MeshSmooth by David Johnson. The model on the right is a subdivision iteration of 2, using the control mesh on the left.

simplified mesh or original object primarily from quad polygons (polygons made up of four vertices) and triangular polygons. Viewing a higher-resolution subdivided version of your control mesh is as easy as pressing the Tab key in Lightwave. Figure 2b shows the results. To go back to your original mesh, just press the Tab key again.

In Maya 4 Unlimited, you can create subdivision surfaces from polygonal surfaces. You can then edit them with subdivision surface tools as well as polygonal tools, and also convert a subdivision surface back to a polygonal surface. You can create subdivision surfaces from NURBS surfaces as well. This doesn't even include the many subdivision tools Maya provides in a convenient SubDiv Surfaces menu.

I used Maya 3 Complete to model the original male model in Figure 3. Here I took the detail-out approach I mentioned earlier to create the various isolated parts of the character. These were done using a low-resolution base mesh that I divided further only as needed. Again, using fewer control vertices makes it easier to adjust or sculpt your shapes. I later attached it all together as you would for any polygonal model. I find that unlike working in NURBS or spline patches, I'm never limited by the need to maintain a rigid gridlike surface with a predefined number of endpoints that I have to match while attaching the various parts. Nor do I have to worry about seams that are not visible in my viewport but might show up later when I deform my mesh for animation or during rendering of the model. In attaching polygonal objects together, you simply weld their control vertices without the need to add more vertices or faces between the objects. With subdivision surfaces the smoothing at the seams is nicely addressed for you. You can control the curvature or tightness of a surface at the seam by increasing or decreasing the number of polygons (and thus vertices) in the control cage mesh.

Unlike true subdivision surfaces found in the 3D packages I've mentioned, you can emulate subdivision surface workflows with Maya Complete. There are well-established techniques that are too involved for me to cover in this article. Several tutorials are available on the Internet that do a thorough job of going through methods which produce similar results.

Here are a few other things to consider. When creating the human body or face using subdivision surfaces, following some basic guidelines will make the job of shaping the forms much easier. The density of the control mesh relates to the curvature of the underlying shapes. In Figure 4, notice that fewer control vertices were used to define flat or smooth areas of the forehead. Aim to create your model with just enough polygons to define the shape you want in each area (this often takes a bit of trial and error). Try to get the gridlike structure of your mesh to follow the contours of the underlying form. Doing so leads to a more efficient use of geometry.

Summary

As 3D character modelers faced with changing artistic demands and improving technology, you will naturally need to mature your modeling methods so that you can continue to create desirable content of greater sophistication and subtlety. In an industry that always promises to deliver more in record time to its audience, game developers need to leverage the skills they already possess without falling behind or blowing their production schedules. Of all the various modeling methods that produce high-resolution smooth organic models, subdivision surfaces provide a smooth transition for modelers who are already familiar with low-polygon surface modeling tools and techniques while offering some of the best benefits of NURBS.

Regardless of the 3D package you use, subdivision surfaces are a very useful method for creating scalable objects and characters out of simple geometric structures. The resulting advantages are so great that knowledgeable art directors would consider the effort of learning subdivision surfaces as time well spent by their art staff. My hope is that you are at least curious enough now to consider exploring this method in future professional or personal projects, as I have. Go conquer and subdivide.

FOR MORE INFORMATION

Sharp, Brian. "Subdivision Surface Theory" (January–February 2000).

Guymon, Mel. "Skin Deep: Surfacing Strategies for RT3D Characters" (Artist's View, March 2000).

Playing

Illustration by Dorothy Remington

by Ear

Using Audio to Create Blind-Accessible Games

Have you ever played a game with a configuration option to turn off the graphics? I'm not talking about an option to turn down the level of detail or switch off textures, but to turn off the graphics completely?

How many games have you played with options to turn off the sound?

Most people can't imagine playing a videogame with no graphics — even the name videogame indicates that they're a visual activity. At Zform, we've decided to be different from most game companies. We're developing games with parallel graphical and audio user interfaces (GUIs and AUIs). In our case, we're doing it because we want to bring the excitement of online multiplayer competition to visually impaired people around the world.

There are over 7 million people in the U.S. that can't see well enough to read this magazine article. Many millions more need to find their glasses to read it. The percentage of the population that has trouble seeing is getting larger every year as the baby-boom generation ages. If you'd like to sell your game to the largest possible number of people, you should think about using audio to reinforce the information you present graphically.

Another area where audio interfaces shine is on nontraditional gaming platforms such as mobile phones or PDAs. Perhaps the next blockbuster gaming platform will be audio-based games running on portable MP3 players. After all, MP3 players have all the requirements of a good gaming platform: lots of memory, a fast CPU, high-quality stereo sound, and several buttons for user input. The lack of a high-resolution color display shouldn't impede a creative game designer.

So what if you're not creating games for visually impaired players? Even if you are creating another first-person shooter with a target demographic of able-bodied 18-to-34-year-old males, you should still consider using audio for more than just gunshots, grunts, and death screams. No matter what type of game you are creating, paying careful attention to the audio user interface and 3D audio environment will enhance the player's experience.

Technology Platform

In this article, I'll be describing the techniques we use to create an audio user interface for a first-person 3D game we're developing. Our goal is to create an interesting, compelling 3D environment in which both blind and sighted players can compete as equals.

GAVIN ANDRESEN | *Gavin started his technical career in 1988 at Silicon Graphics. He was part of the core development team for the Open Inventor software toolkit, and led the VRML specification effort.*

Gavin co-founded Wasabi Software after leaving SGI in 1996, where he created SkyPaint, a tool used by game developers to create 3D panoramic backgrounds.

He joined Resounding Technology as head of development in 1999. There, he led the team that created Roger Wilco, the voice-chat-while-gaming technology used by over 500,000 online gamers.

Gavin is now the technology director of Zform, leading a team of programmers creating games that let blind and sighted gamers play online games together as equals. Gavin can be reached at gavin@zform.org.

We decided to use the QUAKE 1 engine as our technology base, for several reasons. First of all, older technology is great because it runs great on older machines. Blind folks usually don't have the latest and greatest PCs with state-of-the-art sound and video cards. Our target system is a 200MHz Pentium with VGA graphics and any DirectX 7–compatible sound card. The second reason we chose QUAKE 1 is because it's open source. Kudos to id Software for making it available as a starting point for innovative projects. Finally, we have the source code. We knew that no matter what engine we chose, we'd have to make lots of modifications to the audio and navigation code to create a blind-accessible game.

All of our audio is created in 22kHz, 16-bit format and played back in stereo via DirectSound. We assume that our blind players can hear stereo sound (that they're not deaf in one or both ears).

2D Audio Interface

Our first task was to make all of the introductory menus and text both audible as well as graphical. A little bit of programming extended the menu and option GUI to play back arbitrary sound files, instead of making the generic QUAKE "clank" sound. It was simple to record somebody reading each of the menu entries so that each entry is identified aurally when selected. Some of the game options were trickier than others, such as entering an IP address to set up a multiplayer game, but none was too difficult.

One simple rule we followed that many other games do not was to make narrations interruptible. This was especially important for our audio menus; it's no fun to listen to six options play back when you know you want the seventh.

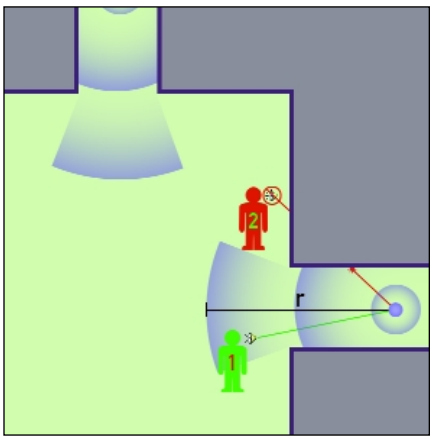


FIGURE 1. This point source audio entity, with an attenuation radius of *r*, cannot be heard by player 2 due to occlusion.

Speaking of narrations, another thing we did that was very effective was to use the game's main character voice for all of the game's menus. Our main character, Momo the monkey, has a distinct, silly accent. Using Momo's voice for the initial game-setup menus was a great way to introduce the player to Momo and to set the right mood for the rest of the game.

Navigating by Ear

Giving players enough cues to let them know where they are in the 3D world, but not so many that their ears are overwhelmed with sound, was our biggest challenge. There is a lot of information to convey:

- What exits are near the player?
- Is the player moving or standing still? Is he or she walking toward a wall or along a corridor?
- Is the player moving north, southwest, straight up, or straight down?
- What objects are around the player? What objects does the player possess?

The hardest pieces of information to convey audibly tend to be navigation issues that are nonexistent or trivial in a graphical interface, such as the location of the exits. For example, a simple solution for making exits easy to find in a graphical user interface is to make them look like familiar exits in the real world (such as doors and corridors).

Exits. Initially, we installed doors at all of the exits of each room in our level (gameplay occurs in an indoor environment). Our doors were of the electric *Star Trek* variety, automatically sliding open as you approach, so it seemed natural to have them emit an electrical humming noise. The idea was that when standing in the center of the room, you would be able to identify the exits just by the noises they were making; if you heard a hum to your left, you would know there was a door to your left.

That implementation was a complete failure. It was difficult to navigate out of any room with more than one exit, unless you cheated and peeked at the screen. If you stumbled around long enough you'd eventually hear a door make its opening "whoosh" sound, but even then it was hard to navigate through the doorway.

