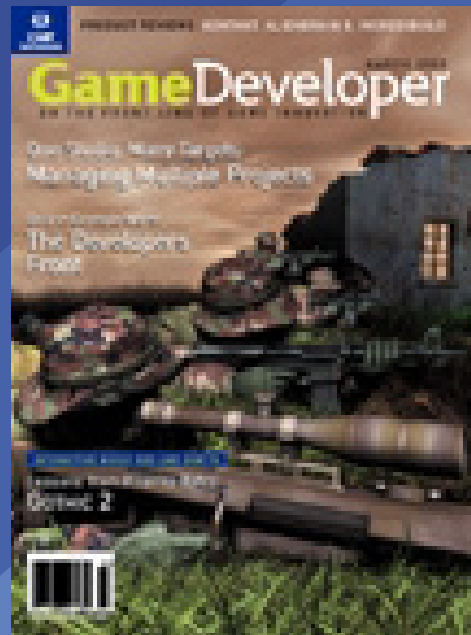




GAME DEVELOPER MAGAZINE

MARCH 2003





GAME PLAN

LETTER FROM THE EDITOR

Our Kingdom for a Producer

Producers in this industry play a unique and hard-to-define role. Good ones are gold; smart developers understand this fact, and also how to work with them on all fronts to make sure that a vision for a game is something akin to what winds up on the shelf and, hopefully, in consumers' shopping bags. Bad ones are often to blame for a product that combines sound technology, dazzling art, and interesting gameplay into a game that is ultimately, for some elusive reason, just not fun to play. The unmentionable ones confound the industry with the sheer longevity of their careers.

The qualifications for being a game producer are far more vague than those for the traditional development disciplines, which are generally quantifiable and demonstrable. There is also the perception that production positions are often either from-within promotions (out of test, for example), or filled through connections and thus go unadvertised — the dreaded friend-of-a-friend hire, a favor called in by someone who doesn't know what else to do with their life, most likely because they haven't found anything they're good at. The primary qualification cited for most of the jobs that are listed on online game-job boards? Usually something akin to "Must have passion for games" (all caps optional).

Given how crucial a role producers at all levels play in the ultimate success or failure of a product, why isn't there a better way to define and quantify the skills candidates need for one of these jobs? As much of it revolves around marshaling relatively pedestrian administrative tasks as chasing — and advocating for — a vision for the elusive "fun factor." Microsoft Project comes with help documentation, but how can we hold others up to any standard of enforcing a "fun factor" that we don't fully understand ourselves?

Anyone in a position to hire a producer or choose one to work with on a project must come up with better qualifications and standards by which to make such an important decision. Too many bad games

are on the shelves, and the gulf between good productions and bad productions is ever widening. If consumers are demanding higher production values in games, it follows that the bar for producer value has been raised as well.

In the Mel Brooks film and hit Broadway musical *The Producers*, the farcical story line revolves around a crooked scheme to produce a sure-fire stage flop in order to bilk unsuspecting little-old-lady investors out of their money. Let me not get too far in thinking about potential parallels in the game industry; it's supposed to be comedy after all. But it's a serviceable parable about the fall that comes when abject greed is combined with insufficient ability to predict market demand. In *The Producers*, the sure-thing bomb they create, "Springtime for Hitler," becomes a surprise smash hit and exposes the scam.

Like Broadway, games should aspire to be an industry where "if you can make it here, you can make it anywhere," not "my friend gave me this job because I really like games." We need better developed concepts of applying a vision for a game experience to analysis and prediction of eventual market performance. If there were a more successful formula for this than the current status quo, we might even be able to fund more risk in the industry rather than watch those potential funds sucked up by poor products that never should have made it to market.

In the future, all game developers will themselves have to be, to one degree or another, producers. Developers' contributions to a combined effort cannot exist in a vacuum. The most successful scenarios will involve developers who can understand and anticipate the far-reaching ramifications of their creative decisions and implementations, as they relate to those of their co-collaborators and the ultimate successful execution of the vision for their game.

Jennifer Olsen
Editor-In-Chief

GameDeveloper

www.gdmag.com

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

Publisher

Jennifer Pahlka jpahlka@cmp.com

EDITORIAL

Editor-In-Chief

Jennifer Olsen jolsen@cmp.com

Managing Editor

Everard Strong estrong@cmp.com

Production Editor

Olga Zundel ozundel@cmp.com

Product Review Editor

Daniel Huebner dan@gamasutra.com

Art Director

Audrey Welch awelch@cmp.com

Editor-At-Large

Chris Hecker checker@d6.com

Contributing Editors

Jonathan Blow jon@number-none.com

Hayden Duvall hayden@confounding-factor.com

Noah Falstein noah@theinspiracy.com

Advisory Board

Hal Barwood LucasArts

Ellen Guon Beeman Monolith

Andy Gavin Naughty Dog

Joby Otero Luxoflux

Dave Pottinger Ensemble Studios

George Sanger Big Fat Inc.

Harvey Smith Ion Storm

Paul Steed Independent

ADVERTISING SALES

Director of Sales/Associate Publisher

Michele Sweeney e: msweeney@cmp.com t: 415.947.6217

Senior Account Manager, Eastern Region & Europe

Afton Thatcher e: athatcher@cmp.com t: 828.350.9392

Account Manager, Northern California & Southeast

Susan Kirby e: skirby@cmp.com t: 415.947.6226

Account Manager, Recruitment

Raelene Maiben e: rmaiben@cmp.com t: 415.947.6225

Account Manager, Western Region & Asia

Craig Perreault e: cperreault@cmp.com t: 415.947.6223

Account Representative

Aaron Murawski e: amurawski@cmp.com t: 415.947.6227

ADVERTISING PRODUCTION

Vice President, Manufacturing

Bill Amstutz

Advertising Production Coordinator

Kevin Chanel

Reprints

Cindy Zauss t: 909.698.1780

GAMA NETWORK MARKETING

Director of Marketing

Greg Kerwin

Senior MarCom Manager

Jennifer McLean

Marketing Coordinator

Scott Lyon

CIRCULATION



Game Developer is BPA approved

Group Circulation Director

Catherine Flynn

Circulation Manager

Ron Escobar

Circulation Assistant

Ian Hay

Newsstand Analyst

Pam Santoro

SUBSCRIPTION SERVICES

For information, order questions, and address changes

t: 800.250.2429 or 847.647.5928 f: 847.647.5972

e: gamedeveloper@balldata.com

INTERNATIONAL LICENSING INFORMATION

Mario Salinas

t: 650.513.4234 f: 650.513.4482 e: msalinas@cmp.com

CMP MEDIA MANAGEMENT

President & CEO

Gary Marshall

Executive Vice President & CFO

John Day

Chief Operating Officer

Steve Weitzner

Chief Information Officer

Mike Mikos

President, Technology Solutions Group

Robert Faletta

President, Healthcare Group

Vicki Masseria

President, Electronics Group

Jeff Patterson

President, Specialized Technologies Group

Regina Starr Ridley

Senior Vice President, Global Sales & Marketing

Bill Howard

Senior Vice President, HR & Communications

Leah Landro

Vice President & General Counsel

Sandra Grayson

Vice President, Creative Technologies

Philip Chapnick



GamaNetwork

SAYS YOU

A FORUM FOR YOUR POINT OF VIEW. GIVE US YOUR FEEDBACK...



Not So Fast

In “The Live Orchestra Recording: A Producer’s Awakening” (Sound Principles, December 2002), Andy Brick argues that using a live orchestra is financially equivalent to re-creating the orchestra with MIDI and sampled instruments. He makes much of the fact that it only takes 10 hours to record 50 minutes of music with a European orchestra, whereas it takes a sample-based composer 35 to 40 hours to create a typical four-minute piece. In his example, he claims that recording a live orchestra saves 400 hours of work by eliminating the time spent tweaking sampled instruments to make them sound real.

The problem is that the live-orchestra composer also has a lot of work to do — work that Mr. Brick seems to have overlooked:

1. Preparation of the printed score for conductor and each instrument in his orchestra, for 50 minutes of music. Adding articulations, dynamics, and phrasing is a considerable amount of work and even physically printing out that much paper takes a while! On a large-budget film, copyists are hired to help with this task alone.
2. The logistics of setting up a large recording session in a foreign country. That kind of organization takes time.
3. Preparing a MIDI version of the music for the client (the game developer) to review. Although this version needn’t be prepared to the level of detail that a sample-based composer requires, having done this, the live-orchestra composer is maybe 50 percent of the way toward having a sample-based score.

Given the factors I’ve just mentioned, it seems unlikely that the live-orchestra composer will save anything like 400 hours of work (in fact, I wouldn’t be surprised if he ends up taking more time than the sample-based composer).

Don’t get me wrong — I’m definitely in favor of recording live orchestras, as it’s probably the most fun a composer can have, but putting too low a price on it is not helping the cause in the long run.

Simon Amarasingham
via e-mail

Not Without the PNG

One big “con” Spencer Lindsay missed in his January 2003 review of Photoshop 7 was its still-terrible PNG support. At a minimum, it needs to fix its gamma handling, include better transparency support, and it needs to do a far better job at compression. While PNG’s use is limited on the web, being supported fully only by Mozilla-based browsers (Netscape 6 and 7, AOL for Macs, and Konqueror among others) and the Mac version of Internet Explorer, many of the upcoming mobile browsers have good support for it, and upcoming web formats like SVG mandate it.

Unlike screenwriters, most writers in other fields are not attached to some inflexible notion of story structure.

On the game front, it offers a host of useful features: 1- to 8-bit images, 1-bit image transparency (like .GIF), full 8-bit alpha channel (this alone makes fringing on web graphics a thing of the past for browsers that support it) and gamma support. Gamma support is useful for graphics that may be displayed using a variety of methods, for instance, the traditional CRT displays most of us still use, or LCD displays found in handheld phones and PDAs.

While versions 6 and 7 brought needed web optimization functions to Photoshop, let’s hope they keep the needs of game developers in mind for future releases. In the meantime, users may want to look to a back-up program, like Jasc’s PaintShop Pro for PNG handling, or a Photoshop plug-in like Brendan Bolles’ excellent SuperPNG saver.

Zoltan Hunt
via e-mail

Who Needs Hollywood?

In “What Screenwriters Don’t Know About Games” (January 2003), David Freeman makes a strong case for the utility of good writing in games — and the cultural clashes that sometimes occur between writers and game developers.

He also makes a purely gratuitous swipe at comic book writers, maintaining that they’re incapable of creating work of the emotional depth of the *Lord of the Rings* or *Buffy the Vampire Slayer*. Tolkien is a hard one to match, but I’d put Alan Moore up to the task. Gaimain, Ellis, Ennis, Kyle Baker . . . but I don’t want to continue this laundry list. I have to assume Freeman hasn’t been reading comics in recent years.

In general, I can’t imagine why one would want to hire an overpaid, arrogant screenwriter for whom working in games amounts to slumming, when there are fine writers who work in many other fields for far less, and would find game

industry rates of pay munificent by comparison to their other work. As an illustration, Asimov’s magazine pays seven cents a word for fiction. And unlike screenwriters, most writers in other fields are not attached to some inflexible notion of story structure — there are, as Kipling says, four and twenty ways of writing tribal laws, and every single one of them is right.

Personally, I’d look to genre fiction, comics, and radio drama for writing talent. Those writers charge less, are a lot more humble, and eager for the work — and as talented as the drones from Hollywood.

Greg Costikyan
via e-mail

CORRECTION

In the January 2003 “Says You,” we ran a letter mistakenly that stated that DEER HUNTER 3 was made by Southlogic Studios. In fact, Sunstorm Interactive deserves credit for that title. We regret the error.



Let us know what you think: send us an e-mail at editors@gdmag.com, or write to *Game Developer*, 600 Harrison St., San Francisco, CA 94107



INDUSTRY WATCH

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

Acclaim starts fiscal 2003 with a loss.

Acclaim's Q1 performance slid from a \$17.4 million profit in Q1 2002 to a \$13.9 million loss (15 cents per share) for its Q1 2003, which ended December 1, 2002.

According to Acclaim, the loss was a result of "weak performance of [Acclaim's] titles in the North American retail market." These titles include *BURNOUT 2: POINT OF IMPACT*, *BMX XXX*, *LEGENDS OF WRESTLING 2*, *DAVE MIRRA FREESTYLE BMX 3*, and *MARY-KATE & ASHLEY SWEET 16* for various platforms.

And the Oscar goes to ... Maya?

Alias/Wavefront, the makers of Maya, will be awarded an Oscar for the contributions of its 3D graphics program to the motion picture industry. The award ceremony will be held March 1, 2003.

Goldman Sachs downgrades growth.

Though Goldman Sachs originally projected 2003 growth of between 10 and 12 percent for the interactive entertainment industry, those numbers have since been toned down to five to 10 percent growth. Citing low profit warnings from such publishers as Activision and



Low sales of *BURNOUT 2: POINT OF IMPACT* impacted Acclaim's Q1 financial picture.

THQ, the company suggested that the retooled numbers might even be an aggressive outlook.

On a positive note, Goldman Sachs believes that if sales of *THE SIMS ONLINE* surpass half a million subscribers by the end of March (Electronic Arts' fiscal year-end), EA could see a 20- to 25-cent-per-share boost. However, if total subscribers come in under 350,000, the company could take a hit. Though Electronic Arts has issued a statement saying that sales of *THE SIMS ONLINE* were better than expected, Goldman

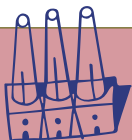
Sachs stated that initial consumer feedback on the game was "lukewarm."

THQ teams up with Konami. THQ Wireless, a subsidiary of THQ, signed a co-publishing partnership with Konami's subsidiary, Konami Mobile and Online, to release several of THQ's games through the NTT DoCoMo i-mode network.

BAM! Entertainment lowers outlook. BAM! Entertainment lowered its revenue outlook for the quarter ending December 31, 2002, from previous estimates of \$25 to \$30 million range down to \$22 to \$23 million. In addition, the company announced it will sell the assets and operations of its London-area product development studio to Scotland-based VIS Entertainment and cease all internal development efforts.