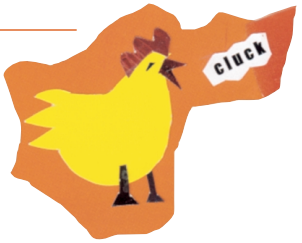
We tried several variations — we gave each individual door a slightly different sound, created doors that beeped instead of hummed, and tweaked attenuation parameters so you didn't hear doors until you were fairly close to them. None of them worked very well.

We finally realized that the doors weren't really what the player needed to hear. Blind players actually needed audio cues to navigate into the room or hallway that lay beyond the door. We ripped out the doors and installed air conditioning vents — point audio sources with a pleasant hum — into the center of each hallway, adjusting their volume and attenuation so that they could be heard down the length of the entire hallway and slightly into the adjoining rooms. We also modified the game engine so that walls occluded point audio sources.

After making those changes, finding exits by ear became easy. When you hear the air conditioning noise, you have an unoccluded path into the hallway. You then rotate yourself until the noise is coming from directly ahead (so it has the same volume in both ears). You can then simply walk forward into the hallway (see Figure 1).

This same technique can be used to make it a little easier for sighted players to find hidden passageways and rooms in a level. Observant sighted players will notice a quiet hum coming from their left as they walk by a hidden exit and will use the audio to navigate their way into the hidden area. This is one case where blind players would have an advantage; hidden passages would sound the same as any other exit.

Footsteps, bumps, and scrapes. As in many games, we generate footstep sounds as the player walks. They are a tried-and-true solution to the problem of giving players feedback on whether or not they're moving and letting them know how fast they're mov-



ing. In fact, it would have been more work to make movement silent, since footsteps are hard-coded into the QUAKE 1 engine.

We have, however, extensively modified the audible movement cues to make blind navigation easier. Originally, the game engine played a simple "ugh" sound when the player walked into a wall. Sighted players can easily see if they've walked directly into a wall and are stuck, or walked into it at an angle and are sliding along it. To give blind players the same information, we adjusted the stereo pan of the "ugh, I ran into a wall" noise based on the angle at which they ran into the wall. If you run into the wall with your left shoulder, you hear it in your left ear; walk straight into a wall and you hear it equally in both ears. We also added a scraping noise to indicate that a player is moving forward but contacting the wall. The scrape sound is stereo-panned in the same way as the bump sound.

Supporting artificial stereo panning did require us to modify the game engine's sound code. We added an extra floating-point parameter (balance) to the play-a-sound function. Zero is the normal setting, which plays the sound using the standard 3D stereo spatialization algorithm. A value of –1 results in the sound occurring completely in the left ear, and +1 results in a sound completely in the right ear. Of course, values in between pan the sound from left to right.

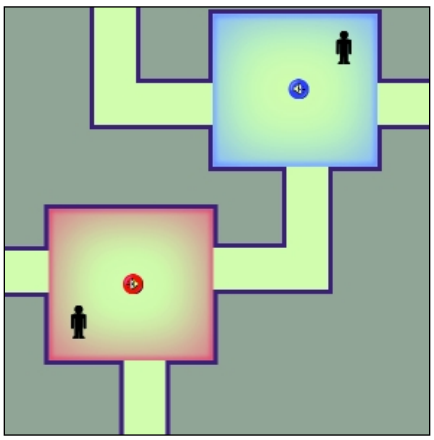


FIGURE 2. Room center entities create different auditory environments for each room.

Getting oriented. Letting players know which way they're facing is always a challenge when you allow unrestricted movement in a 3D world. As in many games, we simplify the problem by restricting movement to 2D movement along a ground plane. Therefore, the player's orientation can be described in north/south/east/west terms; players can't fly up and down. Even on a 2D plane, however, after a few turns down a series of passageways it is easy to lose track of which way you're facing. We've found that some of the same techniques help both blind and sighted players keep themselves oriented.

The most basic technique is to simplify level design. Unless getting lost is part of the game, avoid creating a maze of twisty passages, all of which look and sound alike.

Another technique we use is to build a consistent orientation cue into the 3D environment. For sighted players in an outdoor environment, that might be moss on the north sides of trees, or clouds in the sky that always move from west to east. In our case, we modified the hallway air-conditioning noises so that north-south-oriented hallways make a slightly different noise from east-west hallways.

We also bound a keyboard key to an audible compass. Pressing the key announces which way the player is facing, rounded to the nearest compass point (such as "northwest" or "south"). Implementing the audible compass was much easier than a graphical compass and gives the same information.

Audible objects. Besides exits and walls, the other objects that sighted players can see and that blind players need to hear are the other players (our game is multiplayer) and any object that can be picked up, be poked, or otherwise affect the gameplay.

The other players are easy to hear, since they're making footstep, wall-bump, and scrape noises as they walk around the level. We do implement special code so the artificial stereo panning of the bump and scrape noises (indicating the angle of impact) is only done for your own bumps and scrapes. As other players bump into walls, you hear their grunts as ordinary point sound sources, emanating from the point at which they hit the wall.

All of the other visible objects in our game are assigned a reasonable, regular idling sound so they are always audible. We've created a noisy, silly, fun environment, choosing objects that appeal to both the eyes and the ears. Walk around a level and you might hear chickens clucking, a grandfather clock ticking, and pigs squealing as they're picked up and thrown.

All of these noises are occluded by the walls of the level, which limits the number of sound sources audible at any one time and prevents blind players from trying to walk through walls to get to objects that they can hear but can't see. Sound occlusion is a wonderful thing. Our implementation silences sounds if the line segment from the center of the listener's head to the center of the sound source intersects any of the walls of the level. The result is not physically accurate, but works very well as a navigation aid and was easy to implement. Occlusion also prevents sighted players from becoming frustrated trying to find a path to an object that they can hear right next door.

Ambient noises. After working through the navigation considerations to allow blind folks to move around our 3D world, we then added audio decoration to make the world more interesting. We tried to give each part of the level a distinct character by adding audible landmarks. For example, you might hear the sounds of pots and pans clanking in the kitchen or hear a stately old grandfather clock ticking in the study. We added a generic `arbitrary_noise` object type to make it easy for our level designer and audio engineer to sprinkle interesting sound throughout the level.

We also implemented a quick and dirty form of environmental audio. If our minimum system requirements had allowed it, we would have used EAX environmental audio (more on that later). Instead, we created `room_center` objects and placed them around the level (see Figure 2). They are simply invisible boxes that the level designers used to mark out the various rooms in the level. One of the attributes of the `room_center` object is `footstepNoise`. By using different footstep noises for different rooms, we give the impression of the player being in different environments. A carpeted study has quiet footsteps, while a kitchen has sharp, echoing footsteps. It was easy to modify the game engine's footstep-playing code to play the appropriate footstep noise depending on what `room_center` object the player is in.

Sounds other than footsteps should also be affected by the sonic properties of the room. That's easy for any audio sources

that are part of the room; our audio engineer just "precompiles" the room's environment into the sound file. In our game, objects that can walk or be carried into rooms sound the same no matter where they are, which is unfortunate but not a huge problem.

Gameplay Considerations

After conquering the navigation problems, making all of the gameplay accessible via an audio-only interface was easy and fun. One of the goals in our game is to collect a set of objects, so we had to figure out how to tell players what they need to collect. We associate a name sound with each type of object. The name sound is just the narrator reciting the name of the object (for example, "chicken" or "water balloon"), so telling players what they need to collect is just a matter of stringing together a narrative introduction ("To complete the whatchamacallit, you'll need . . .") with the name sounds for each item on the list.

We also play the name sound when the player bumps into an object. This improves gameplay for both blind and sighted players, especially for objects that might look or sound unfamiliar. I played QUAKE 1 for several days before I figured out that the floating blue Q-like thing made me do more damage to enemies. I had never noticed the tiny text message "You got the quad damage" scroll by on the status bar; I would have been clued in more quickly if I had heard "Quad damage!" announced when I ran across it as happens in QUAKE 2 and 3, instead of a generic beep sound.

We could also use name sounds to implement an audible inventory, reciting the list of objects that the player is holding. However, we've chosen to limit the number of objects a player can hold to just two (one in each hand), so instead we just play the objects' idling sounds once when the inventory key is pressed. We put artificial stereo panning to good use again, playing the left-hand object's sound in the left ear and the right-hand object's sound in the right ear.

We follow a couple of general design principles to ensure our game is fully accessible to blind players. First, we make sure that if two items look different, they must sound different. That isn't usually a problem; most objects in the real world make unique sounds, if they make any sound at all. We just avoid populating our game with items that make no sound.

We also make sure that item or game state changes are accompanied by audio cues. For example, items make a "grabbed" sound when they are picked up. Pick up a chicken and you hear it squawk. While it's in your hand, it will make a disgruntled clucking noise, instead of its normal, "I'm a happy chicken" noise.

Stuff We Haven't Figured Out Yet

As I write this, there are still a few problems that we haven't solved and a few solutions that we haven't tried. The thorniest issue is the up/down, front/back problem.

We are using the simplest possible stereo spatialization algorithm for 3D sound sources, which makes it impossible to distinguish whether a sound source is behind or in front of (or above or below) the listener. Preliminary experimentation with the HRTF (head-related transfer function) algorithms built into DirectX is discouraging — the more complicated algorithms sound better but aren't good enough to tell players whether objects are ahead of or behind them. We are currently experimenting with nonrealistic techniques to indicate the fore/aft position of objects with respect to the listener. Since our game doesn't require unrestricted, up-and-down 3D movement in order to be fun, we're not going to do anything to indicate the up/down position of objects.

Player-generated text, such as player names or text chat, is a problem for which we don't yet have a solution. The standards for accessing text-to-speech functionality under Windows are just emerging, so even though we can assume that all of our blind players have a speech synthesizer for converting text into speech already installed on their systems, we have no way of sending text to the synthesizer to be spoken. We may end up licensing a synthesizer to include with the game, but for now we're simply avoiding features that would require text-to-speech conversion.

We intend to reintroduce doors to the game, because the simple mechanisms of allowing doors to be open or shut and locked or unlocked will add strategic elements and make the game more interesting. When we do, we will probably modify the occlusion algorithm so that closed doors muffle sounds coming through them. That, combined with hallway and room noises, should solve the problems we had earlier with blind players being unable to find the exits in the level. We will still have to figure out how to tell a blind player there is an open door nearby that can be closed or locked.

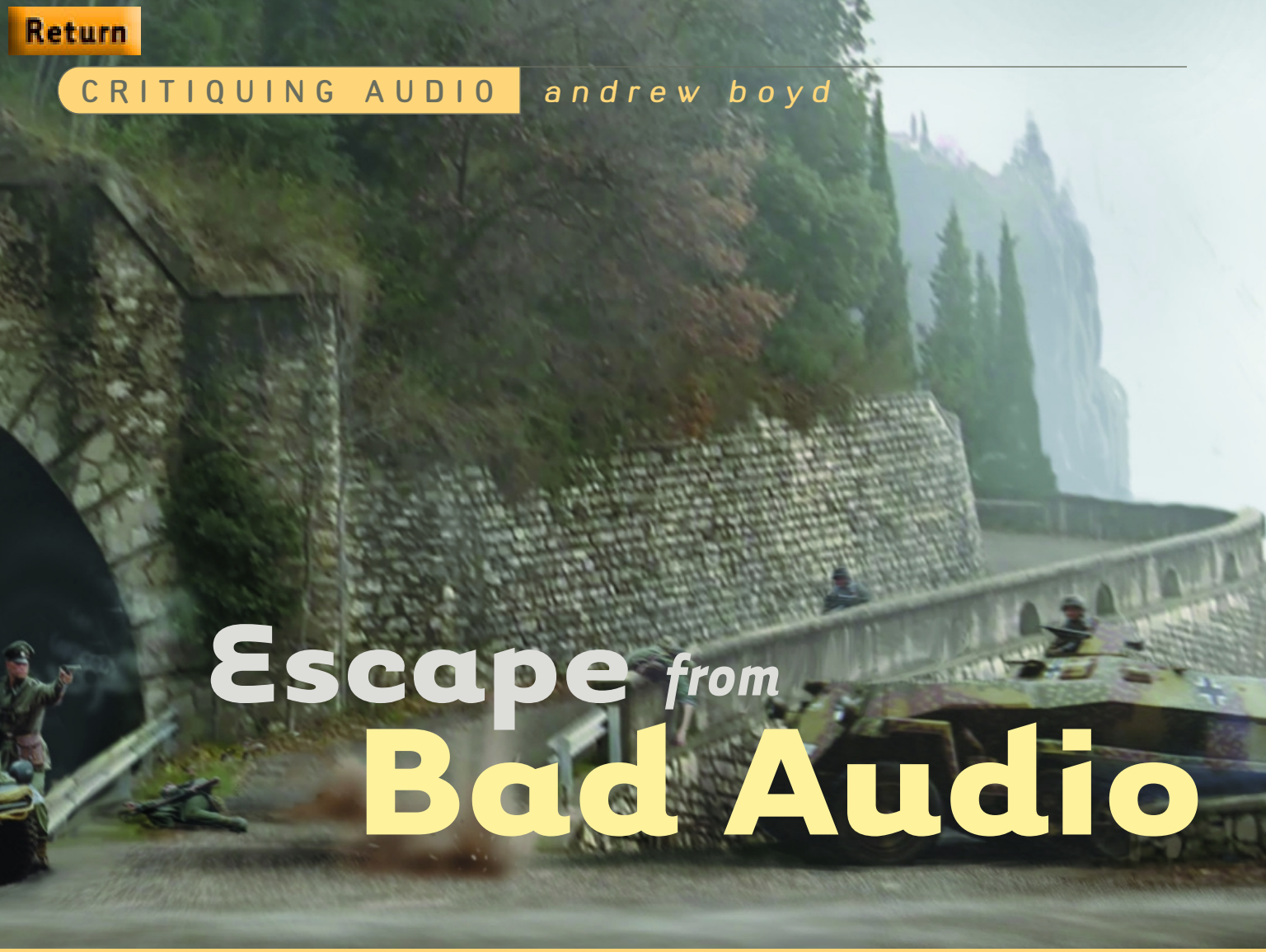
As mentioned earlier, we are using a quick and dirty hack to approximate true environmental audio. Implementing EAX environmental audio is on our list of things to do, but has been a low priority because we can't assume that our players will have an EAX-capable sound card. We think that supporting EAX will increase the quality of the game, but don't think it will improve accessibility or gameplay.

Better Games for Everybody

Oxo's Good Grips brand kitchen tools were designed for people with arthritis or other joint problems (see www.oxo.com/eyeonoxo for the full story). They've been hugely successful selling them to able-bodied people. I don't have arthritis, but I own their potato peeler, ice cream scoop, and cheese grater. Designing products that work for people with disabilities creates products that work better for everybody. All of the techniques we've used to make Zform games blind-accessible can be applied to any game. None of them makes the game any harder for sighted people to play; on the contrary, most of them either help to reinforce the graphical interface or make the game more interesting and fun.

ACKNOWLEDGEMENTS

Matt Goetz, Tim Keenan, and Chris Sulham ("The A-Team" at Zform) implemented most of the accessibility techniques described in this article. Paul Silva, Jeremie Spitzer, and the author helped with much of the brainstorming.



Escape from Bad Audio

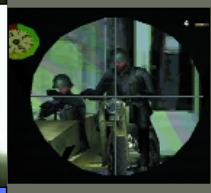
Fewer and fewer games seem to ship without the claim that the sound designers and composers that worked on it are not just your ordinary sound designers and composers, but are a superior breed: *award-winning* sound designers and composers. But what does it really take to win an award for game audio, and what does it mean? Is it anything more than PR nonsense? We're all striving for it, but do we really know what the judges want to hear? Are the judges even focused on the same things as our audience? Are there things we can do to maximize the potential of our games to win awards if we decide that it's our goal? I decided to listen to the winners of a few prestigious awards from the last few years to hear what I could find.

The Assignment

In order to divine what the judges who pay tribute to game audio consider in each year's offering of games, I had to produce some real analysis of the sound in the games at hand. Listening is a subjective practice by nature, but I've tried to listen to the games cold, as it were, and report what I hear. I'm certainly not trying to second-guess or dispute any judges' findings, but my reactions may in some cases be surprising if these awards carry a great deal of weight to you. And I encourage you to think for yourself about what awards like these mean to you and the industry at large, and what, if any, effect they have on your work.

Also, the world doesn't need one more article lamenting the troubled history of game audio and predicting its limitless potential for the future if only the industry would wake up and take it seriously. So I won't bore you with the usual rigmarole, but let's get some assumptions out of the way up front. I'll assume that if you are reading this, you're likely already well versed in all that stuff and have formed many of the same opinions as those that persist among our colleagues. For example, game developers and critics should: recognize the fact that audio is at least half of the game-playing experience, learn that audio costs money and time but that investing in it will reap dividends in sales and prestige, understand that music and sound operate on unconscious levels that focus groups and the general public will

ANDREW BOYD | *Andrew is the audio director at Stormfront Studios. Drop him a line at aboyd@stormfront.com.*



never be able to articulate accurately, acknowledge the contributions of game audio to the general state of the art in mass-media production, and, last but certainly not least, make the programmers and artists answer to us once and for all.

Let's also assume that those of us who produce game audio professionally have been working for a long time to realize these goals and more, and that we don't need to be told again and again about them. Let's propose instead that we who toil for a better auditory experience in games would like to be recognized for our efforts, and would like the chance to recognize others in the field for their contributions. To do this we've formed industry associations and use them to award exemplary work with commendations.

This is a complicated business, recognizing individual contributions to a multi-disciplinary product such as a computer game, and a number of awards now exist to handle different aspects of it. Most magazines give some form of awards every year, as do many online publications. Larger publishers and developers give awards to their in-house teams and contractors. And several trade organizations, such as the Academy of Interactive Arts and Sciences (AIAS) and the International Game Developers Association (IGDA, formerly the Computer Game Developers Association), also give awards in the same way that the film, television, and music industries have trade organizations that recognize contributions to their respective fields. (The IGDA is an independent nonprofit organization under a management contract with CMP Media, publishers of *Game Developer*.) In most of these cases, audio is a discipline that is individually recognized — sometimes music and sound are even called out separately. The games that I listened to for this article represent a sampling of awards from the last few years:

- Road Rash 3D for Playstation 1 (AIAS 1999, Sound and Music)
- Half-Life for PC (CGDA Spotlight 1999, Best Use of Audio)
- Um Jammer Lammy for Playstation 1 (AIAS 2000, Original Musical Composition)
- Medal of Honor for Playstation 1 (AIAS 2000, Sound Design)
- Diablo II for PC/Macintosh (IGDA Game Developers Choice 2001, Excellence in Audio)
- Medal of Honor Underground for Playstation 1 (AIAS 2001, Original Musical Composition and Sound Design)

SPREAD. Medal of Honor Underground concept artwork. INSET SCREENSHOTS (from top). Medal of Honor, Half-Life, Road Rash 3D, Medal of Honor Underground, Um Jammer Lammy, and Diablo II.