Sony's Christmas dreams come true. Reporting over 6.5 million Playstation 2 units sold in Japan, North America, and Europe between Thanksgiving and Christmas, Sony announced that 2002 was its biggest year since 1995, when the original Playstation was released. In comparison, Sony sold roughly 5 million units during the same period in 2001. Sony also announced it has passed the 50 million mark in PS2 worldwide sales. 🎮

Send news items and product releases to news@gdmag.com.



THE TOOLBOX

DEVELOPMENT SOFTWARE, HARDWARE, AND OTHER STUFF

Metrowerks teams up with Nintendo.

Metrowerks will be distributing TDEV, Nintendo's new Gamecube development hardware, bundled with its CodeWarrior for Nintendo Gamecube, Version 2.0. According to Metrowerks, the new hardware will provide developers a less expensive network-capable development kit. www.metrowerks.com

Kaydara launches free 3D file viewer.

Kaydara recently released FBX for QuickTime, a real-time 3D file viewer that supports content from the major 3D programs. The free program allows producers and artists to look at 3D

content using QuickTime, without needing a special file-viewing application. You can download the viewer directly from Apple's QuickTime page. www.apple.com/quicktime

Character Studio 4 now available.

Discreet has released the newest version of its character animation software, Character Studio 4. According to Discreet, version 4 includes improvements such as dynamics-based mixing, constraint-based mixdowns, and quaternion function curves. The software has a suggested retail price of \$995. www.discreet.com



UPCOMING EVENTS CALENDAR

GAME DEVELOPERS WORLD

BELLA CENTER
Copenhagen, Denmark
May 8-10, 2003
Cost: variable
www.gd-world.com

E3

LOS ANGELES CONVENTION CENTER
Los Angeles, Calif.
May 13-16, 2003
Cost: Free-\$550
www.e3expo.com

PROFILES

TALKING TO PEOPLE WHO MAKE A DIFFERENCE | *everard strong*

Richard Garriott: Look East, Young Man

South Korea has become a huge market for online gaming, with top players of games like STAR-CRAFT and LINEAGE being elevated to the level of sports stars, competing in grand tournaments for lots of cash. In comparison EVERQUEST, one of the most popular North American MMOGs, is mostly played in the seclusion and anonymity of subscribers' living rooms and bedrooms. As head of NCSoft's Austin, Tex., division, famed ULTIMA creator Richard Garriott's job is to localize and support NCSoft's Asian-created products in the western markets — including the multi-million-selling LINEAGE series — and make them successful in the United States, and also to export North American games to European and Asian markets. It has not been smooth sailing. Richard explains why Koreans love online gaming, and why American developers need to understand this unique market before exporting their wares over there.

Game Developer: You recently came back from a trip to South Korea. As NCSoft's American counterpart, what was your main purpose for going over there?

Richard Garriott: One of the things we were interested in looking at as American developers going over there was to see what kind of a market for American-developed games that area represents. And likewise, the games developed in Asia, how marketable will they be here in the United States? For us now at NCSoft, as a global company, one of our big decisions is going to be, When do we globalize a product? When do we take a single product and ensure that it has worldwide success, and in what cases do we take a game and figure out that it is really best suited for an individual territory?

GD: The MMOG market is huge in Korea and Southeast Asia, with the LINEAGE series currently boasting over 4 million subscribers. MMOGs with American subscribers (like EVERQUEST) have maybe a tenth of that number. With those differences, how does NCSoft as a company view American and other western audiences?

RG: The Korean development team's number-one job activity — based upon pure economics — is to ensure that the 4 million players they already have in Southeast Asia are entertained to the maximum level, ensuring the continuation and growth of that subscriber base.

Even if LINEAGE were the number-one game in the United States (which EVERQUEST, with close to 500,000 subscribers, currently is), the number is not very consequential when compared to the existing Asian market.



Richard Garriott left his ULTIMA kingdom to champion NCSoft's online games, including LINEAGE.

GD: What are some challenges you've encountered bringing LINEAGE from an Asian market to a North American one?

RG: One of the biggest problems with taking LINEAGE and trying to maximize its success in the United States is the fact that the game, which has been around for almost five years, is already a mature game with a large customer base.

GD: Are there noticeable differences between the Korean, Japanese, Taiwanese, and Chinese MMOG markets?

RG: Well, I would break them into three principal areas. There's Korea, Taiwan, and Singapore, which are similar in ideas. Japan and China are both separate groups, with a number of special factors at play and different market dynamics.

GD: So why, out of all these countries, has Korea become such a huge market for MMOGs?

RG: After World War II, countries like Korea and Taiwan had a substantial grudge against Japan, and banned the import of a lot of Japanese products. Because of this, console systems like Playstation and Nintendo have not become very popular. Therefore if people [in Korea] are gamers, they are PC gamers.

But in Korea, while salaries are almost half of what they are in the States, computers are about the same price. So game rooms have popped up throughout Southeast Asia, and many people choose to use these computers (for which you pay by the hour) instead of buying their own.

And the city of Seoul, where over 5 million residents live, is almost 100 percent broadband wired. These factors make South Korea and its neighbors one of the most attractive places on the planet for online gaming.

GD: Do you see a difference that exists at the level between how a Korean or an American plays an MMOG?

RG: If you watch players in the Korean game rooms, they are very team-oriented, and they're all very well coordinated as a team, with leadership granted to one individual, and the others playing their roles very efficiently. If you look at American players, American kids grow up to be mavericks. Be it in a game like LINEAGE or BATTLEFIELD 1942, it's every man for himself. It's a "Let's all jump in and rush the other team" mentality, and if you've coordinated who's on offense or defense, you've done pretty well. And it's definitely a factor in how we now think about not only the design, but even the support and the operation of these online games. *ES*



Native Instruments' Kontakt

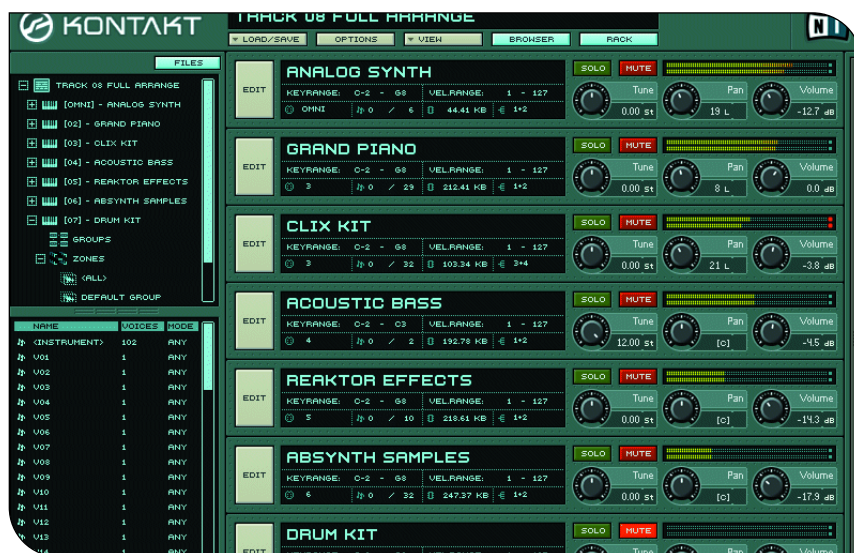
by aaron marks

Anyone considering a versatile software-based sampler should take a close look at Native Instruments' Kontakt.

This full featured, easy-to-use program mirrors the capabilities of expensive hardware samplers while adding its own unique sound creation tools, giving the \$399 (MSRP) price tag greater value.

Kontakt is designed as a stand-alone VST 2.0 or DXi instrument for use with Windows 98, ME, 2000, XP, and MacOS 9.0 or higher. It combines a virtual rack of sound modules which depict your loaded samples, with an interface similar to that of Windows Explorer. Accessing sample CDs or sound files stored on the hard drive is easy, with a simple drag and drop to the rack window instantly loading them for use. Additionally, numerous filter types, modulation capabilities with real-time display, onboard effects, granular time-stretching and re-synthesis options, and a loop editor round out this program's ability to play back standard samples and shape fresh sounds.

The program ships with over 3 gigabytes of playable samples. Drums, percussion, bass, guitar, piano, and synth samples are keymapped so you can get started immediately. Kontakt can load a multitude of formats, including Akai S1000/S3000, Gigasampler, SF2, Battery, Halion, EXS, Reaktor, LM4, and 8- to



Kontakt's intuitive control panels offer users several options in developing their sound.

32-bit AIFF/WAV files. NI plans to add support for Emu very soon.

Kontakt currently boasts a polyphony of up to 256 stereo voices per instrument with the capability to run 16 instruments at once. Output capability ranges from 32 channels when used as a plug-in to 16 as a stand-alone unit. Like normal hardware samplers, the real limitation is the amount of RAM available when loading samples. However, the folks at NI have announced a direct-from-disc extension which, by the time you read this, should be available for download. As samplers

go, Kontakt has all the standard features you'd expect. Keymapping and multi-creation is simple; ranges and layered patches are easily established with a simple click and drag. Other parameters are highly useable; LFOs, envelopes, and modulators can be controlled via MIDI or separate controller. Everything is depicted graphically, giving you a distinct perspective of what's actually happening.

The LFOs come fully equipped with the standard options: sine, triangle, sawtooth, rectangle, random, and a multi setting to combine them for some interesting effects. There are three types of envelopes on board which can add some nice depth to your sounds, all with the ability to stretch over several minutes. AHDSR (Attack, Hold, Decay, Sustain, and

AARON MARKS | Aaron is a game composer, a sound designer, and proprietor of On Your Mark Music Productions (www.onyourmarkmusic.com). He is currently hard at work on projects for 1 Up Studios, Beyer Productions, and Enemy Technology. Aaron is also the author of The Complete Guide to Game Audio, published by CMP Books.



Release) and Flexible are the two standard envelope types, and DBD (Decay time 1, Break, Decay time 2) is an envelope specifically designed to affect pitch.

Step modulation allows for 32 steps of custom modulation effects, Glide (or portamento, in more musical terms) adds a sliding pitch transition between consecutive notes. Envelope Follower translates a sample into a control signal which can then be applied to other samples to give them the same characteristic.

A great selection of effects processors, including distortion, saturation, lo-fi, compression, stereo enhancer, delay, chorus, flanger, and reverb satisfies even the most discerning musician — they all sounded first-rate. One surprise was the low CPU usage; when using multi-track software with numerous plug-ins running, I'm always looking at the meter, and I found Kontakt's effects to be easy on the overhead.

Kontakt's two most distinctive features

KONTAKT



STATS

NATIVE INSTRUMENTS

Los Angeles, Calif.

(866) 556-6487

(323) 467-5260

www.ni-kontakt.com

PRICE

\$399 (MSRP)

SYSTEM REQUIREMENTS

Mac: MacOS 8.6 or higher, G3 300MHz, 128MB RAM.

Windows: Windows 98/2000/XP/ME, Pentium II 300MHz, 128MB RAM.

PROS

1. Cheaper than a hardware sampler with more powerful features.
2. Unique and fresh sound-shaping capability.
3. Works as a stand-alone or with most sequencers.

CONS

1. Small font size.
2. Awkward window functioning.
3. No undo.

are its Tone Machine and Time Machine modules. The Tone Machine is a granular synthesizer, which will analyze a sample and then provide several knobs such as tune, smooth, speed, and format to adjust that sample. Time Machine also uses granular synthesis, and allows for playback of a sample in its original pitch while changing its length. This is a great feature for matching the sample to the tempo of the song, or trying something interesting such as bringing the sample to a complete halt with the pitch still ringing.

The biggest complaint I have is with the small font size. Most sound designers have their screen size set at 1024×768 or higher. At these resolutions, it's very difficult to see the text clearly.

For those of you who haven't made the move to a software-based sampler, Kontakt may just be the one to motivate you to do it. Since I gave up my hardware sequencer for a software-based one many years ago, it's almost a necessity now to see my sounds and be able to manipulate them graphically. This program is perfect for that continuing tradition. For Gigasampler owners or those who have hardware samplers with some life left in them, Kontakt would make another great tool in the shed.

Xoreax Software's Incredibuild

by Justin Lloyd

As a programmer, I've looked on with jealousy at the distributed rendering systems for packages such as Maya and 3DS Max. Discounting large multi-processor mainframes or Unix and Linux platforms using distributed makefiles, programmers were stuck with their single-platform compilation capability, spending several hours waiting for a full rebuild of the tool chain or application executable.

Until recently there has been no viable Windows or Visual C++ solution. But now, Xoreax's Incredibuild provides the answer for the programmer looking for the equivalent of distributed makefiles spread over a compilation "render" farm.

Incredibuild performs a distributed compilation of your C and C++ source files across multiple computers connected via a TCP/IP network — whether a 56Kbps modem or a 802.11b WiFi — running Microsoft Windows NT, 2000, or XP.

Incredibly (no pun intended), it does all of this on machines that don't need to have Microsoft Developer Studio installed. A single machine having Microsoft Developer Studio 6.0 up and running, along with the Incredibuild DevStudio Add-in installed, is all that's needed for a development machine. Agents on remote machines and the Coordinator do not require DevStudio to be installed, forgoing both the expense of an unused software package and the requirements of a capable machine. This also means that the Incredibuild Agent can be placed onto office machines not directly involved with development.

Incredibuild performs its magic by sharing out the compilation of source files to individual machines running the Incredibuild Agent. A source file is considered atomic — no further subdivision of labor in compiling it is possible — so the more source files in a project, the better the performance gains witnessed.

Each Incredibuild Agent, and there can be up to 100 Agents in a build farm (though 40 is the recommended maximum), creates a small cache on the HD, and, utilizing spare CPU cycles — it runs at the "Idle" priority by default — compiles C and C++ source files as though they were being compiled on the machine that's running Microsoft Developer Studio. Should an Agent detect that the host processor usage has spiked from user interaction, it stops processing and informs the Incredibuild Coordinator to find another Agent to perform the work.

In DevStudio, Incredibuild integrates as an "Add-in," providing another Build menu and toolbar. Remapping the Build Project key to Incredibuild makes it completely transparent. The Incredibuild DevStudio Add-in supplants the standard Build Output window with its own, providing extra tabs showing how many Agents are available, the progress of the

build across each Agent, and how long your build is taking. The Agent progress window is also available from the icon in the system tray. Configuration of available machines is trivial via the Build Coordinator, giving the administrator the ability to add and remove machines from the build farm through an intuitive interface.