A quick glance at this list shows that the variety is impressive, both of game type and soundtrack styles, but a couple of trends jump out immediately. Note that the AIAS awards all went to Playstation games, and the IGDA/CGDA awards went to PC games. In fact, only Playstation and Windows games are represented at all (apart from DIABLO II being a hybrid Windows/Macintosh CD) — nothing from the other big players of the past few years, including Dreamcast, Nintendo 64, or Playstation 2, is to be found.

No obvious commercial failures are present either, though the rules of these awards do not stipulate commercial success as an entrance requirement. This could be because good audio is enough to make a game successful. Or it could simply be because games with audio good enough to win a major award are always so good that they are also major commercial successes. Of course, it could be that the syllogism that says that awards are industry driven, the industry is money driven, and thus awards are money driven is true. The truth is probably somewhere in between.

The Methodology

I listened to all the games in a controlled environment using quality equipment — the PC games I played on a 1.2GHz Athlon Windows 98 SE machine with a Sound Blaster Live! card, and the Playstation games on a PS2 (I didn't have access to an original Playstation for this article). I monitored through Mackie HR-824 studio monitors and Sony MDR-7506 headphones.

This brings up another long-standing debate in game audio about lowest-common-denominator mixing, which I will touch on very briefly. This is a battle fought between two camps. One adheres to the dictum that since the majority of game players have lousy OEM sound systems, games should be mixed to sound as good as possible through these setups. The other camp testifies to the gospel of the absolute quality of the game's sound — let the player's experience be controlled by his or her investment in playback hardware. Be advised that I fall very much into the latter category. I believe that players who care about audio will have a good sound system hooked up and should be rewarded for their investment. If players don't bother to upgrade the 49-cent speakers that came with their PCs, or don't bother to route their consoles through their stereo system, they clearly don't care about audio and therefore won't care if the game's been mixed for their system. In any event, the listening I did for this article was very much informed by and dedicated to that belief. So I put the games in, and here's what I heard.

A Closer Listen

ROAD RASH 3D. In 1999, the AIAS gave its Best Sound and Music award to ROAD RASH 3D. It's hard to know from the title of this award what exactly was being recognized, but the soundtrack to this Playstation motorcycle racing game was unquestionably effective. Pop it in now (especially if you have fond remembrances of its audio component) and you may be surprised at the audio quality, though.

The fidelity is remarkably poor — the music is dull and compressed, the player-bike engine sound is thin, buzzy, and quite artificial sounding. The player-character vocalizations, limited for the most part to pained grunts and gasps, are repetitive and truncated. The skids and bike crashes are pretty good, and the Doppler effect on competing bikes is convincing, but overall the limitations of the Playstation's audio system were quite clear. I was tempted to approach listening with an attitude of "well, it's pretty good for PSX." I soon found that this kind of relativism, whereby audio quality is judged against the competition rather than against some kind of absolute measure, would permeate all of my considerations of these award-winning games.

ABOVE. Screenshot from ROAD RASH 3D.

Given its fidelity issues, what makes ROAD RASH 3D's sound worthy of winning a prestigious award? First, of course, is the fact that while the game suffers from myriad problems resulting from the low-memory, low-bandwidth audio that the Playstation necessitated, every Playstation game at the time suffered from the same problems. And the award is, properly, "Best Sound and Music," not "Perfect Sound and Music." Factor in that this was a high-budget, high-profile, successful game from a formidable publisher, and it is clear that it must be very enticing to an awards committee to want to give it some recognition. It is easy to grow cynical about this, but let's be fair — ROAD RASH 3D does some things very well, indeed; maybe it earned that award after all.

First there's the music. This is not a game that tries to be more than it is. It's about jumping on a motorcycle, going as fast as you can, and beating the crap out of your opponents while doing it. Clearly, the appropriate music for such a game should be high-energy, angry, loud, raucous, and rebellious, right? The music for ROAD RASH 3D is all this and more. Instead of creating a custom soundtrack, the developers licensed songs from Kid Rock, CIV, and Sugar Ray (who play much harder than their recent hits would indicate they might). This was a key move. I'm certain that if ROAD RASH 3D had featured music of the same style but done by some talented but unknown in-house staffer with no preexisting credibility, for instance, it would not have garnered the critical attention that it did.

The award was probably not so much for the style of music (since that was a predictable choice), but for the truth of the music. And for that kind of music to be true, it cannot be purpose-made — rather, it must be "real" music. This musical soundtrack does what it sets out to do, and does it well, no more and no less — it sounds like a fairly hip mix tape circa 1998. That accomplishment is no small feat, and it is my guess that it won the game the award.

And, despite their limitations, the sound effects do manage to help bring the game's world to life. The engine, though thin and whiny, is quite responsive to player input and bike speed. The tire squealing builds in a believable way. The impacts and slides that result from a wipeout have good

punch. The occasional horn honk from opposing traffic is hilarious, a wonderful touch. So, while there are faults to be found with the audio in this game, it's not hard to imagine a thoughtful group of judges feeling that this game earned their commendation.

ABOVE. Screenshot from HALF-LIFE.

HALF-LIFE. The same year that ROAD RASH 3D won the AIAS's award for Best Sound and Music, HALF-LIFE took home the CGDA's Spotlight Award for Best Use of Audio. There seemed to be a general consensus in the industry that HALF-LIFE was one of the best-sounding PC games ever at that point.

Here's an exercise: listen to the sound without letting the gameplay immerse you, maybe record the audio output and listen to the recording without seeing the game. You'll find the sounds are uniformly noisy and gritty, the footsteps are repetitive, the loops are obvious, and there are no real high frequencies to be found. The voices of the various interactive characters that can be found throughout the levels are heavily compressed and repeat awkwardly, and often consecutive lines sound very different from each other. What little music the game features is ambient, electronic, slightly industrial sounding, quite intriguing texturally, but pretty boring musically. And yet, with the game played as intended, no one would deny that the audio is brilliant. Again, fidelity clearly takes a back seat to other factors, but what are they?

Listening closely, and with an ear trained by many large-scale project productions, I was primarily struck by the sense that the overall soundtrack of the game was really clearly thought out. I don't necessarily agree with some of the choices, but everything about the game's sound and music seems quite intentional and premeditated, as if the project had been planned by someone with a clear vision of the final sound right from the beginning. The music doesn't need to be very strong musically, because its function doesn't require it — there is a great deal of combat in the game that is plenty exciting itself, and it doesn't really require a big soundtrack to pump it up. The rest of the time the desired feeling is one of disoriented creepiness and tension, which the abstract music here creates perfectly.

The sound effects, too, exhibit this kind of consistency and thoughtfulness. For the most part, while the footsteps themselves don't sound very good, they do sound as if they represent the materials upon which the player character is walking. And swing a crowbar against a wall or a window or a door and the sound is a little different for each. None is a spectacular sound, but the attention to detail indicates that a conscious decision was made to trade off the quality of each sound for the variety of all sounds, as a method to immerse the player further in the experience of the game as a whole. So the sound and music tend to play a supporting role here, and the judges apparently recognized this fact and rewarded the extent to which this was successful, despite the reality that the fidelity of the sound had to suffer in the process.

MEDAL OF HONOR. The first thing I noticed about the sound in MEDAL OF HONOR, the winner of the 2000 AIAS Sound Design award, was the sense of depth and space it creates. Eschewing the ability to adapt the sound directly to gameplay, the developers seem to have used long, streaming sounds for the audio backdrop. The risk with this type of system is that players can feel separated from the action — that their actions have no effect on the world of the game. In this case, though, the benefit is that the sound creates the impression of a world well beyond what players can actually see for themselves (and thus, logically, beyond what players can affect): Somewhere over that next ridge another fight is raging, better duck so the passing planes don't spot me.

ABOVE. Screenshot from MEDAL OF HONOR.

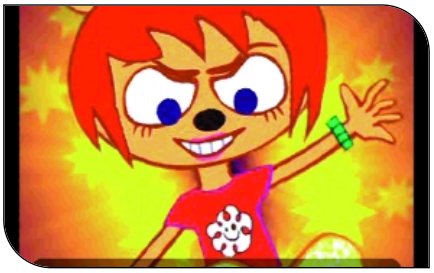
One of the most remarkable things about the ambient sound in MEDAL OF HONOR, considering this depth and spread, is that it is monaural. While there are occasional one-shots that appear panned around the stereo field, and the sounds of specific enemies are located according to their position in the world relative to the player, the basic background sounds are always right in the center. Still, they manage to communicate the sense of a huge, active, and very dangerous world surrounding the player. Start a level over and over and you'll quickly notice that the same sound effects are present in the background loop. But it doesn't detract from the sound's effectiveness, and that fact itself is worthy of note.

On top of all this are very finely made weapon and combat sounds that provide the immediacy and focus that really allow the ambient sound to do its job. One interesting decision to note: the player character does not make footstep sounds as he runs through the world. This is a great choice, as the footstep sounds would likely have suffered the same fidelity problems as those in HALF-LIFE, and that would have taken away from an otherwise very clean sound set.