As the build proceeds, the Incredibuild DevStudio Add-in aggregates all of the output (errors and warnings) from individual Agents into a single list.

By my conservative estimation, a 15-member programmer team would pay for the program in 50 days, and save the team the equivalent of an extra programmer for two weeks in the process. As a lead developer, that is infinitely more valuable to me than the expenditure. I would definitely recommend it for large projects, adding to the arsenal of high-powered tools available to developers.

Incredibuild is one of those software packages that after you've used it for a few weeks, disappears into the background. You still lament slow compile times, but on a sizable network they are an order of magnitude lower.

Currently Incredibuild does not support target platforms other than Windows, but Xoreax claims they are addressing issues with the Microsoft Xbox, still leaving Playstation 2 and Gamecube unsupported.

Data from my Incredibuild test results will be made available on Gamasutra.com.

★★★★★ | Incredibuild
Xoreax | www.xoreax.com

Justin Lloyd has over 18 years of commercial game programming experience on almost every released platform.

NXN's Alienbrain 6

by *jeremy gordon*

Alienbrain is an artist- and designer-friendly digital asset management system. The recently released version 6.0, however, touts beefier software configuration management (SCM) features, which when combined with a new price-



ing model (\$690 for the Developer Client, \$990 for the Designer Client, and \$1,990 for the Manager Client (all for Windows), though custom packages are available) allow you to get the whole team using the same software package for revision control.

As proof that NXN is taking the needs of programmers as seriously as the needs of artists and designers, they have split the Alienbrain offering into three different flavors: the designer client, the manager client, and for programmers, the developer client. The designer and manager clients present the familiar artist-friendly features found in version 5, but differ primarily in the depth and breadth of reports that can be generated from the integrated task-tracking system. The developer client eschews the graphical goodness in favor of a more spartan GUI, but at a price much more inline with the competition's offerings.

This time around, NXN has included the excellent Araxis Merge Professional as an integrated part of the developer client. In addition to integration into Maya, 3DS Max, Photoshop, and Microsoft Office, Alienbrain also features Visual Studio .NET integration for the programmers on the team. SCM support now includes branching, merging, sharing, and pinning. And while SCM features have expanded, they're still not as robust as the competitors'. They have also added Perforce-style change lists (for transacted check-ins) as well as big performance increases. In informal benchmarks — using the Unreal engine for PS2 code base — Alienbrain's check-in and check-outs for both large binaries and source files were as fast as or better than SourceSafe and Perforce. Several different transports exist, including SMB, HTTP, and an NXN custom protocol. With the client and server now available under

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

Windows, Linux, and Mac operating systems, Alienbrain can handle a broad variety of platform and network topologies.

Although nifty, Alienbrain's "pin" functionality isn't what I would have expected based on similarly-named features in packages such as Microsoft's Visual SourceSafe. In Alienbrain, "pin" acts more as a label editor; this is great for those times when you thought everyone understood that you were in code freeze, and so you don't have to add the "Demo for publisher (for real this time)" label when you discover that someone forgot to check-in their changes with the rest of the team.

My only real nitpick with the product concerns the GUI presentation of the delete (as opposed to destroy) functionality. Like other packages, deleted files are more like "hidden" files that maintain their version history but otherwise appear removed from the project. Alienbrain displays these files in a global recycle bin inside the folder tree. Unfortunately, expanding the bin to recover a file reveals subfolders named by date, requiring you to remember when you deleted the file. (I'm lucky if I remember where the Alienbrain icon is on my desktop, much less the day I deleted a file.)

Fortunately, the search and query functionality is very robust, and it's easy to make custom query views in HTML and Jscript (or any windows scripting host language, for that matter). Alienbrain also offers an extremely customizable client extendable with a C++, API, and Windows scripting host (WSH) access to virtually all client functionality.

With added features designed for programmers, performance gains, new price points, and the availability of evaluation versions, I wholeheartedly recommend checking out Alienbrain for your current or next project. 🍷

★★★★★ | Alienbrain 6
NXN | www.alienbrain.com

Jeremy Gordon is the president and CEO of Secret Level, a boutique game developer located in San Francisco.

 **PRODUCT REVIEWS**

Unified Rendering LOD

Part 1

Last year, in my “Rendering Level-of-Detail Forecast” column (August 2002), I discussed a few of the most popular methods of level-of-detail (LOD) management. I opined that most LOD schemes are too complicated for the benefit they provide, and I gave a high-level overview of the method I would choose to implement LOD management over large, generalized scenes.

Now I’m going to put my money where my mouth is and actually implement such a scheme. It’s a big project; the resulting source code will be the biggest this column has seen, by far.

Goals of the System

We would like our system to reduce the number of triangles used to render distant objects, where precise detail is not necessary. Some algorithms, like the Lindstrom-Koller and ROAM descendants, do this by modifying the world mesh at a granularity of individual triangles. These algorithms are unacceptably slow at high detail levels. Also, many of these systems try to utilize frame coherence, which makes them unsuitable for interactive systems.

So, we want the system to operate at a large granularity, raising or lowering the detail of big chunks of geometry all at once. Thatcher Ulrich’s appropriately named “Chunked LOD” system (see For More Information) works like this, and it runs very quickly. However, it uses the “binary triangle tree” tessellation, the same one that ROAM uses. This tessellation is very inefficient about how it allocates its triangles (though you won’t find a serious efficiency analysis in any research paper promoting binary triangle trees). Also, it means that regular-grid height fields are the only kind of geometry on which the algorithm can be easily used. With a lot of work, the algorithm can be extended to other topologies. But the amount of work necessary is large, it complicates the run-time system tremendously, and the system still won’t handle arbitrary input meshes.

Aside from topology, we want the system to place a minimal number of other constraints on mesh storage and rendering. An example of such constraints is seen with progressive meshes. It’s a complicated and somewhat inefficient task to render a progressive mesh as a series of triangle strips; it’s nigh impossible to render it as a series of vertex-cache-optimized triangle strips. If we introduce too many constraints, then when we come along to add new features to our engine, such as stencil shadows or normal-mapping geometry enhancement, we may find that constraints of the new features clash with the old LOD constraints.

This forces us to eliminate the new feature, or to dump our LOD system and make a new one. Neither outcome is ideal.

Finally, we want the system not to exhibit popping from one LOD to another. Popping looks ugly and is distracting to players; often we are looking for movement in our surroundings, and LOD popping creates false movement. Because we are choosing to adjust detail at a large mesh granularity, we will naturally get a lot of popping unless we spend significant effort to avoid it.

Reducing the Detail of Distant Geometry

My overall approach to rendering an LOD scheme will be to cut the world geometry into pieces at preprocess time, and generate several detail levels for each piece. At run time, I will choose an appropriate detail level for each piece and then render it, filling in the cracks.

To generate the various levels of detail, I will use Garland-Heckbert Error Quadric Simplification, or EQS (see For More Information). The input to EQS is a mesh of arbitrary topology, and it produces a mesh of arbitrary topology.

When reducing the mesh, I use what Garland and Heckbert call “subset placement”: I pick a vertex, drag it to the position of a nearby vertex, then remove all the triangles that have become degenerate. The alternative is “optimal placement,” where you solve for the position of a new vertex, then move two vertices into that new position. Subset placement is easier to program than optimal placement, but it probably produces a lower-quality output triangulation. Nevertheless, I chose subset placement because of its extreme generality. The output vertices are a subset of the input vertices, which means that they comprise valid and undistorted mesh data. With optimal placement, you must interpolate and extrapolate to produce the values for each vertex (not just position, but texture coordinates, perhaps tangent frames, bone blending weights for animated meshes, or other arbitrary vertex-associated data). Linear interpolation might not be good enough, or might require specialized renormalization; extrapolated values might need to be bounded in



JONATHAN BLOW | *Jonathan Blow doesn't like how fricking cold it gets in Austin during the winter. His e-mail address is jon@number-none.com.*

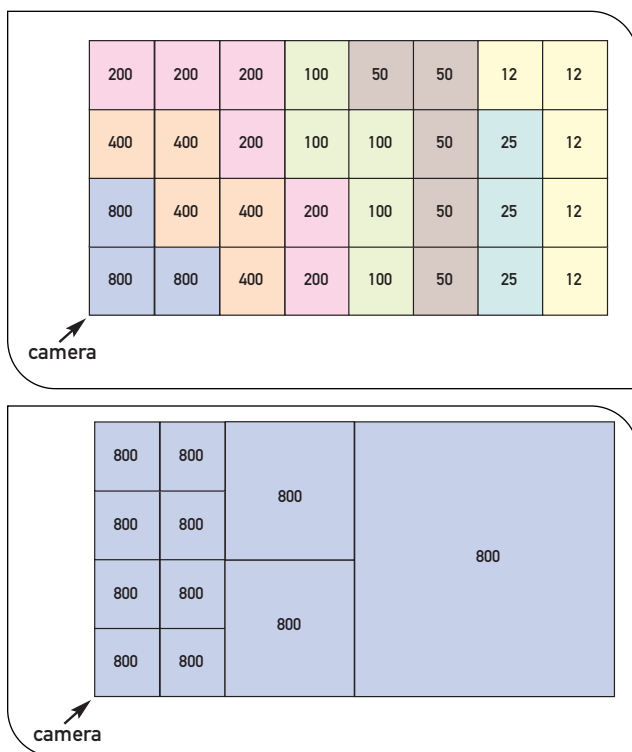


FIGURE 1A (top). The squares represent blocks of our world mesh; the numbers denote how many triangles make up each block. If we merely reduce the detail levels of a fixed set of blocks, we will inefficiently render blocks that contain small numbers of triangles. **FIGURE 1B (bottom).** By combining blocks, we can eliminate this problem. Note that the total number of triangles in 1b is higher than in 1a; we will worry about this in a future article.

application-specific ways. It is definitely possible to do these things, but there's still no guarantee that the machine-generated coordinates will look good. It costs a lot of extra work and software complexity to gain a potentially small visual benefit. Also, since the work is highly dependent on the type of data stored at each vertex, it's difficult to make a generalized tool that operates in a "hands-off" manner.

Clinically speaking, the choice to use optimal placement creates extra dependencies between the mesh simplification tool and the data format. Since dependencies are the bane of software engineering and project management, it's best to use subset placement in almost all cases.

Dividing the Geometry into Pieces

For now, in order to make it easy to divide the world into blocks, and to simplify other tasks like crack filling, I'll limit myself to rendering height fields. By the end of this series, though, I'll be operating on triangle soups. I want the initial system to be simple enough that the programming can progress quickly, so I start with just height fields. At the same time, I don't

want this to be a permanent restriction, so I must be careful not to rely on techniques that cannot be extended beyond height fields. In other words, for every simplification I impose now, I need to have a believable story about how that piece of the system will be upgraded in the future. It's a sort of algorithmic bootstrapping, if you will.

With a height field, it's easy to divide the world: you have some big array of height samples for the entire world, and you copy out rectangular subsections of that array. Because the array is evenly sampled, we can easily match up the vertices along the edges of the blocks, which is necessary for crack filling (I'll discuss this later). Note that there is no limit on the size or aspect ratio of these subsections; you can choose them arbitrarily based on your needs. On the other hand, binary triangle tree algorithms want your blocks to be square and power-of-two-plus-one in size, which is often inconvenient.

In a future upgrade, to handle unrestricted input meshes that extend arbitrarily into all three dimensions, I'll clip the meshes against axis-aligned planes to divide them into a bunch of cube-shaped regions. When clipping triangles against planes, we create all the new vertices ourselves, and in fact we create them in pairs. By saving the information about which vertices correspond, I'll make it easy to perform crack filling. But that's a subject for a future article.

Operating at a Large Granularity

Suppose we choose subsections of the height field that are 21×21 samples. That gives us 20×20 quads, or 800 triangles per block. Suppose we use mesh simplification to generate low-resolution versions of the block at 400 triangles, 200, 100, 50, and 25. Based on the distance from each block to the camera, we choose one detail level for each block and render it as Figure 1a shows. Unfortunately, most of the rendered blocks consist of a small number of triangles. Current graphics hardware and drivers do not like that very much; the game will run slowly.

To solve this problem, we can hierarchically combine terrain blocks before putting them into the EQS routine to reduce them. Since we're working on a height field, I chose to combine four blocks at a time, prior to simplification. If the original blocks are 800 triangles each, I combine four of them to get a 3,200-triangle block; then I simplify that mesh back down to 800 triangles. Thus all the rendered blocks have the same number of triangles, regardless of scale, as you see in Figure 1b. This simplifies some of the math we'll look at later on.

With fully 3D input geometry, I would be combining eight blocks into one, requiring a more extreme reduction ratio. You may wish to think about the implications.

Filling Cracks

After rendering two blocks, we need to fill the gap between them by rendering an appropriate set of connecting triangles. I will call this set of triangles a seam.

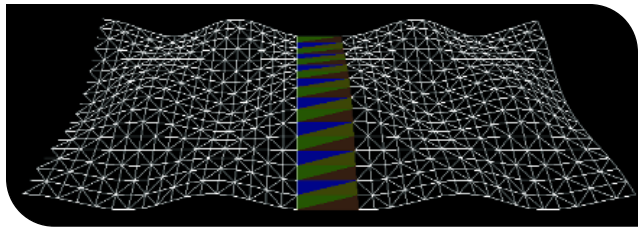
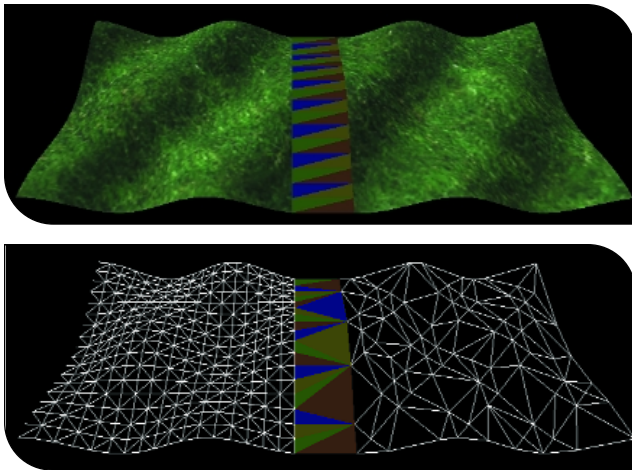


FIGURE 2A (top left). Two blocks of terrain, connected by a seam. The blocks are drawn with an exaggerated gap between them; in actual game rendering, they would be touching. The seam is drawn with intentionally strange coloring to show the individual triangles. **FIGURE 2B (top right).** A wireframe version of 2a. **FIGURE 2C (bottom left).** The right-hand block has been reduced to one fourth of its original triangle count. The seam is created by cross-referencing the seam of 2b through an index map provided by the mesh simplifier.