Behind these foreground sounds, and expertly mixed with the ambient bed, is the musical soundtrack, a sweeping and adventurous orchestral score. This score works extremely well for a number of reasons; not least is that it has the richness and texture that only real orchestral performance can create. The music is of a style consistent with Hollywood's precedents for 1940s-era war movies, and as such it develops a strong feeling of time and place. But it also blends well with the ambient sound as it follows the same sort of architecture. It is made of long pieces that have their own sonic and musical integrity, even outside the current happenings in the game. In some ways this seems to communicate the sense of a big world operating by its own rules and logic that you, the player,

are trying to find your way through. This is precisely what the game needs, and this helps tell a bit more of the story and make the game feel like more than just another shooter. I wonder if it wasn't too subtle, though, to win the award that year for music, or if the next game I listened to was really just that much better.

UM JAMMER LAMMY. In the book *Overtones and Undertones: Reading Film Music* (University of California Press, 1994), Royal S. Brown proposes that a key use of music in films can be to, as he calls it, "narrativize" a scene. In other words, there might be more to a scene than the visuals and acting and script are communicating, and the music can actually work to tell the rest of the story. This concept applies very well to game music, too, and MEDAL OF HONOR is one example of this. However, the general approach in game music, at least conceptually, has been a movement toward adaptive (also often called "interactive") music. In this way, it is assumed, the music can always be appropriate and always be communicating "the rest of the story," regardless of what is actually happening in the game.



ABOVE. Screenshot from UM JAMMER LAMMY.

Now, as I wrote in the Soapbox column in this magazine last year ("Pay No Attention to the Orchestra Behind the Curtain!" September 2000), there are a number of issues to be taken with this — not least of which is that "interactive" music and "adaptive" music are wholly different concepts which cannot be used interchangeably. To quickly reiterate the differences (as I see them), an adaptive music soundtrack follows the action of the game and, well, adapts according to a set of rules. Interactive music, on the other hand, which has mostly been confined to academic circles and installation art, is music that the listener interacts with directly. In the case of a game soundtrack, usually the last thing a developer wants is to have the player interacting directly with the soundtrack. MEDAL OF HONOR avoids this dilemma by letting the music follow its own musical logic, rather than that of the game. But in the case of UM JAMMER LAMMY, the winner of the 2000 AIAS award for Original Musical Composition, interacting with the music is the primary gameplay mechanic.

The musical soundtrack of UM JAMMER LAMMY is split into two parts: the gameplay music, wherein the player controls the musical performance of a character on screen; and the cinematic, or noninteractive music, that underscores the bizarre, hallucinogenic transitions between levels. This game is sort of a sequel to the sleeper hit PARAPPA THE RAPPER, in which the player controlled aspects of the character's rapping. In this case, you control the guitar playing of a talented young female lamb by tapping the correct buttons on the Playstation controller in time to the music being performed.

While there is a strong sense of continuity and cohesiveness between the interactive and cinematic sections, they are clearly distinct. The gameplay music is an odd hybrid of Japanese pop, funk, and rap not wholly unlike PARAPPA THE RAPPER's music, but a little catchier and just a little less cool. The cinematic music is fully Carl Stalling cartoon style (Carl Stalling was the composer for Tex Avery and Chuck Jones cartoons). Orchestral strings and sound effects sit on top of strong beats and more catchy tunes.

ABOVE. Screenshot from MEDAL OF HONOR UNDERGROUND.

The game supplies feedback to the players on their progress through the music as well. While playing the game, you are trying to make the songs sound as good as possible. If the songs start to sound weird (though they're certainly pretty weird to begin with), you know that you're doing something wrong and you should try harder to improve. The guitar playing falls out of time, of course, if the player isn't keeping up. But beyond this, the song becomes progressively more distorted, filtered, and otherwise funked-up as more and more beats are dropped. It's true that this feedback is not strictly auditory, as there is a strong visual effect tied to it (the entire screen is modulated by a sine wave). But you ought to get the idea from the sound; if you do manage to improve, you get a "crowd cheering" sound to congratulate you, too. This kind of integration is handled with skill and subtlety and is clearly what earned the game the award.

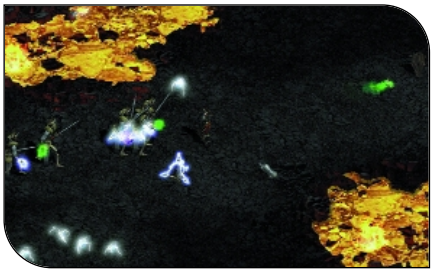
MEDAL OF HONOR UNDERGROUND. The 2001 winner of both the Sound Design and Original Musical Composition awards from the AIAS was MEDAL OF HONOR UNDERGROUND, the sequel to MEDAL OF HONOR. Obviously, the developers of these titles' audio are onto something. MEDAL OF HONOR UNDERGROUND takes much the same approach with its soundtrack as MEDAL OF HONOR, which is a great choice considering the quality and success of that game's sound.

UNDERGROUND features the same kind of mono ambience with panned effects on top. Here the sounds seem somehow more crisp and sharp; perhaps the memory allotment was more generous the second time around; perhaps the sounds were prepared somewhat differently; perhaps there are just fewer sounds and they are of a higher resolution than before. In any event, the sounds are all clean and elegant, and the game sports a surprising number of unique sounds. That's just plain hard to do on a Playstation. Otherwise, my thoughts on this game's audio are the same as for its predecessor. This game is an obvious winner, where originality, effectiveness, and fidelity marry well — a class act all the way through.

DIABLO II. Also in 2001, the IGDA's Game Developers Choice Award for Excellence in Audio went to DIABLO II. Though the award's title suggests that it is a general audio award, the recipients listed (the composers) indicate that the award is in recognition of the game's music. Listening to the game, it's easy to see why — the music's immediately accessible but unique sound stands out from the loads of

silly techno and boring synthesized orchestra music that suffocate our industry. In a sea of predictability and copycat soundtracks, this originality alone should earn the game kudos. While ROAD RASH 3D's music seemed perfect for its genre, forcing the developers to be extremely careful so that it did not feel trite, DIABLO II's music takes wonderful chances that give it a great deal more flexibility — which it uses to great advantage.

DIABLO II's music oscillates between dramatic orchestral passages (that sound synthesized but are well done nonetheless) and a kind of ambient progressive rock. It's like Michael Kamen meets Pink Floyd meets Harold Budd, and overall it's really quite effective. It manages to walk a very fine line between being recognizable for its genre without becoming cliché, grand without being too pretentious (though, truthfully, it is pretty pretentious), and consistently interesting while staying in the background.



ABOVE. Screenshot from DIABLO II.

The use of electric guitar is especially interesting, as most game soundtracks that are not strictly "rock" shy away from guitars, and certainly most soundtracks for games that take place in the fictionalized Middle Ages wouldn't go near an electric guitar. Instead, it seems that most games of the hail-fellow-well-met variety are interested in sounds that are either "accurate" to their time period (usually manifested in the strolling minstrel sounds of lutes and recorders), or hint at some Gothic representation thereof, with Gregorian chanting and lots of church bells. But here the guitar doesn't seem anachronistic, nor does it seem to pull the soundtrack too far into the rock genre. Instead it provides a good sonic hook — the game-playing audience is certainly accustomed to the sound and voicing of guitar — and an interesting and unusual texture that really helps keep the music intriguing for long periods of time.

Besides the music, though, DIABLO II's audio is good, but unremarkable. Sound effects consist of basic ambiences and general Foley — footsteps, grunts and groans, sword swings, and the like — with some interesting spell effects and nicely done voice acting thrown in for flavor. Nothing is missing, but it's all rather predictable.

Upon first listen, DIABLO II's cinematic sequences sound quite nice, but the impression does not survive closer scrutiny. In the opening movie, for instance, the sound effects that form the ambience of the madhouse are impressively wide and deep. But the mix was clearly aimed toward very limited dynamic range, and when large foreground events happen, they do not seem loud enough, or forward enough in the mix. The exaggerated stereo field also means they never quite feel anchored to the screen. The movie sounds heavily equalized, too, like the developers have attenuated between 500Hz and 2kHz, and boosted around 100Hz and above 4kHz. While this does leave room for the voice to sit into the mix, it also is a trick used to compensate for the mid-range heavy tendency of cheap speakers — a sound that seems to have become associated with cheapness, even when listening to a mix through higher-end systems. As I say, it works upon first hearing, but listen deeper and it feels as if some substance has been taken away. Overall, the sound does what it needs to do, but does not prove the accomplishment that the music does, and the award does seem to recognize that.