Because all of our blocks are precomputed, we can also precompute the seams. At run time we only render precomputed arrays, which doesn't take much time at all.

Most people think of the crack-filling problem in a way that makes it needlessly difficult. They think about how, when given two arbitrary-LOD blocks, to match up the vertices along their edges. Fortunately, we don't need to solve this difficult problem.

Instead, we can first build seams between the highest-LOD blocks. You can accomplish this easily by just iterating along the array and generating a row of triangles (see Figure 2b). For each vertex of each seam triangle, you store an integer telling you to which block the vertex is connected, and another integer telling you the index of the vertex inside that block. So we're only storing indices, not spatial coordinates. That's an important fact to remember for the next step.

Suppose we reduce one of the blocks as shown in Figure 2c. All we need is for the EQS routine to tell us which vertex in the source mesh corresponds to each vertex in the destination mesh. That's especially easy, since we used subset placement; when we collapse a vertex into another vertex, we simply record the index of where it went. When detail reduction is complete, we return an array that tells us "For each index in the source mesh, here is the index of the corresponding vertex in the output mesh." Now we use this array to remap the indices of the seam triangles, and we throw away any degenerate triangles. Now we have a valid precomputed seam for two blocks at differing resolutions. We can repeat this process as often as we want, reducing either block as much as we want. When we combine blocks, we merge their seams. (Seams that end up interior to the block are tossed in with the block's regular geometry; seams on the borders are merged to make bigger seams.)

The seams between the highest-resolution blocks, which we generated from the original height field-sampling pattern, are not necessary for rendering. If we were to render them, all their triangles would be collapsed to zero area; Figures 2a-c are drawn with the blocks artificially pulled apart to make the filling pattern clear. So we throw those seams away after the preprocess, keeping only the seams that involve detail-reduced blocks.

Now the question arises: How many of these seams do we need to precompute, and how do we organize them in memory? Clearly we need an appropriate seam between a given block and any of its possible neighbors. To reduce the number of possibilities, I decided to place a restriction on the LODs of rendered blocks: two neighboring blocks are not allowed to differ by more

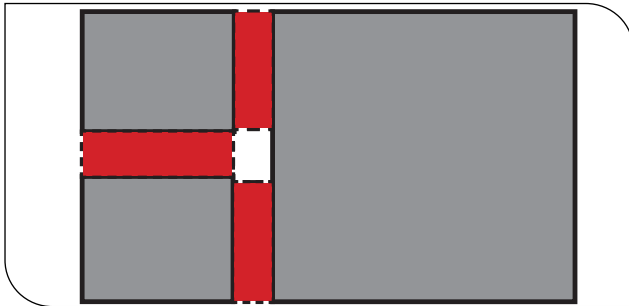


FIGURE 3. Three terrain blocks (gray with black borders) and the seam fills between them (red with dotted borders). Note the hole in the middle. These blocks are drawn with exaggerated gaps; the actual hole would be very small.

than one level of detail. So for any given block, all its neighbors will either be the same size, half-size, or double-size.

If every block stored all the seams for all its neighbors, we'd be storing every seam twice: both block A and block B would store the seam that attaches A to B. Instead, I store only seams for neighbors in the +X and +Y directions.

In the end, each block has eight seam pointers. There are four for the +X direction and four for the +Y. The four pointers are:

seam to lower-resolution block, seam to same-resolution block, and two seams to the two higher-resolution neighbor blocks.

After all this seam filling, there will still be some small holes in the terrain, where the corners of blocks meet, as Figure 3 shows. For the case of a height map, these can be filled very quickly at run time. I won't explain the details now, since the procedure will change significantly when we adapt the algorithm to generalized meshes.

Sample Code

This month's sample code (available from the *Game Developer* web site at www.gdmag.com) renders a height field with filled seams. It also does some work to eliminate popping. Next month I'll discuss popping in detail and look at methods of choosing which LOD to use for a given block. 🐉

FOR MORE INFORMATION

Garland, Michael, and Paul Heckbert. "Surface Simplification Using Quadric Error Metrics." *Siggraph 1997*.

<http://graphics.cs.uiuc.edu/~garland/research/quadrics.html>

Ulrich, Thatcher. "Chunking LOD"

<http://tulrich.com/geekstuff/chunklod.html>

Creating

Creatures

As a young child, I never once remember having any trouble starting a picture. Perhaps it was the rampant imagination of youth, but as I have become older, I have often fallen victim to the Curse of the Blank Canvas.

Faced with nothing but empty space in front of me, I find it difficult to kick-start my creative juices (if indeed juices can be kick-started), and I scribble junk for hours until I'm forced to give up and skulk away into a dark corner, defeated.

It is true that as artists, we are sometimes in "the zone," and creativity explodes from us like lava from Mount St. Helens, but the reverse of this elevated state seems to be the great white void that occasionally finds form in empty sheets of paper that stare blankly at us from our desk, daring us to fill them with something worthwhile. Like writer's block, it is an affliction that can cause the sufferer no end of torment; valuable time can be wasted if it drags on for too long.

Artists over the years have developed many approaches to help keep the ideas flowing, and what follows are some of the methods I find useful, specifically when dealing with the area of creature design.

Evolving

The game itself may require certain creatures to evolve in some way. However, even if the evolution process isn't part of your game, the design aspect of evolving a creature can lead ideas in interesting directions.

If time is not too much of an obstacle, you can choose to illustrate as many evolutionary steps as you wish, starting with a quite simple creature and evolving it

into an imaginary, intelligent descendant. If you're more pressed for time, you can perhaps take a sketch you have worked on previously and attempt to evolve it an extra step, seeing where that takes you.

Why bother with this approach? The main thinking behind evolving some-

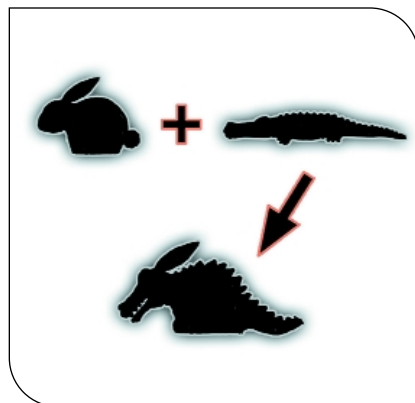


FIGURE 1. A Crocobunny, the result of combining two different animal forms.

thing, rather than simply drawing another version, is that this process forces the artist to think about the world within which the creature exists, as well as the specific characteristics of the creature. Looking at how a creature works in its particular habitat and how it could change to better adapt to its surroundings could seem like design overkill. But coherence of a game world and its inhabitants can add to the play-

er's immersion, while also upping the level of visual quality. You may even be left with a few creature designs that can be used, resources permitting, in areas where the environment is reasonably consistent.

You can also use the same principles you've used in the evolution process to work backward toward a more primitive life form. You can use these more primitive characters to populate parts of a world where some of the principal character design has already been done, in order to provide more consistency in those areas.

Collision

Some of the craziest ideas often seem to be the ones that catch on. A man that turns green and sprouts an insane amount of muscles when he gets angry can't have been a completely straightforward pitch, even in the world of comic book heroes. Who would have thought that a game that lets players watch semi-autonomous characters mimic the mundane tasks of everyday life while they give them simple commands and buy them new furniture would practically outsell the Bible?

So, in the spirit of mad invention, another method for generating ideas for creature design involves the juxtaposition of two (or more) unconnected ideas, which then need to be absorbed into a



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

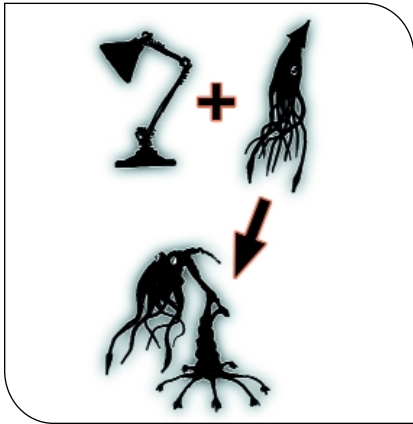


FIGURE 2. A Squidlamp, combining an animal with an inorganic object.

single compound creation.

Here are some example methods to start the ball rolling:

Animal combinations. This is pretty much as it sounds, but with some imagination and a bit of help from someone else, it can be worth a try. Essentially, you need two lists of animals (any living creatures, in fact), preferably created by two separate people. These lists are then paired up and you attempt to draw the resulting creature combinations.

This can obviously be dull if the two lists aren't that imaginative, but with some work, you can achieve interesting results. Figure 1 shows a "crocobunny" conceived through this process.

Extreme animal combinations. This time your second list is of any object you can think of, not just animals. Sometimes the obscure collision of a living creature with an everyday object can work. Figure 2 shows a "squidlamp" that resulted from one experiment.

Group Design

The group design approach is quite popular, and can take several forms. The most straightforward method is to have a group of artists begin drawing a creature, and after a short period pass their work to the next artist who then continues it, and so on until the creatures are complete.

This mixture of styles and ideas is

often more hilarious than useful, but it can certainly help shake things up if they have become stagnant.

Good and Evil

As most famously illustrated in a game context with Lionhead's *BLACK & WHITE*, creature designs can be modified along the lines of good and evil. After starting with a creature that looks harmless or benign, the process of making this creature look evil can yield interesting results. Exaggerating aggressive features such as teeth and claws, lowering the central brow area, adding gratuitous spikes — all these methods can create the effect of evil. In *BLACK & WHITE*, the use of a domesticated farm animal (a cow, the very essence of friendliness) made a great starting point for turning it evil. This process is usually most interesting if the point of departure is especially cute or fluffy.

In contrast, redeeming an evil creature to reverse the process and make it look friendly and harmless can also be good fun. The best starting point for this procedure can often be creatures that are already present in films, books, TV, and so on, where plenty of quality time has already been put into designing the very epitome of menace. Try, for example, to make Giger's alien look cuddly, or the Uruk-hai in *Lord of the Rings: The Two Towers* look like they just want to sit down for a friendly game of chess. It may sound like ripping off someone else's design, but if taken to extremes, the resulting creature is almost certainly going to be a completely original creation.

Playing with Scale

I hate spiders, and even though the worst a spider in the U.K. can do is run menacingly across the floor in front of you, when seen up close with all those eyes and hairs and giant pincerlike fangs, well, I'm just glad I'm not half an inch tall. The world of insects and microscopic creatures is a goldmine for interesting creature design ideas.

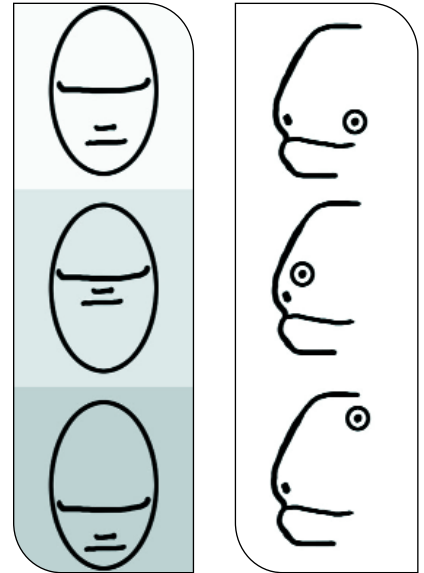


FIGURE 3 (left). Feature placement for three basic faces. **FIGURE 4 (right).** Simple alterations to feature placement can yield significant results.

Most of us have seen electron microscope images of such things as dust mites and any number of tiny parasitic beasts, the majority of which are weird and incredible and often quite scary. Rescaling these creatures to the size of a lion or elephant can be an easy way of coming up with interesting-looking creatures that actually have reference photos from which to work. Sometimes, it will be best to adapt these tiny creatures somewhat, so that they look less insectlike and more mammalian, by removing antennae or replacing mandibles with a regular jaw.

The opposite scale process is usually less successful. Large creatures (unlike their microscopic counterparts) are very recognizable, and a small rhino, for example, isn't exactly exciting. In general, a game world deals best with creatures that are around the player's scale or above. In this respect, taking larger animals and shrinking them usually isn't going to yield impressive results.

Feature Positioning

Creating new and interesting creatures isn't just about the initial

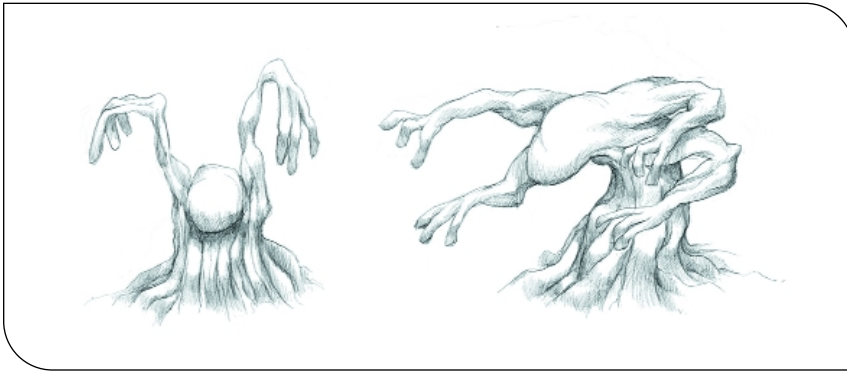


FIGURE 5. Weak and strong variations of a creature.

design ideas. Once you are happy with certain elements of your design, or once you have a specific idea that you feel works, there is plenty of scope for small-scale changes to refine your design.

For instance, you'll have to address the issue of feature placement. Consider faces: implicit in our interpretation of faces are certain ideas about the character traits of the person or creature we are looking at.

Figure 3 shows three basic faces all made from the same lines. The top face represents normal human feature placement, the middle face has its features compressed to emphasize the chin and give the impression of a lower brow, and the bottom face has a significantly higher forehead. Crudely speaking, we are inclined to interpret the second of these faces as belonging to a person of lower intelligence, and the third as someone of higher intelligence.

While the exact meaning of where features are placed may not be that important for a particular design, it is sometimes worth experimenting with these features, to see if the same basic design can be improved by moving them. You can see in Figure 4 how, by changing the placement of the eye, you can change the interpretation of a creature's personality.

Other Things to Try

Big brother, little brother. Using either a design you have already done, or an existing creature from another source, attempt to create both its big and little brothers. This is not a matter of merely making scale changes but extrapolating a design for a particular creature to three stages of its development, assuming that the starting point you have is the middle stage. Once again, resources permitting, having three age-dependent variations of the same creature can add depth to a game's visuals, especially if animations and sound can also reflect these three states of maturity.

Creativity is sometimes about industry, method, and finding ways to help crystallize ideas and explore them in new directions.

Random word generator. In the same way that music can sometimes create a visual impression in your mind, so can words. Using one of the random word generators on the Internet (www.fourteenminutes.com/fun/words, for example), attempt to derive a creature

that fits a name generated at random.

Weak and strong. Keeping in mind the same concept as Good and Evil, create two variations of a source creature: one that is pathetic and weak, the other that is powerful and intimidating. Figure 5 shows how this use of contrast can be effective.

Alien food chain. Starting as low down as you like, create a food chain (entirely imagined) that leads to what would be an apex predator. If you take this food chain seriously (within reason), it should take into account such things as habitat, and include a physiology that would lend itself to catching and eating the creature that is one step below.

Additional limbs. Take a design and add extra legs, arms, or both. Rework it so that the creature is able to maintain a reasonable center of gravity, and so that it doesn't look like you've just added extra limbs for the sake of it.

From the sea to the land. Taking one of the exceptionally bizarre creatures of the deep and transforming it into something that would survive on the land can produce interesting results.

Silhouette or outline. You can try this technique on your own or in a group. Draw a creature in silhouette (or just its outline) and then fill in the detail. This way you can avoid the sometimes daunting

task of creating a whole detailed creature in one go. The outline describes its mass and basic form, and then you can fill in as much or as little of its features as you find necessary.

Even if you don't always find these methods usable in practice, elements of the individual designs should stand out as interesting or worth pursuing. After a while, a workable result can begin to emerge.

Creativity is sometimes about industry, method, and finding ways to help crystallize ideas and explore them in new directions. Ultimately, the more industrious you can get, the less you'll have to stare at a blank sheet of paper waiting for inspiration. *✍*

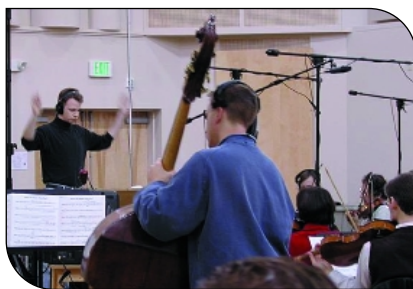
Using Living, Breathing Musicians in Game Music

It is the composer's job to add as much realism to game music as possible to bring the player into the experience. I can't think of a better way *not* to achieve this goal than to produce orchestral music solely with samples.

Whether you record a small group of players to sweeten your tracks or an entire orchestra, the result of live performance is so much richer and more satisfying to the player. Yet I still hear so many lost opportunities when game music is produced only with samples. Don't get me wrong: many, many tracks sound great produced only with samples. But there are certain instances when it is simply unforgivable, such as the use of solo instruments or melody instruments that clearly fake real ones.

According to the NPD Group, by the time all the holiday dust finally settles, the North American videogame market is expected to hit \$10 billion for 2002. That's a lot of scratch. Advertising in 2002 for the game industry is up more than 247 percent over that amount in 2001. Makes sense, right? Tons of ads on TV for dozens of games. And then you go to the movies and there, among the 20 or 30 minutes of movie previews, are a few game trailers playing in the same theater. That's exciting! But I have one question: Why am I listening to a MIDI track in these game trailers, while film people would never even consider anything but a live recording of the soundtrack for their trailer?

Of course, there are budgetary constraints. But here we are, getting right up there in gross bucks to film, and the music recording doesn't cut it. Next to the best offerings from John Williams, Hans Zimmer, and their ilk, it doesn't hold up. The composition may be great, it's just the sound of the recording itself that makes me shrink in my seat. One guy in his studio working with samples will never produce a sound as rich, as



The author conducting the Northwest Sinfonia in Seattle.

powerful and, well, as good as a recording with 60 to 90 virtuosi expert at the craft of music performance. Just ask the many game composers and companies that decided over the last two or three years to go ahead and record an orchestra for the game. People are getting it. But many still haven't.

With 30 percent of all console players hooking their consoles up to 5.1 surround systems, more games are being played in the same place in the home that movies and music are. We are now also competing directly, at least musically, with these entertainment genres. For the most part, we're making great strides. The Game Audio Network Guild is diligently working with companies like DTS and Dolby on the 5.1 front. More composers and sound designers are becoming vocal in preproduction about these issues, and it's making a big difference. As a composer and music producer for games with experience in the film and music worlds, my only real complaint is that we need to hear a lot more live recorded music in our music. I don't want to hear a sam-

pled solo cello in a game trailer in a movie theater next to Hans Zimmer's latest recording at Air Studios. It feeds the myth that games are standard entertainment, "kids' stuff."

The making of videogames is a true multimedia art form, and as all other elements of this art form get more real and immersive, so should the music. Music sounds more real and honest if you don't try to imitate real instruments with fake ones. Electronic music is honest because it is what it is. But an electronic cello doesn't sound like a living, breathing musician playing a cello. Regardless of your budget, you have to make sure to record at least your melodies, your important parts, with realism — that is, with real players.

As Andy Brick wrote here a few months back ("The Live Orchestra Recording: A Producer's Awakening," *Sound Principles*, December 2002), it's O.K. once in a while to foot the bill out of your own pocket. But first, ask your designers and producers to add a line item in the budget for recording music. You never get what you don't ask for, and in the face of a \$10 billion industry, it's peanuts. Whether you use a full orchestra or just a few players, the difference in the quality of your music will be enormous.

If your budget doesn't permit the recording of a live orchestra, then do your best to record enough living, breathing musicians to bring the game alive. And please, if you are composing for a game trailer that will play next to John Williams' *Episode III* trailer, remember . . . it will be playing next to John Williams' *Episode III* trailer. 🎻



JACK WALL | Jack (www.jackwall.net) is an L.A.-based game music composer whose credits include *MYST III: EXILE*, upcoming titles *BEN HUR* and *WRATH*, and additional music for *UNREAL 2*. Jack is also the senior director of the Game Audio Network Guild (www.audiogang.org) and event producer for "Video Games Live at the Hollywood Bowl" (www.videogameslive.com).

Evolving The 400

This column will appear around the time of the 2003 Game Developers Conference, two years after Hal Barwood's "Four of the 400" lecture provided the impetus for what has become The 400 Project. The idea of creating a compendium of rules of good game design has from the start been intended to produce a practical collection of tools. Instead of introducing a new rule this month, I'll focus on how to use the rules I've discussed thus far effectively. To facilitate their use, all the rules are now on display at my web site, www.theinspiracy.com.

Stone knives and bearskins. How do we become tool-using game makers? There is no one right answer, and some experimentation is necessary at this relatively early stage when there are many rules still to be listed. Some people work best intuitively, and if you are one of them I would suggest reading over the rules, pondering the examples and counterexamples, and letting them sink into your subconscious as you work on a game.

Other people are more systematic and orderly. For them, I suggest paying attention to the reports of play-testers or others on the game team when they voice concerns about the game, then considering the various rules in turn to see if any may apply. If you've found your own methods to apply these rules, please e-mail me and let me know.

The Bronze Age. Whether or not there are really 400 rules of game design, you can put the handful assembled so far to good use. They serve everyone on a project, from the leads to the greenest recruits. If you're a staff artist, you might find "Begin at the Middle" (October 2002) useful, beginning your initial illustrations or models from scenes or levels near the middle of the game. A programmer could take care to set up the game



Color combinations in PUSHER provide clear short-term goals.

variables efficiently so that the game can be saved at frequent intervals, or implement some of the simple AI rules covered in January 2003 ("AI without Pain").

Game designers and producers or project leads can apply virtually all of the rules at one time or another, and may find them a helpful unifying influence for the team. An upcoming rule that bears mentioning is "Provide a Single, Consistent Vision." Anyone wishing to use these rules to change the course of a game's development should clear it with whoever is in charge, and those of you in company management should be sure to make it obvious who the person in charge is!

Ironing out the wrinkles. It has been convenient for me as the keeper of these 400 Project rules to put them into practice, and I've found them very helpful in my work. In the recently released puzzle game PUSHER, from JoWood's Vienna studio, I did a design review and was able to suggest some changes based on several of the rules I've published here. The addition of

a small set of three-color combinations provided short-term goals (March 2002) as well as a parallel challenge with mutual assistance (April 2002). The core gameplay involves pushing colored spheres around a playfield to score points by lining up three or more spheres of the same color. Adding combinations gave the players additional clear goals of color sequences to aim for. Mutual assistance comes from completing combinations to grant extra time or change the playfield in various helpful ways that make it easier for the player to succeed, while simultaneously the core gameplay makes the combinations possible.

What I've discovered is that this toolbox of game design rules does not usually make it easier to come up with design concepts or changes, but it does make it easier to be sure that the concepts or changes will be effective. One of the hardest things to do as a designer is to project how design decisions will affect the final gameplay, and anything that makes that easier is welcome.

Steeling ourselves for the future.

Eventually, I expect that the game industry will have a wide variety of well-tested and thoroughly documented tool sets and methods to use them. As with most tools, they will not make you more creative or talented, but they can help you apply what creativity and talent you do have more efficiently. To apologetically paraphrase Archimedes and Origin Systems, give us the right tools and the time and funds to use them, and we can create worlds. 🛠️



NOAH FALSTEIN | Noah is a 23-year veteran of the game industry. His web site, www.theinspiracy.com, has a description of The 400 Project, the basis for these columns. Also at that site is a list of the game design rules collected so far, and tips on how to use them. You can e-mail Noah at noah@theinspiracy.com.

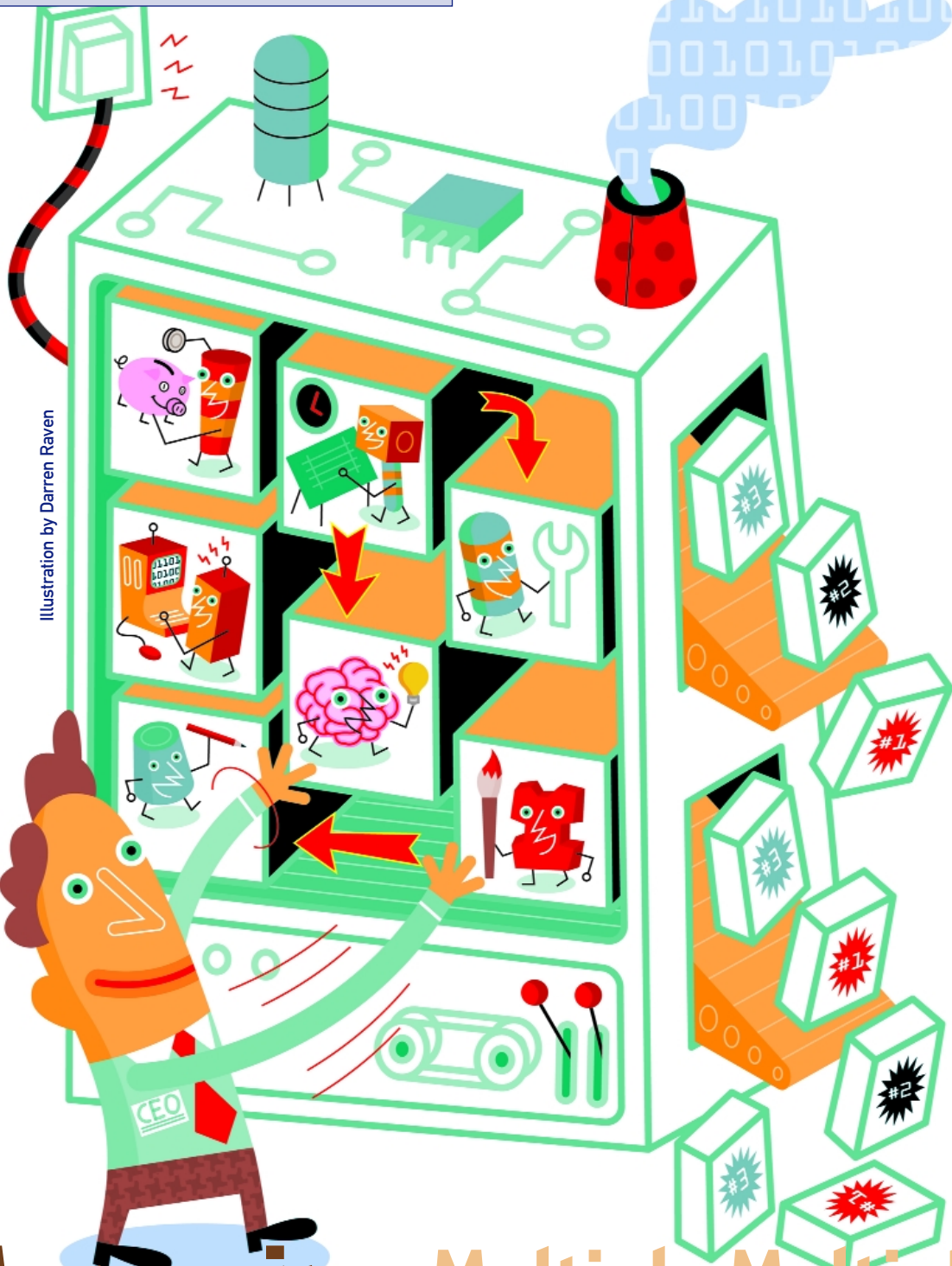


Illustration by Darren Raven

Managing Multiple Multiple Multiple Multiple Multiple Multiple Multiple Projects

After two years of preparatory work on medical education projects and early game engines projects, BioWare Corp. was incorporated in February 1995 by three medical doctors who had no game industry experience whatsoever but did have a strong desire to make games. The staff who worked on SHATTERED STEEL and BALDUR'S GATE, BioWare's first two titles, similarly came from a range of non-game industry jobs. This collective naïveté in those early years afforded us the freedom to make some management and organizational decisions that ultimately proved very fortunate. Our collective initial lack of experience has permitted us (with additional training by both CEOs in executive MBA programs over the past few years) to view solutions to a management problem from multiple, fresh angles.