Making the Grade

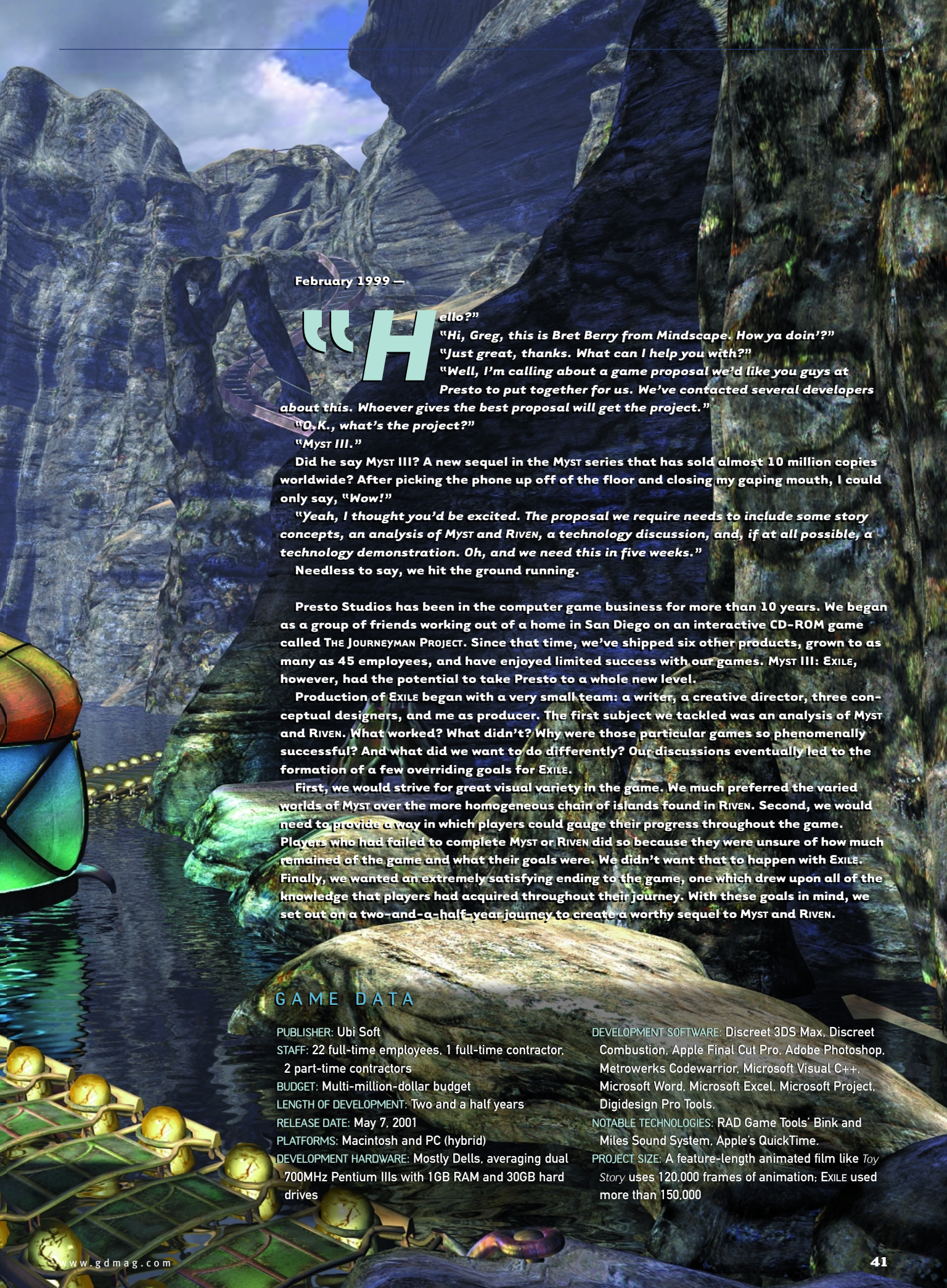
So how do I win one of these awards? At the risk of sounding overly cynical, it certainly helps to have a big hit on a popular platform and a game that's been critically well received in general. Besides that, though, there certainly must be some binding thread that runs through these games and makes them worthy winners. The awards committees and voters have clearly spoken on some matters: there is no one particular musical or sonic style they are looking for over another — in three successive years the AIAS gave their award for music to punk/thrash songs, weird Asian pop/funk/rap, and orchestral music. Also, there does not seem to be one genre of game that is more likely to win than another, though the MEDAL OF HONOR series has done a nice job of sweeping the AIAS's Sound Design award the last couple of years. However, given the other awards here, I doubt that this streak speaks to an affinity for military sounds as such, but rather recognizes the impressive achievement of these particular titles. There does not seem to be a consistent endorsement of one technology over another, either, as none of these games, as good as they are, seem to demonstrate any breakthrough techniques or processes.

Absolute fidelity has little influence on whether a game soundtrack is considered good. Much more important, apparently, is the sense that thoughtful choices were made between fidelity and interactivity, between the limits of the technology and the gameplay experience. Also, the contributions that the soundtrack makes to the overall game experience seem much more important than the audio itself — which may contribute to the fact that only successful games have won. They were successful game experiences, and therefore likely to indicate to a thoughtful listener that the audio played a part. The judges of these awards are obviously looking for clarity and cohesiveness in the overall sound experience, much more than in the sounds themselves. It doesn't seem to matter whether you choose to create orchestral music, punk rock, or new hybrid-pop, as long as it fits with the game and stays true to itself.

But I have to wonder if the judges aren't willing to cut game audio a bit of extra slack because of all the problems that we know are inherent in the discipline. And after playing all these games, while they clearly represent a phenomenal body of work, I have to wonder if the consistent demonstration that sound quality, as it were, does not include actual fidelity, might be a disincentive to ever improve the fidelity to the levels possible in other media. In any event, it does become clear that the judges have a good track record of thoughtful awards, and I have little reason to believe that won't continue.

Presto Studios' Myst III: Exile

GREG UHLER | *Greg is the producer for Myst 3: Exile at Presto Studios. He is a founder of Presto, serves as vice president, and was the lead programmer on Journeyman 1 and 2. In his spare time, he enjoys racing his car and sneaking his family into every project he works on: his father Ray in Journeyman 1, 2, and 3, his wife Catalina in Journeyman 2, and his daughter Audrey as Yeesha in Exile. He can be contacted at greg@presto.com.*



February 1999 —

"*Hello?*"

"*Hi, Greg, this is Bret Berry from Mindscape. How ya doin'?*"

"*Just great, thanks. What can I help you with?*"

"*Well, I'm calling about a game proposal we'd like you guys at Presto to put together for us. We've contacted several developers about this. Whoever gives the best proposal will get the project.*"

"*O.K., what's the project?*"

"*Myst III.*"

Did he say Myst III? A new sequel in the Myst series that has sold almost 10 million copies worldwide? After picking the phone up off of the floor and closing my gaping mouth, I could only say, "*Wow!*"

"*Yeah, I thought you'd be excited. The proposal we require needs to include some story concepts, an analysis of Myst and Riven, a technology discussion, and, if at all possible, a technology demonstration. Oh, and we need this in five weeks.*"

Needless to say, we hit the ground running.

Presto Studios has been in the computer game business for more than 10 years. We began as a group of friends working out of a home in San Diego on an interactive CD-ROM game called The Journeyman Project. Since that time, we've shipped six other products, grown to as many as 45 employees, and have enjoyed limited success with our games. Myst III: Exile, however, had the potential to take Presto to a whole new level.

Production of Exile began with a very small team: a writer, a creative director, three conceptual designers, and me as producer. The first subject we tackled was an analysis of Myst and Riven. What worked? What didn't? Why were those particular games so phenomenally successful? And what did we want to do differently? Our discussions eventually led to the formation of a few overriding goals for Exile.

First, we would strive for great visual variety in the game. We much preferred the varied worlds of Myst over the more homogeneous chain of islands found in Riven. Second, we would need to provide a way in which players could gauge their progress throughout the game. Players who had failed to complete Myst or Riven did so because they were unsure of how much remained of the game and what their goals were. We didn't want that to happen with Exile. Finally, we wanted an extremely satisfying ending to the game, one which drew upon all of the knowledge that players had acquired throughout their journey. With these goals in mind, we set out on a two-and-a-half-year journey to create a worthy sequel to Myst and Riven.

GAME DATA

PUBLISHER: Ubi Soft

STAFF: 22 full-time employees, 1 full-time contractor, 2 part-time contractors

BUDGET: Multi-million-dollar budget

LENGTH OF DEVELOPMENT: Two and a half years

RELEASE DATE: May 7, 2001

PLATFORMS: Macintosh and PC (hybrid)

DEVELOPMENT HARDWARE: Mostly Dells, averaging dual 700MHz Pentium IIIs with 1GB RAM and 30GB hard drives

DEVELOPMENT SOFTWARE: Discreet 3DS Max, Discreet Combustion, Apple Final Cut Pro, Adobe Photoshop, Metrowerks Codewarrior, Microsoft Visual C++, Microsoft Word, Microsoft Excel, Microsoft Project, Digidesign Pro Tools.

NOTABLE TECHNOLOGIES: RAD Game Tools' Bink and Miles Sound System, Apple's QuickTime.

PROJECT SIZE: A feature-length animated film like *Toy Story* uses 120,000 frames of animation; Exile used more than 150,000

What Went Right

1. Identifying the customer. Who played MYST and RIVEN? What did they like and dislike? What type of computer do they own? We felt that answering these questions would be instrumental in shaping what a MYST sequel should be. So we obtained data from registered owners of the two products, read all of the reviews and articles we could get our hands on, and became active readers of the MYST-related Internet fan sites and web boards. All of the information we gathered was used by the preproduction team to evaluate every part of our game — the visuals, story, puzzles, music, and technology.

For example, the hottest debate in preproduction was whether or not EXILE should use prerendered or real-time 3D graphics. By using prerendered graphics, we felt that we could meet or exceed the visual quality that RIVEN had achieved, but our puzzles would be more limited than if we used real-time 3D, because we would need to precalculate all the possible states for each puzzle from every viewable location. Conversely, real-time 3D would allow us to make the worlds more active and constantly changing, but to achieve the graphic realism of the worlds, the customer would be required to have a very fast CPU and a high-end 3D card.

In the end, this debate was resolved by identifying the MYST consumer. Our research showed us that many MYST players only played a few games each year. They are not out buying new computer systems every few years, so they typically have a slightly older computer, one that would have a hard time keeping up with an advanced real-time 3D game. So, we decided to use

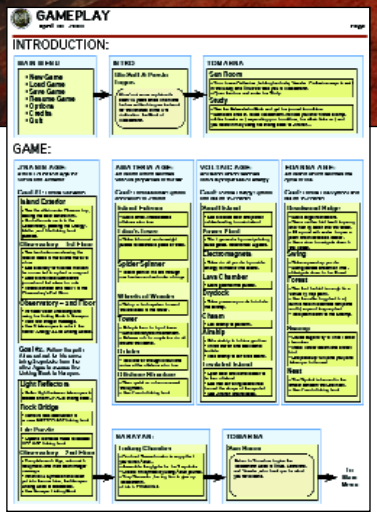
TOP. Spiney bridge in Voltaic. LEFT. Atrus's costume sketch.

prerendered visuals for EXILE, in order to give us the largest possible customer base.

2. Preproduction and planning. Having created adventure games for eight years, you'd think that we would have been able to skip a lot of preproduction and just get to the production of the game very quickly. Actually, the opposite was true. For EXILE, we wanted more preproduction and planning time than we'd had for any of our other products. We had been burned too many times in production, and didn't want that to happen again. There is nothing more disheartening for an artist than to see his or her work go down the drain because of last-minute changes or redesigns. With this foresight, we convinced our publisher that we required a full nine months to design EXILE on paper, before we would create a single graphic. Though I'm sure it made our publisher quite nervous, we knew that this amount of preproduction time would help ensure a smooth production phase.

Preproduction of EXILE began with two teams, one working on story and the other on visuals. We didn't want either team constrained by the other, so we kept them separate for the first month or so. Then we met together and began bouncing ideas around. It was amazing to see the two teams inspire each other — concept sketches led our writer down new plot lines, while story ideas and characters caused our artists to break out their sketchbooks during the meeting. Gradually the two camps met more frequently until the story and visuals became inseparable, ensuring continuity between the final game's world, plot, and characters.

Once the overall story and visual ideas began coming together, we focused on what we call the gameplay structure. This refers to the puzzles or challenges and their accompanying solutions and rewards that the player experiences during the adventure. This gameplay structure needed to allow for the story to be revealed over the course of the entire game, the level of difficulty to increase during the course of the game, and nonlinear events to be employed. We created a flowchart of the game, listing all of the challenges along the way, what they reveal when solved, and any interdependencies between puzzles or areas of exploration. This flowchart was used as a tool to see the game at a glance and make sure that it met our goals of gradual story revelation, increasing difficulty level, and a nonlinear experience.

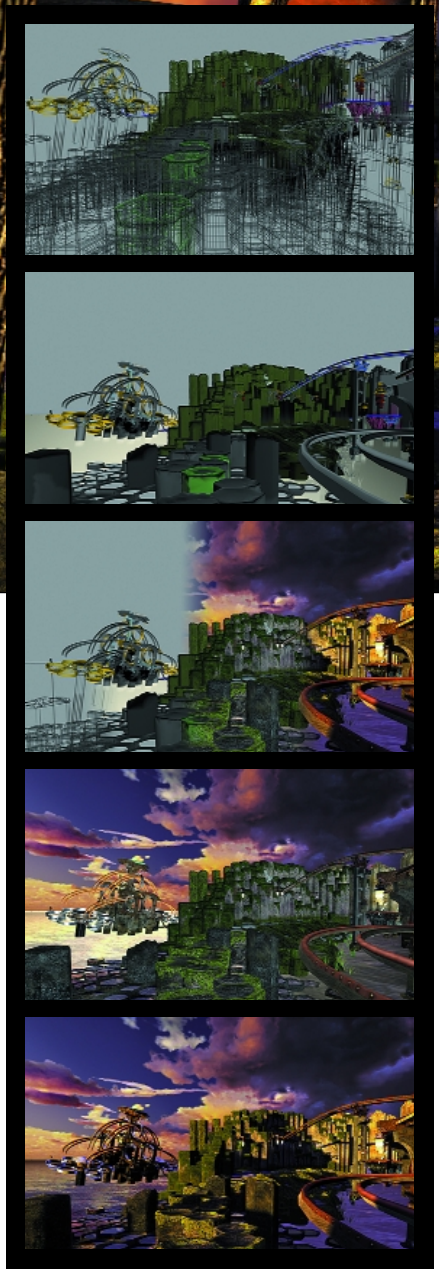


ABOVE. Gameplay flowchart.

When the gameplay flowchart, visual concept sketches, and story were complete, we had our blueprint for the game. This 160-page document was required reading for the entire team. But now it was time to put our money where our mouth was and develop the graphics for the game.

3. Using 3DS Max for art. To be honest, we were at a bit of a crossroads when it came to what 3D package to use for the art of EXILE. We had been using Electric Image on Macintoshes for many years, but had also recently been using Discreet's 3DS Max on PCs for our real-time 3D work. Could 3DS Max create the type of prerendered photorealistic scenes we required for EXILE? And what else could it offer? Our lead animator, Mike Brown, was convinced that with the right finesse, 3DS Max could rival any high-end rendering package, so he set out to do a few tests. In about a week, he re-created one of the small islands in RIVEN and also built a prototype of one of our concept worlds. The results were very encouraging.

We also explored the work flow of an artist using 3DS Max and discovered that the benefits of it being an integrated package were tremendous. The ability to model, texture, light, and animate in the same package solved a huge production nightmare for us. In the past, if an animator found something wrong while he was lighting a scene, he'd have to tell the modeler,



ABOVE. Amateria world from wireframe to final image.

who would fix the problem in a different package and send it to the animator, who would need to update his file with the fix. Talk about a recipe for disaster when you're dealing with tens of thousands of objects.

Our evaluation of 3DS Max proved that it was also the right choice for many practical reasons. First, we knew that we would need to hire many more artists to create EXILE. We found that there were many knowledgeable, talented 3DS Max artists available all over the world. Second, 3DS Max's open architecture and resulting litany of third-party plug-ins meant we could pick and choose additional features for the program at a very low cost. Why spend huge R&D costs to develop realistic ocean water when you can buy the plug-in for a few hundred dollars and have the same water that was used in *Titanic*? We took great advantage of this not only for water, fire, and hair, but also for production tools that helped us model, texture, light, and render much more quickly. 3DS Max proved to be a great choice — loved by the artists and indiscernible from more expensive packages by our customers.

4. Technology. Having decided to use prerendered graphics for EXILE, we were faced with the challenge of making these traditionally static images as immersive as possible. In one of our previous games, we had licensed a technology that displayed still images as 360-degree panoramic views (similar to QuickTime VR) but were unhappy with its performance and image quality. To overcome these deficiencies, we decided to write our own 360-degree technology. We developed a technique that took advantage of the speed and quality of real-time 3D cards. Quickly prototyping our idea with existing imagery, we realized that it worked flawlessly, allowing for very high frame rates without any degradation in image quality. We felt this was exactly the right technology to immerse a player in the worlds of EXILE.

With the basic technology complete, we pursued how to integrate animation, video, and water movement into the panoramic views. For animations and video, we wanted to avoid the QuickTime bounding-area rectangle and harsh compression artifacts that were typical of MYST and RIVEN. So we investigated several other compression algorithms and playback engines, finally deciding on RAD Game Tools' Bink technology. Bink provides fantastic compression and high image quality, playback engines for both Macintosh and PC, a fairly low processor speed requirement, and a host of special features. For instance, using the alpha channel support inherent in Bink, we were able to display animations and video in such a manner that only the changing pixels were drawn. This meant that the bounding box rectangle of the movie was gone and the compressed pixels were much less noticeable in the changing image.

The last piece of the technology puzzle was the procedural effects, such as the moving ocean water. The waves needed to move realistically, look correct from altitudes ranging from five to 400 feet, and fade off into perspective toward the horizon. To accomplish this, we first generated alpha channels of the ocean water for each panoramic image in the game. Next, we wrote an image manipulation algorithm (similar to a Photoshop filter) that properly bent and twisted the water pixels. The altered pixels were then applied to the ocean water texture (using the alpha channel) 15 or more times per second to give the water the illusion of movement. Variations of this technique were also used to simulate bubbling and ebbing lava as well as one puzzle's visible sound waves.

5. Web support and fan community. One of our early concerns for EXILE was that MYST fans seemed to believe that only Cyan (the creators of MYST and RIVEN) could create a great MYST game. We had to find a way to convince the MYST/RIVEN/D'ni fan community that even though Presto was creating the game, EXILE would meet or exceed their expectations. Our solution was the Internet. First, we identified two leading fan sites on the Internet, and in May 2000 (one year prior to shipping) invited their webmasters to come to Presto for a special sneak preview of EXILE. After viewing our teaser trailer, getting a hands-on demo, and browsing our concept sketches, both webmasters were sufficiently impressed and vowed to share their positive experience on their sites. Furthermore, we established such a good relationship with both webmasters that we coordinated with them over the next year and worked with one of them to create the official Myst3.com site.

Our second solution using the Internet was to release a teaser trailer and early screenshots. The trailer was intended to evoke the spirit of MYST and RIVEN while providing a sneak peek at what EXILE had to offer visually. Supporting the trailer were numerous screenshots that showed the detail and beauty of the first world of EXILE. The trailer was downloaded more than half a million times in the first month, and the screenshots were posted on numerous web sites and scrutinized by the die-hard fans. In short, our underground public relations campaign was working well, and the fans were rejuvenated, eager to follow EXILE's progress over the next year, culminating with the game's launch.

ABOVE. An example of EXILE's ocean water texture.