In this article, we provide an overview of the challenges and potential solutions to issues that center around creating a successful multiple-project development company, and sustaining the high-performance culture required to keep that studio running.

A Common-Sense Approach

Two years after forming BioWare, we made the decision to run multiple simultaneous projects. There were three of us who started the company, and we each wanted to participate actively in creating a game. At the time, we really had no idea how to successfully manage multiple projects. Aside from the obvious hurdle of making great games, a more subtle challenge we faced was setting up



BIOWARE: CLASS OF 2002. Currently the company has 140 employees working on five different projects.

the systems and structures that would eventually allow BioWare to grow to its current size of 140 employees working on five concurrent projects.

In most cases the decisions we made as BioWare grew were generated on the fly rather than strategically planned. In retrospect, we could describe our methods as being developed as part of a carefully designed framework, but in reality, all of our methods were grounded in pure common sense, as we chose whatever best suited the situations we were facing. We continue to use common sense, both in creating solutions to the company's ongoing challenges and as a foundation for BioWare's management style.

In this article, we'll detail BioWare's common-sense approach to developing multiple simultaneous projects in a dynamic and challenging environment. We'll discuss the methods used in creating an environment capable of generat-

ing multiple simultaneous AAA games while managing to stay completely independent. We'll describe two distinct phases in the process of building successful multiple-project studios: the establishment phase and the ongoing management phase.

The Establishment Phase

The establishment phase includes the decisions you should make prior to beginning development on any products, and some of the actions you should take when setting up your studio, or transitioning from single-project to multiple-project development. Starting with a blank page is both daunting and exciting; if you had a chance to build (or rebuild) a studio from the ground up, what would you do? Would it be an easy or a difficult task? Choose wisely, as your company will often be forced to live long-term with any decisions you make during the establishment phase. Most of us don't have the luxury of starting again from scratch, but often new ideas will occur to you if you consider the optimal solution, one that isn't blocked by existing systems. The following ideas will be equally applicable to both start-ups as well as existing single-project studios hankering to grow.

DRS. RAY MUZYKA AND GREG ZESCHUK | *Ray and Greg are the joint CEOs and co-executive producers at BioWare Corp. Currently they are working on BioWare's upcoming project with LucasArts, STAR WARS: KNIGHTS OF THE OLD REPUBLIC, as well as the recently announced but unnamed original-IP Xbox (to be published by Microsoft) and PC titles (no publisher announced yet), and two NEVERWINTER NIGHTS expansion packs. Ray has recently finished his executive MBA program at the Ivey School of Business, University of Western Ontario. Ray is also a board member of the Academy of Interactive Arts and Sciences and co-chairman of the Business Committee of the International Game Developers Association. Greg is currently completing his executive MBA at the School of Business at Queen's University and is a board member of the International Game Developers Association.*

Setting Company Goals and Values

Figuring out what you want to do will be the first step in establishing a new studio or reorienting your existing one. Determine your company's goals, then set your company's values, which will ideally be derived from these goals. Do you want your company to make a small number of only AAA games, or do you want to create a massive organization that will pump out dozens of titles per year? In BioWare's case, we chose to create a small number of extremely high quality games. We continually added to the goals, eventually building our twin company values of quality in the workplace and in our products, but we never changed our path.

One of the keys in this process is to ask your employees continually what their vision is. They need to have a similar goal to the founders' and buy into the benefits that growth can bring. Otherwise, the growth plans will be less likely to succeed as the company changes over time.

Linking Corporate Culture to Your Company Goals

The next step in the establishment phase is to build a solid foundation at your company by aligning your company culture with your goals. This might sound like an airy-fairy management concept, but it does have significant value: For example, if you announce that your company is going to work exclusively on AAA games that are "done when they are done," and yet you keep getting your staff to cut corners to get your games done on a very tight schedule or low budget, you are going to have a very confused (if not angry) group of employees.

If the culture is one striving for excellence, then, to paraphrase Bill and Ted, you need to be excellent in every way; if you say one thing but do another, people are going to be unhappy. Aligning culture and goals is an ideal way to help your people make decisions; almost all of

BioWare's decisions are made by referring back to our company values (quality in our products and in our workplace). Because BioWare's culture is aligned with its values, people automatically know how to act — which helps as the company grows and gets more complex.

From the very beginning, we worked to establish a culture that would be congruent with building multiple projects. Everyone that joined BioWare became aware that we were either working on — or planning shortly to work on — multiple games simultaneously. As a company, it was much easier to start working on multiple projects, than to convert from a long-term, single-game studio to building multiple games. However, we believe based on discussions with other developers working on multiple projects that the transition from a single-project development studio can occur in much the same way if the same steps are followed.

One of the keys to growing a multiple-project studio is setting the correct expectations collaboratively with your employees, such as aligning the compensation and reward systems within the goal of working on multiple projects. There are two potential alternatives in compensation structure: team-based compensation and company-based compensation. If you choose to reward people on the basis of how well their individual project performs in the market, you will have a strong team spirit but most likely a weaker company spirit. If you reward employees based on the company's overall success, you might not see the same individual motivation a team-based system instills, but if everyone is treated well, they will likely pull together for the company's overall success.

BioWare chose to pursue the company-based compensation system through both stock options and a yearly gain-share (a variant of profit-sharing) based on the company's overall success. Our fundamental motivation was to create a system where people are willing — and happy — to help out on projects that need help, even if it's a game in which they've invested little personal time or

MAJOR DIFFERENCES BETWEEN MULTIPLE-PROJECT COMPANIES AND SINGLE-PROJECT COMPANIES

HIERARCHY AND MANAGEMENT

STRUCTURE: You will need an upper level of management to sort out personnel issues and focus on business development in a multiple-project company, while you might be able to get by without this in a single-project development studio.

Expect conflicts to exist between project managers based on personnel moving from one project to another project in a multiple-project studio, and recognize someone needs to either prevent these conflicts or resolve them when they do occur.

CASH FLOW: Cash flow should be improved if you have more than one publisher for your titles in a multiple-project development situation.

Single-project studios may experience cash-flow problems near the end of a project unless their advance structures are carefully planned to cover the "project gap."

INFRASTRUCTURE: Expect the need for dedicated infrastructure personnel (admin such as HR/finance and systems administrators) in a multiple-project company, while these people may be part-time in a solo-project studio.

ADDITIONAL BUREAUCRACY: Some formalized processes and structures may be required in both more mature single-project studios as well as multiple-project studios for things such as compensation, education leaves, HR benefits, training, and so on. These may need to be formalized a bit sooner in a multiple-project studio.

Larger companies can no longer evaluate personnel issues on a case-by-case basis, but instead need some general policies that can be applied to a variety of situations.

effort. We found this system creates an environment where the next-to-be-released project becomes the most important game in the company; everyone knows that if that next game does well, they will do well.

ORGANIZATIONAL OPTIONS FOR A MULTIPLE-PROJECT STUDIO

TEAM-BASED APPROACH:

- No overlap between projects; each team is separate and shares neither personnel nor technology.
- Pro: No problems with team resource management conflicts.
- Con: Little intracompany spirit, and little sharing of learning, ideas, or technology between teams.

DEPARTMENTAL APPROACH:

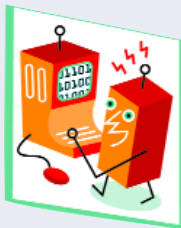
- Most development duties overlap and are assigned to specialized groups that take care of their one task.
- Pro: Easier scheduling and solutions to personnel management.
- Con: May not promote team spirit.

MATRIX:

- Effectively a hybrid of the team and departmental approaches, drawing on the strengths and weaknesses of both.
- Everyone is on both a team and a department.
- Often have competing goals from the team and department, hence this structure needs strong leadership, strong corporate values and goals to ensure that the leadership is oriented in the same direction on both axes, and extremely clear communication to be effective.
- Can be challenging to manage, yet powerful if effective.
- Based on the inherent conflict between the team and the department and between teams.

LOOSE OR UNSTRUCTURED APPROACH:

- Multidisciplinary teams form and dissolve on the basis of the needs of a project. This is more chaotic than the preceding approaches.
- Pro: Much less management or structure is required than with preceding approaches.
- Con: The success of the team and project is much more directly related to the personal qualities of the team members. With incorrect team members, an unbalanced team, or poor senior management, this is guaranteed to fail.



Structure

The studio's specific structure is another factor that must be decided on during the establishment phase. It is also possible to change the company structure mid-stream; at BioWare we've changed company structure at least three times. We started out as a team-based studio with one team, then added a second team. This transitioned to a mixed team and departmental structure as we worked on two concurrent projects, then we merged the departments with the teams into a formalized matrix structure as we reached three concurrent projects (BioWare has refined its matrix since that time and is now working on five concurrent projects).

At least three different ways exist to structure your studio: a project-oriented structure, where projects are clearly distinct from one another; a departmental structure, where people are pulled from pools of expertise (with groups of artists, programmers, and designers) and only marginally assigned to projects; and a matrix structure, which is a combination of the two along a spectrum.

Each distinct method has its pros and cons, and each one is suited for particular cultures and goals, as noted in the sidebar, "Organizational Options for a Multiple-Project Studio." BioWare chose a matrix structure, because we wanted to pursue a path of shared tools and technology where fluid personnel resources could be shifted depending on the needs of the projects. The matrix structure also supports our overall company culture where BioWare is the team, and everyone is always willing to help each other whether they are on the same project or not. One of the specific challenges related to a matrix structure is the need for close communication between projects and departments to allow resources and staff to move between them without disrupting the completion of specific projects. We provide more specific examples of the forms of communication used at BioWare in the following sections.

Ongoing Management Phase

Once you've created, or rebuilt, a solid foundation to continue to build upon, you can complete the establishment phase and move on to the second phase, ongoing management. This phase requires working out the rules of engagement.

The rules of engagement come into play whenever there is conflict between the projects or departments. Most conflicts arise in one of two scenarios: projects are competing for resources, or resources don't neatly fit into departments. The rules of engagement predict the problems and establish processes to work out the inevitable disagreements that occur when competing for a limited amount of resources.

Organizational Substructures

At BioWare, we've established what we call synchronization meetings for each development discipline (art, audio, programming, QA, and design), in which we discuss the usage of current resources and plan for upcoming resource requirements. These meetings are essential in making sure that the people needed to do the job are working on the most appropriate projects. We strive to include all the stakeholders in these meetings, so the people in attendance include the department director, discipline leads, the producers, the co-executive producers (us), and our HR manager.

The goal of the synchronization meetings is simple: to work out all resource issues in detail so all projects have the resources they need to do their job. As it seems we're perennially stretched to the limit with regard to supply and demand for staff, this is a tall order. But thanks to careful shuffling and a lot of hard work on the part of the employees at BioWare, things always seem to keep functioning at a high-quality level.

Exact conflicts to emerge during these resource allocation meetings. At BioWare, two of the ways we reduce these potential

BENEFITS OF BEING A MULTIPLE-PROJECT STUDIO

CASH FLOW:

- Cash flow is more stable if your revenue streams are diversified (multiple publishers), and ongoing (multiple products in the market).
- Overall the company should be more stable and better able to weather a problem with one publisher or one project not selling as much as expected.

LARGER, MORE FLEXIBLE WORKFORCE:

- Easier to put out fires on projects, since there may be free personnel available to be reassigned.
- Allows more specialization of personnel and more opportunities to let people focus on what they like to do.

ADVANCEMENT:

- Employees have more room to grow into technical leads or managers (or both).
- Employees have more opportunities to pursue different kinds of projects without having to leave the company.

TECHNOLOGY LEVERAGING:

- Technology developed for one project can be used on more than one project at a time, reducing overall cost per project.
- Common tool paths can be established in some cases, reducing training time for new, one-use tools.

COMPANY-WIDE PROCESS IMPROVEMENTS:

- Some things that may be impractical for smaller developers may make sense for larger studios, such as more aggressive benefits packages, dedicated HR and finance staff, and full-time systems/network administrators.

conflicts are to have a lot of communication between project leads and department managers before the formal synchronization meetings, and to clearly assign priorities to our projects based on clearly stated objectives. These objectives might include: next project in the pipe, next full project (versus smaller projects like expansion packs), or potential profitability forecasted for the various projects.

In addition, these synchronization meetings give us a clear indication of our hiring needs for the short-, mid-, and long-term, as well as direction on where we should focus training and growth

endeavors. We try to avoid changing priorities mid-stream unless the factors that we used to assign the priorities (next in the queue, profitability, scope, and the like) have changed, or one or more of the projects are experiencing unanticipated problems with scheduling.

It's important to point out that not all work is organized during our synchronization meetings. We have created an environment where individuals are often motivated to make the effort to help out other projects informally — either by helping to show people some of the methods used on other projects, or by doing odd tasks on other games. Everyone at the company is always busy, but people aren't too busy to help their coworkers.

Communication

In the ongoing management of a multi-project, matrix-structure studio, communication is one of the most important elements to consider. We strive to make sure there is continuous communication between projects occurring at multiple levels, such as producer meetings, full-team meetings, team leads meetings, and interproject departmental leads meetings. For example, at producer meetings our project leads discuss issues facing their projects as well as general company topics. Often informal and designed to allow everyone to talk to each other, producer meetings also serve as a learning opportunity for less experienced production staff.

As BioWare has grown, we've instituted a number of different company-based meetings and gatherings in order to discuss company information. These include full-company monthly meetings, and what we call "yearly" meetings, where everyone hired in a particular year meets with us on a monthly or bimonthly basis to ask questions about the company. A quick and visible response to problems identified during these meetings is essential in dealing with issues that come up. If you don't take care of things quickly and definitively, other problems result.

Sometimes communication systems can be expressed in the form of subdepartments within a department. Two art

POTENTIAL PROBLEMS OF BEING A MULTIPLE-PROJECT STUDIO

COMMUNICATION ISSUES:

- The feeling of "family" can be lost as a company grows. Maintaining it requires new types of communication systems different from what the typical small or single-project developer requires.
- Fragmented communication or rumors can be more common in a larger company.

DAILY KNOWLEDGE AND INVOLVEMENT BY FOUNDERS:

- Delegation is critical with multiple projects: if the founders of the studio are not willing or able to delegate, a multiple-project studio is likely doomed.
- The founders will inevitably either have to delegate overall management of the company or of the day-to-day project development. There isn't time to do both in a larger company.

DANGER OF GROWING TOO FAST:

- Too rapid a growth curve can result in new cash flow issues from which the studio may not be able to recover.
- Taking on new projects means that you need the staff to handle these projects. Are the hiring and training systems in place to bring new staff on in a timely manner?

LESS PERSONAL ACCOUNTABILITY:

- Feeling of "ownership" by individual members of a team can be diminished in a larger company.

subdepartments, our Technical Art group and Visual Development group, and two programming subdepartments, Tools Programming and the Graphics Engine team, are examples of a few initiatives in formalized communication that we've undertaken.

These are loosely formalized groups of individuals that have interests in specific areas. The Technical Art Group is a group of technical artists from various projects that share techniques and support each other when trying to solve difficult technical art problems. The Visual Development Group (VisDev) is a group of artists that work as a unit to explore conceptual design issues at both early and later stages

in games; the VisDev group is starting to contribute significantly to our new intellectual properties. For example, some of the group's designs are being used in our upcoming new Xbox intellectual property. While both groups are formally recognized at the company, they remain a volunteer-based initiative. As they become more successful, though, we are considering formalizing them to a greater degree.

Along with other formalized groups such as animation, production art, web team, community/live team, and promotion are two additional examples of formalized subdepartments. First is the Tools Programming group, which is responsible for tools development (in-house editors, and external products like the tools for NEVERWINTER NIGHTS) and installers, as well as database management for the company's projects. The other is the Graphics Engine team, which is responsi-

ble for maintaining existing engine technology in-house as well as planning and developing the next generation of engine technology for projects a few years away.

Be Objective

Finally, when dealing with every aspect of managing the studio, objectivity is a necessity, especially when resolving issues of competition between projects. Objectivity doesn't mean you treat everyone and every project equally — it does mean you gather as much data as you can and base your decision on that data, not on factors that could be perceived as arbitrary to people on the team or at the company. Some projects are more important than others, and if you base your decision on this fact, it should be out in the open.

Keeping compensation based on the overall success of the company has

allowed us to successfully prioritize projects while still keeping all teams motivated. We also motivate people by striving to assign them to projects they are interested in; people who choose their projects are much happier than those that don't. Everyone understands that sometimes they will have to work on some less exciting projects and tasks, but this is normally rewarded with a greater choice of future opportunities.

Airing company objectives and project priorities repeatedly and clearly defining your studio's goals and values at regular company and/or team meetings helps immeasurably in maintaining the fairness of resource allocation between projects.

Execution

In this article we've undertaken a high-level view of running multiple game projects — the real test will be in the execution of both the games and in building your company as a sustainable business. The key for success in both these goals is smart, creative, and passionate employees. At BioWare, we've been lucky to get consistently exceptional employees to work with us to achieve our company goals. 🚀

FOR MORE INFORMATION

Company Management:

Jim Collins. *Good to Great: Why Some Companies Make the Leap . . . And Others Don't*. HarperCollins, 2001.

Jim Collins and Jerry Porras. *Built to Last: Successful Habits of Visionary Companies*. HarperCollins, 1994.

Thomas J. Peters and Robert H. Waterman Jr. *In Search of Excellence: Lessons from America's Best-Run Companies*. Warner Books, 1998.

Personnel Management:

Charles A. O'Reilly III and Jeffrey Pfeffer. *Hidden Value: How Great Companies Achieve Extraordinary Results with Ordinary People*. Harvard Business School Press, 2000.

Jeffrey Pfeffer. *The Human Equation: Building Profits by Putting People First*. Harvard Business School Press, 1998.



Developing Online Console Games

PETE ISENSEE | Pete (peteis@xbox.com) is the lead developer for the Xbox Advanced Technology Group at Microsoft. His Gamertag is LightSleeper. If you see him online, remind him to get back to work.

STEVE GANEM | Steve (steve@neversoft.com) is the lead network programmer for Neversoft Entertainment. If you see him online as The Kraken, go easy on him.

All the major consoles — Playstation 2, Xbox and Gamecube — now support online gaming. PC developers have been writing online titles for years, but this is relatively uncharted territory for console developers. Until recently, only a handful of multiplayer online games had been created for consoles, and even fewer had been successful. The target audience for consoles has different expectations from the PC market. Console game designers are faced with limited input devices and console manufacturer requirements that demand a different set of design decisions. Console game developers are faced with adding network programming expertise to their list of skills. And now console producers have to schedule new online features and carefully consider network skills in hiring decisions.

This article examines the new world of multiplayer possibilities with current-generation consoles, along with the limitations online console game development presents. It contrasts the different approaches the console makers have taken, and details the design and development considerations involved in creating online console games.

Online Strategies: The Big Picture

Each console manufacturer is taking a different approach to enabling online play. Microsoft is building an integrated service that emphasizes a uniform experience across titles, while Sony and Nintendo are investing less on back-end services and allowing more flexibility across games. Each scheme has its benefits and drawbacks. Microsoft requires that customers have a broadband connection, whereas Sony and Nintendo are also supporting players with dial-up connections. Xbox game designers have fatter pipes for game traffic, at the cost of a smaller potential market. PS2 and Gamecube developers have a larger potential market, but a wider range of bandwidths to support.

The next section examines each company's approach in detail.

PS2 Strategy

Sony is the undisputed leader in terms of console market share. Sony launched its first online game in Japan in May 2002 and in North America in August 2002. Sony encourages PS2 game developers to support both dial-up and broadband users, although some games have chosen to support broadband only. For instance, *SOCOM: U.S. NAVY SEALs* is a broadband-only game. PS2 online players must purchase a separate network adapter (\$40), but there is no sign-up fee. Players can use their pre-existing dial-up or broadband connection. Some games support additional peripherals, such as a keyboard for online chat, a voice headset, or hard drive. Sony encourages its developers to provide multiplayer online gaming for free as an added feature, although developers can charge a subscription fee for games such as persistent-world games. For instance, the PS2 version of *EVERQUEST* will charge a monthly fee.

From a development standpoint, Sony provides an online API via the SCE-RT library. This library is free to licensed PS2 developers. Sony is building out facilities for hosting game servers, but they also encourage developers to build and host their own services or use third-party middleware and services. The advantage of this approach is that the game developer has maximum flexibility. They can build or buy, and they can manage their own customer base. A downside of this flexibility is the inconsistency of online capabilities and UI from title to title.

Sony is allowing considerable latitude for online developers. For example, developers can pick and choose how to implement multiplayer services like matchmaking. The possibility exists of having cross-platform compatibility with Gamecube or PC games. Buyers of online PS2 games could conceivably find a large existing community of PC players the very first

time they log on. One potential disadvantage of this open system is the lack of a global security infrastructure. As online PC game developers have discovered, security is critical to preventing cheaters from ruining the game experience. With many different network library options available, there is a greater possibility that any particular online title may have unsatisfactory security built in.

Sony has placed few requirements on online games. For instance, voice is not a mandatory feature but rather left up to the developers to decide if it's appropriate for a game. In general, the Sony strategy is to maximize the potential audience for online games and give game developers broad flexibility in choosing what features their games should support and how those features should be implemented. However, Sony is reportedly developing a more integrated service for the European market called the Network Gaming Service (NGS), with indications that it will support single player IDs and other global features for titles that use the SCE-RT library. Whether this indicates Sony is moving toward a more consolidated service philosophy, at least in Europe, remains to be seen.

Gamecube Strategy

Nintendo is taking a cautious course, with the view that online gaming is not yet a viable market. Nintendo launched its first online title with *PHANTASY STAR ONLINE EPISODE I & II* for Gamecube in October 2002. Players must purchase a separate dial-up or broadband adapter (\$35) to play online. Like Sony, Nintendo is not charging a sign-up or subscription fee. Online services must be built by the game developer or accessed via third-party middleware.

Nintendo has been characteristically tight-lipped about future online plans for Gamecube. At the time of this writing, Nintendo does not appear to be enforcing any types of policy decisions about voice communication, keyboard chat, or global identities. There are also no indications that Nintendo is committing

large resources to back-end services or other online infrastructure.

Xbox Strategy

Microsoft's online approach for Xbox is bold and hence, risky. Xbox itself was designed with online play in mind; all consoles include a built-in Ethernet port and hard disk. Microsoft launched its online service, Xbox Live, in North America in November 2002, and it has announced plans to launch in additional territories over the course of 2003.

To use Xbox Live, players must have broadband, typically via a cable or DSL modem using any ISP. Microsoft chose not to support dial-up connections, limiting their market significantly. In addition, unlike Sony and apparently Nintendo, Microsoft is charging players a fee for the service. For \$50 players get a one-year subscription, a voice headset, and two games (*MOTOGP* and *WHACKED* in North America). Games may charge additional fees if they wish, although there are no current games doing so.

Microsoft has built a suite of online services and online game APIs, some of which are optional and some of which are required. From a development standpoint, this is both a benefit and a limitation. The benefit is that standard services like matchmaking are automatically available, tested, and free to use. Developers don't need to roll their own, pay for middleware, or pay to host their own matchmaking servers. The disadvantage is that game developers may find themselves required to use services that they normally wouldn't use or that aren't flexible enough for their needs. Developers also need to budget additional time for coding and testing required services. Currently, Microsoft provides services for peer-to-peer matchmaking, buddy lists (including online presence), game content delivery, voice chat, billing, and persistent storage of player statistics. Developers may write their own game servers if they wish, a necessity for massively multiplayer games.

Unlike Sony's (and presumably Nintendo's) relatively "open" approach, Xbox Live is considered a "closed" service. Xbox Live games cannot communicate with other consoles or with PC games, nor can they access web sites or the Internet at large. Custom game servers must be hosted in secure data centers approved by Microsoft. A closed system is a disadvantage from the standpoint of the community-building issue. It's hard to build community within such a limited environment. The benefit of these restrictions is that games are heavily resistant to cheating.

Microsoft's online service also allows players to have a global identity across all Xbox Live games. When players sign up for the service, they choose a "Gamertag" that becomes their name in every Live game. Gamertags are in turn used to build buddy lists (Friends lists, in Xbox Live parlance), which are also consistent across games. In addition, all Xbox Live games must support voice communication. Microsoft is banking on voice becoming a key differentiator for Xbox Live.

Microsoft's strategy is to provide a consistent experience for players across all Xbox Live games. Microsoft is investing heavily in its online service, and it appears committed to the long-term success of online play. Microsoft is providing a wide range of built-in technologies and services to online game developers, but in turn requires developers to make an engineering commitment to many global Live features. Obviously, Microsoft is also making a calculated bet that the broadband market will grow substantially in the coming years.

Online Console Game Design Issues

It's no secret that the typical console player is different from a typical PC game player. How does this apply to game design issues for multiplayer console games? The first difference is the couch versus back-room mentality. When you ask PC game players how they most enjoy playing games, their answer usually involves something about sitting at the

computer — by themselves — and beating the snot out of players they don't know. When you ask console game players the same question, their answer will typically include something about sitting on the couch — with their friends alongside — and beating the snot out of each other. Console players tend to play with their friends, and they tend to play in a more social atmosphere.

The second difference is that the average console player is less technically knowledgeable and less forgiving than the average PC online gamer. Console game players don't care about round-trip ping time, they don't run traceroutes, they don't know an IP address from a P.O. address, and they certainly don't want to configure routers. If multiplayer gaming doesn't work for them, they will blame the game — not their modem, not the ISP, and not the Internet. This means that multiplayer gaming must appear to be an extension of single player gaming. Techniques for doing this effectively include disguising latency, avoiding unfamiliar terms like ping time, and not putting players into sessions that don't have good bandwidth characteristics.

Consoles are designed for the living room. They output to television screens, not high-resolution monitors. They typically take input from controllers, not keyboards or mice. Multiplayer console games must be designed for the living room as well. Suggestions for online console game designers include:

- Display only the most critical information when showing sessions or player information. Too much information clutters the screen and overwhelms most players. Rather than packing data on the screen, prefer a drill-down approach.
- Avoid long lists; list traversal is difficult given limited screen resolution and controller issues. If you have multiple pages of sessions or players, provide a controller shortcut for getting to the next and previous pages.
- Make it easy for players to get into game sessions. Minimize multiplayer configuration screens. Writing a game with voice communi-

cation presents additional design challenges. First of all, it probably doesn't make sense, both from a bandwidth and discernability standpoint, to allow every player to talk with every other player during the game. That means that the game needs to somehow limit players' use of voice. The most effective games limit voice in ways that make sense to players in the context of the game. Perhaps players can communicate only with their teammates, or only with people near them in the game world. Or maybe the game uses phones, radio channels, or other well-understood means of filtering voice communication.

As with online PC games, online console games must carefully budget bandwidth and understand how to effectively tolerate latency. Even with broadband-only games, broadband is not a panacea. All the lessons PC network game developers have learned about disguising latency, limiting network traffic, and handling dropped packets still apply to broadband.

Topologies

Multiplayer game topology is a key design decision that can have a huge impact not only on gameplay but also on engineering time and long-term costs. There are three general models to consider: peer-to-peer, client-server where one console acts as the server, and client-server when the server is external (Figure 1). These models are applicable to PS2, Gamecube, and Xbox.

In a peer-to-peer game, each console communicates with every other console in the game. External servers are only peripherally involved, perhaps during the match-making phase or for downloading new content. The advantage of a peer-to-peer game is that no game servers need to be designed, coded, or maintained. Peers communicate directly without server involvement, so latency is minimized. The disadvantage is that the number of peers is limited due to bandwidth constraints, particularly with dial-up connections.