What Went Wrong

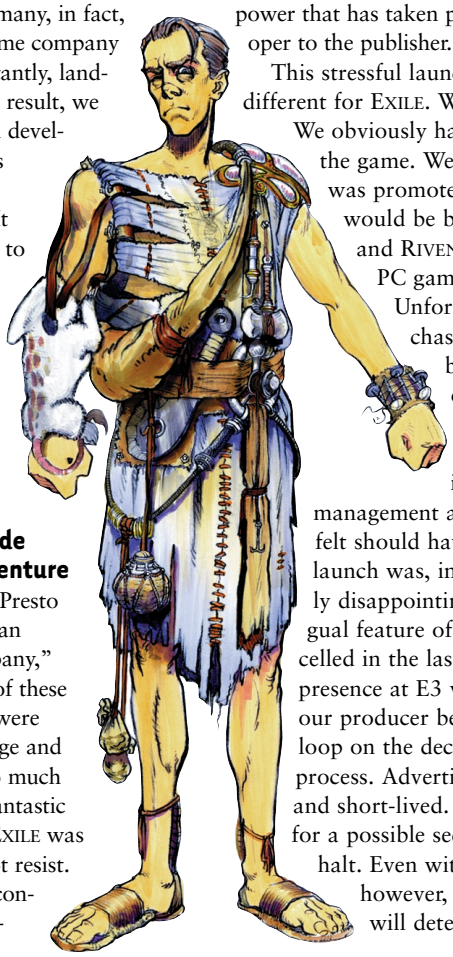
1. Video quality. The live-action video shoot, and everything leading up to it, was probably the game's most important development milestone. It was an extremely hectic time. The script-writing had to be completed, the costumes designed and sewn, the props built, the CG backgrounds rendered, the actors hired and rehearsed, and the studio and personnel booked. All of these individual elements did come together, but one critical oversight prevented our resulting video from being crisp and perfect: we didn't use HDTV cameras.

Months after the video shoot was complete, we began compositing the video footage — removing the blue backgrounds from the live-action video and replacing them with our computer-generated worlds. After running a de-interlace filter on the footage (to remove the inherent NTSC scan lines), it quickly became evident that the lack of source video resolution (the number of pixels) was resulting in a blurry image. Even with image-sharpening tools and the latest filters in our compositing package, we were unable to achieve the crisp video that we had hoped for. This issue would have been avoided entirely if we had used the vastly superior image resolution of HDTV. We certainly now know the adage: Garbage in, garbage out.

2. Underestimating the budget. Having created many adventure games in the past, we had a reasonable estimation of how much it would cost to develop EXILE. However, what we didn't account for was how much extra effort it would take to reach the image quality level that was required. We were no longer working under our own quality levels, but rather ones that MYST and RIVEN had

established. This mistake meant that we underestimated the cost of the game, though we had signed a contract to produce the game for that amount. Underestimating the budget translated directly into not being able to hire enough team members, which translated into an insane schedule. It really was as simple as that. We all had to, and did, work our tails off, but this only resulted in a general feeling of "Whew! Glad that's over" rather than "That was a blast! Let's do another one!"

3. Sacrificing future projects. Undermanned and underfunded, everyone hunkered down and focused solely on EXILE. Not just the team, but the entire company. Personally, I was producing the game, prototyping puzzles, and programming many of our final worlds. Our president was executive producer for the title, technical director for the video shoot, and responsible for compositing all of the live-action footage. Clearly, we were wearing too many hats. So many, in fact, that we lost sight of some company goals and, most importantly, landing more projects. As a result, we only had one project in development after EXILE was complete, and we were forced to lay off staff. It was very disappointing to me personally to see people who had slaved over the project be "rewarded" with being laid off. The devotion that we all had to EXILE clearly had its cost.



4. So we've made another adventure game. Prior to EXILE, Presto was already known as an "adventure game company," having worked on six of these games in the past. We were eager to shake this image and prove that we could do much more. Obviously, the fantastic opportunity to create EXILE was something we could not resist. However, we are very concerned that the completion of EXILE will further brand us as an "adventure game only" company. Will EXILE's completion bring us many new opportunities? Or will we have to try even harder to avoid this moniker? Only time will tell.

5. The launch: It's out of our hands now. The launch of any title is usually one of the most stressful times for the developer-publisher relationship. The developer is exhausted, having just crunched through several years of production, yet hopeful and full of expectations for the success of "their" product. However, the publisher then takes the reins, promoting the product, manufacturing the boxes, selling as many copies as possible, and handling customer support. But where does this leave the developer? In truth, the developer is still financially tied into the product but has absolutely no more control over it. That is why the launch is always so stressful, for it represents the transition of power that has taken place from the developer to the publisher.

This stressful launch period was no different for EXILE. We were exhausted. We obviously had high hopes for the game. We felt that if EXILE was promoted as being "big," it would be big. After all, MYST and RIVEN are the best-selling PC games of all time. Unfortunately, the purchase of our publisher by another publisher changed the decision-making process, the people in power, and the management approach. What we felt should have been a huge launch was, in our opinion, slightly disappointing. A great multilingual feature of the game was cancelled in the last few weeks. Our presence at E3 was minimal, due to our producer being taken out of the loop on the decision-making process. Advertisements were scarce and short-lived. And negotiations for a possible sequel ground to a halt. Even with these occurrences, however, we believe that time will determine EXILE's success, not the minor setbacks from the launch period of the game. Going forward, we expect that the launch will only be a bump in the road, not a sign of things to come.

ABOVE. Background image, source video, alpha matte, and final composite. LEFT. Villian's costume concept sketch.

Closing Thoughts

As you've seen, some elements of our production worked out perfectly, and probably saved us months of time and tens of thousands of dollars. Others didn't go quite as well as we'd hoped, and parts of the product suffered because of them. Throughout the process, we tried to appreciate the positives and learn from the negatives. I've heard it said that good judgment comes from experience, and experience — well, that comes from poor judgment. Here's hoping that we showed a lot of good judgment and that the "experience" wasn't too painful.

Who Are You Hiring?

I want to talk for a moment about whom you are hiring. This question applies to our industry in general but more specifically to whom you are hiring for your audio, whether it's a contractor or for a full-time employee position.

There are three primary components to every game: the game logic, art, and audio. With audio being such a major component of the gameplay experience, shouldn't music composition, dialogue, and sound effect design be given the same diligence as the rest of the game development process?

Normally when you are looking to hire a programmer, level designer, or artist, you prefer to find someone with game industry experience, right? Hiring someone with a background in programming games or creating game art is simply good business; it speeds up their integration into the team, cuts down on training time, and increases productivity. Fewer mistakes are made, because they understand the tools and processes involved. Now that's not to say that you would never want to bring on someone new to the industry, because a fresh perspective can be a good thing, but they would certainly start out in a junior or assistant position where they can learn how things work in the game development world. This same philosophy should apply to your audio personnel. Why would you even consider hiring someone with no experience in creating game audio to be your lead or sole composer or sound designer?

Illustration by Dave Whamond

To the detriment of the industry, too often sound designers and/or composers from other industries such as film, TV, or radio are hired as contractors for a project or as full-time employees in a budding audio department. As for the latter situation, much time will be spent by someone on the team teaching the new recruit the ropes. Just because someone can play an instrument doesn't make him or her a composer or a sound designer. Just because someone is a composer for film or TV doesn't mean he or she understands the process of composing for games, and if they are not an avid game player as well, the learning curve will be even steeper. The same applies to contractors; however, contractors who are not dedicated to this industry may never learn its intricacies nor understand its audience. Relying on contractors outside the game industry as experts to design the correct audio for your genre and audience means that it's not very likely that they are going to be current on trends and technologies specific to our industry. You might think it would be great to have a John Williams or

continued on page 55

continued from page 56

Hans Zimmer compose for you, but what does Mr. Williams know about an online RTS or its audience?

So what's a developer or producer to do, and what should you look for when hiring audio personnel or contractors? If you're thinking about starting up an audio department, look first to people already working in the game industry. You might get lucky and scoop up someone with experience and (if they're a contractor) some gear. If you have to hire someone outside the game industry or with little to no experience, you might want to consider bringing him or her on board as an intern or as a junior employee and possibly work with your contractor for a couple of projects. Then they can learn what to do and how to do it before you turn them loose on their own. This might cost you a little more up front, but it should also reduce your training task and might speed up audio development time. In the long run, it will provide you with an employee who makes fewer mistakes and understands designing audio for games better.

When hiring a contractor, besides the obvious list of credits you should look at what else they do. Are the majority of their credits in other industries? How involved are they in the game community and what are their affiliations? Do they spend the time and money to go to the Game Developers Conference or E3 to learn what's going on and keep up with the changes in technology? What technologies are they familiar with? Are they game players themselves? Being an avid game player is extremely important for anyone you hire but all the more so for your composer/sound designer — understanding how the audio interacts with and affects the game and the player is crucial to getting it right. The last thing you should consider is cost. If cost is your first concern, you will most likely not end up with a game audio professional. People working on the cheap or for free are generally doing so for a reason.

Why did you get into the game industry? To become rich and fabulously wealthy or because you love games? Just like you, there are plenty of composers and sound designers out there that have an interest in the game industry beyond making a buck — like making better games.

ROB ROSS | *Rob has 20 years' experience in the audio industry as a professional musician and mixing engineer. He founded Sound Endeavours Studios in 1998 and has worked on such games as VAMPIRE: THE MASQUERADE — REDEMPTION and STAR KNIGHTS. He moderates the Game Audio SIG for the IGDA, is a voting member of the Academy of Interactive Arts and Sciences, and belongs to the Microsoft DirectX 8 DirectMusic beta team, the National Academy of Recording Arts and Sciences, and the IA-SIG. He can be reached at rob@soundendeavours.com.*