In a client-server game where one console is the server, the console could be either a dedicated server or an active par-

ticipant in the game. Like the peer-to-peer model, no external servers have to be maintained. Unlike the peer-to-peer model, bandwidth becomes less of an issue, since each console communicates only with the server console. However, choosing the server console to maximize bandwidth and minimize latency becomes important, as does handling host migration when the server console shuts down or leaves the game.

A client-server game with a custom external server typically provides the ideal bandwidth characteristics, since the custom server can reside in a high-bandwidth data center. However, this topology introduces the most expense, both in terms of server development and testing, as well as long-term maintenance, server management, and data center bandwidth fees. Experienced server developers who also understand gaming issues are rare.

Another possibility is to mix and match topologies. In a massively multi-player game, for example, it may make sense to use the client-server model for gameplay, but use a peer-to-peer model for voice communication. The important thing is for developers not to confine their thinking to any one model. Evaluate all the possibilities carefully.

Development Issues

All the console makers provide socket-level interfaces for network programming. On PS2, developers can choose from a variety of network stacks, including Sony Inet, Sony libeenet, SN Systems' NDK, and Access AVE-TCP (Japan). Each library has its own unique performance characteristics and features. For instance, the NDK stack includes tools that simulate various network rates and connection types, while Inet and libeenet provide access to ISP setup data that may be used in all titles across the platform.

On Xbox, the network stack is a combination of Winsock 1.1 and additional security-related functions. The Xbox network stack automatically handles packet encryption and authentication. On PS2, developers can use SCE-RT for traffic encryption, use a middleware solution, or

provide their own security mechanisms.

On PS2, developers have many choices when it comes to online service libraries. The SCE-RT library, developed by Sony and RTime (a networking company Sony acquired) provides game server and lobby functionality, including account management, matchmaking, voice chat, buddy lists, clans, and ladders. This library is free to PS2 developers. GameSpy, one of several popular network middleware vendors, provides libraries for peer-to-peer matchmaking, statistics, security, and voice chat. GameSpy provides similar libraries and services for Gamecube. GameSpy's history with PC network solutions makes it a good choice for cross-platform development on PCs, PS2, and Gamecube. Pricing is title-dependent and is negotiated with GameSpy. On Xbox, the Xbox Software Development Kit (XDK) provides all the APIs and libraries required for accessing services like matchmaking, friends, statistics, content delivery, and voice chat. The XDK also includes tools used for Internet simulation.

Xbox provides technology developed by Microsoft Research for measuring connection bandwidth and latency. Game developers can use these functions for detecting the quality of service between various consoles and servers. On PS2, each network stack varies in its ability to provide ICMP-level functionality. For instance, SN Systems' NDK does not provide a ping interface, although the stack will respond to ICMP pings.

Each console vendor is taking a different approach to dealing with network address translators (NATs). A NAT is a device (typically a router or gateway) that allows multiple network devices to share an

IP address. NATs are prevalent in home environments where an Internet connection is shared among PCs and game consoles. Some NATs also act as firewalls, blocking suspicious packets. Such NATs can be difficult for games to negotiate, particularly peer-to-peer games. The only viable option requires a mediator informing both peers about their port mappings. On PS2, games can use Sony's SCE-RT library or middleware libraries and services like GameSpy. On Xbox, NATs are handled automatically using a combination of the network stack and the Live service.

On Xbox, the Xbox Dashboard handles all network configuration issues. If an Xbox game can't establish a connection to Xbox Live, the game simply allows the player to run the Dashboard. The Dashboard saves network settings on the Xbox hard disk. Xbox games don't need to provide custom network management. PS2 players configure the Sony PS2 network adapter for connection to their ISP using the Network Startup Disc that comes with the adapter, which then saves these settings to the memory card for use by all games. Developers of PS2 games may also choose to embed a standard network-configuration UI library provided by Sony, in case players haven't yet set up the network config-

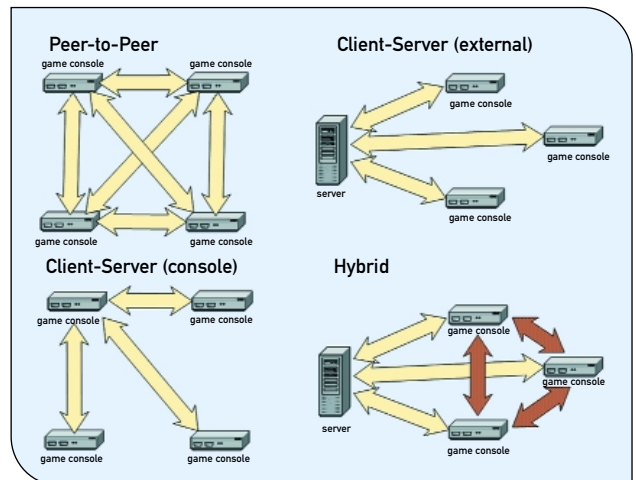


FIGURE 1. Topologies associated with online console gaming. Developers should examine all options and combinations to determine what best suits a game's design and technical requirements.

uration with the Network Startup Disc.

Because of their architecture and memory constraints, consoles have other limitations in comparison to PCs. Even though console network stacks require a comparatively small amount of memory, every byte is precious on a console. For instance, on PS2 the IOP (I/O Processor) memory can be a limiting factor, since it's used to load drivers for USB, keyboards, the network adapter, game pads, and memory cards. On Xbox, memory is

not generally an issue, since the unified 64MB can be utilized by all devices.

However, the automatic Xbox packet encryption and NAT traversal algorithms require developers to be aware of underlying performance issues they're not used to if they've done networking on other platforms.

On PS2, the first-generation network stacks required multithreaded implementations to achieve asynchronous network I/O. These first-generation stacks also con-

tained many operations that would block due to data transfers between the Emotion Engine CPU (EE) and the IOP. The latest Sony network library provides a complete BSD socket API on the EE. This library eliminates previous blocking issues, and also abstracts away the underlying multi-threaded I/O implementation.

On Xbox, the networking libraries are designed to work primarily in a single-threaded environment, although multi-threading is also supported. The Xbox

ONLINE/MULTIPLAYER CONSOLE COMPARISON

FEATURE	PS2	GAMECUBE	XBOX
Console launch date (North America)	October 2000	November 2001	November 2001
Online launch date (North America)	August 2002	October 2002	November 2002
Internet connection	Dial-up/broadband	Dial-up/broadband	Broadband only
Networking hardware	Network Adaptor (\$40); supports both dial-up & broadband	Modem Adaptor (\$35) or broadband adaptor (\$35)	Built-in
Hard disk	Add-on (currently Japan only)	None currently	Built-in
Voice headset	Included with SOCOM (\$50)	Add-on (price unknown)	Included with Starter Kit (\$50)
Online subscription, per year	Free	Free	Included with Starter Kit (\$50)
Ability for games to bill customer for subscriptions or content	Yes	Yes	Yes
Support for custom game servers	Yes	Yes	Yes
Game server hosting	Sony, publisher, or middleware provider may host	Publisher or middleware provider may host	Microsoft or publisher may host (with MS approval)
Games require voice communication	No	No	Yes
Games support voice communication	Some	Unknown	All
Keyboard option	Yes	Yes	None currently
Network configuration	Network Startup Disc, stored to memory card (and optionally in-game)	Unknown	Xbox Dashboard (never in-game)
Packet encryption/authentication	SCE-RT or middleware	Middleware	Xbox network stack
NAT traversal	SCE-RT or middleware	Middleware	Xbox network stack
Supports global identity	Coming in some games	Unknown	Yes
Buddy lists	SCE-RT or middleware	Middleware	XDK
Peer-to-peer matchmaking	SCE-RT or middleware	Middleware	XDK
Game statistics	Middleware	Middleware	XDK
Supports downloaded content	Memory card or hard disk add-on	Unknown	Hard disk
Voice libraries	SCE-RT or middleware	Middleware	XDK
Cross-platform communication possible	Yes	Yes	No
Communication with external web or other types of servers possible	Yes	Yes	No
Mod chip detection	None currently	No	Yes
Network interface	Sockets (multiple net stacks available)	Sockets	Sockets (Winsock)
Supported protocols	UDP, TCP, custom voice protocol, others	UDP, TCP, others	UDP, TCP, custom voice protocol
Supports patching	Via memory card or hard disk add-on	No	Yes; security fixes only

Live libraries use a task-pumping architecture. Each online task (searching for game sessions, for example) is “pumped” once per game loop. During this period, the task performs a minimal amount of work and then returns control to the game. Typically, the minimal amount of work simply involves checking to see if data has arrived on the network.

Testing

Perhaps the most difficult part of developing multiplayer games is testing and tuning. In the PC world, the answer to this problem seems to be to let the players test the game, then release patches, ad infinitum. The console world does not follow this pattern. Once a console game has been certified and released by the console manufacturer, it is virtually never updated. That means console multiplayer games must undergo extensive real-world testing long before they ship.

There are multiple approaches to the problem of testing multiplayer console games, and the console manufacturers still have a long way to go to improve life for developers in this area. For instance, early PS2 online game developers had to test their games on every possible USB modem, which Sony now discourages supporting in favor of the official network adapters. Early Xbox Live game developers had to build custom network-testing tools because tools provided by Microsoft were limited.

The primary problem in testing multiplayer games is emulating real-world Internet conditions. Even sophisticated tools fall short, because the Internet is such an amorphous and changing environment. For instance, the Internet doesn’t randomly drop or delay packets — drops and delays tend to clump together as router queues become clogged. Imitating this behavior in a realistic fashion is extremely difficult.

All the console manufacturers are supporting online beta programs to various degrees. Microsoft runs an Xbox online beta program with selected players on custom Xbox hardware. PS2 developers

have two options: Sony can administer the beta, drawing from a pool of network adapter owners, or the publisher itself can administer the beta with beta discs mastered by Sony. From a development perspective, the important part is planning for a beta period, and including time in the schedule to respond to issues raised by the beta.

Console manufacturers also include online testing as part of their normal game certification process. Microsoft has many additional certification requirements for Xbox Live games, mainly around specific features like friends and voice. Microsoft also has minimum bandwidth and latency requirements which games must work under without noticeable lags or stutters. Sony has no such official requirements. Games are simply expected to perform well for the number of players they officially support.

Microsoft is alone in requiring that all games support a patching mechanism called AutoUpdate. During Live sign-in, if the game detects that a newer version is available, it automatically downloads the updated version to the hard disk. The primary purpose of this feature is for fixing security flaws or scaling issues in the game. Sony also has a limited ability for games to patch after release. Currently, only one PS2 title supports patches, which are stored on the memory card.

The Future

The future for online console games looks bright. The fact that every console manufacturer is supporting online play is a good indication that it will become an important feature of new games. The ability to use the Internet affects single-player console games as well. Console games now have the opportunity to persist game scores, download new levels or rosters, inform players of tournaments and game-related news, advertise fresh content within the game, and even access real-time data like weather and sports scores.

One major frontier for online console

games is figuring out ways of building community. Some new games are already starting to do this in simple ways. For instance, *MOTOGP*, a motorcycle racing game that comes with the Xbox Live Starter Kit, shows how you rank, not just overall, but against your friends. Only a handful of players have the incentive to be the best racers in the world, but almost everybody has an incentive to beat the people they play with every day. This is just one example of using the community aspect of multiplayer gaming to enhance the game.

In the future, expect console games and services that integrate player feedback mechanisms (like *Ebay.com* customer ratings), clans, tournaments, ladders, and detailed game stats. Expect web interfaces with features like sign-up, statistics, session reservations, game ladders, and so forth. Microsoft already supports a minimal upload service for game scores and leaderboards. Future versions of Xbox Live may support content upload for things like replays, game levels, and perhaps even game mods. With the advent of voice as a viable communication mechanism (as in *SOCOM*, for example), developers have the opportunity to innovate in new ways, including command and control mechanisms, character-specific voice masks, and other clever voice tricks.

Online console games have the potential for changing gameplay in the living room. The ability to invite your friends to your virtual couch and enjoy a high-quality gaming experience is irresistible to most players. The market itself is still small — not many folks have an Internet connection next to their TV — but it’s growing quickly, and the technology for enabling online play within console games continues to improve. 🎮

FOR MORE INFORMATION

ONLINE CONSOLE MIDDLEWARE PROVIDERS

GameSpy - www.gamespy.com

SN Systems - www.snsystems.com

Access - www.access.co.jp/english

Piranha Bytes'

GOTHIC II:

The Interactive Score



If all goes according to plan, this Postmortem will be published throughout the English-speaking world roughly around the same time as Piranha Bytes' latest game, *GOTHIC II*. While the game was released in Germany in October 2002, the English version of *GOTHIC II* should hit shelves at about the same time this article reaches you, or soon after.

Like *GOTHIC I*, which topped the charts in Germany and was something of a sleeper-hit RPG in the U.S. following its 2001 release, *GOTHIC II* is a 3D action-adventure RPG set in a medieval world. In both parts of the series, the player takes the role of a nameless protagonist to plow through an epic story. Combat plays a certain role in a literal sense, but much of the acclaim the game received centered around the multitude of lifelike characters, their everyday lives and fates. *GOTHIC I* was praised for the incredibly detailed game world, and with *GOTHIC II* Piranha Bytes strived to create yet another deeply immersive game.

When I started as Piranha Bytes' "audio guy" in 1999, *GOTHIC I* was in the very early stages of development. People had already begun to evaluate several possibilities for the musical aspects, but not a single line of code for

the future music system had been written. This gave me the opportunity to participate as an external "data creator" and produce new ideas. The idea behind *GOTHIC I* (and then *II*) was to present a detailed, atmospheric, and lifelike game world, and this included the music. The team took great care to create a unified vision for the general game atmosphere and in singular locations. Pictures were drawn and stories told until every single one of us could visualize each part of the world.

On the music front, the first question we asked ourselves for was, Would it be profitable to use an interactive music system? The advantages were obvious. As we see so often in movies or on TV, music is a decisive factor in determining the intensity of a scene. Achieving this goal of emotional intensity can be reached only if the music emphasizes the shown scene and changes according to the actor's (or player's) actions. In a game, however, if you want something

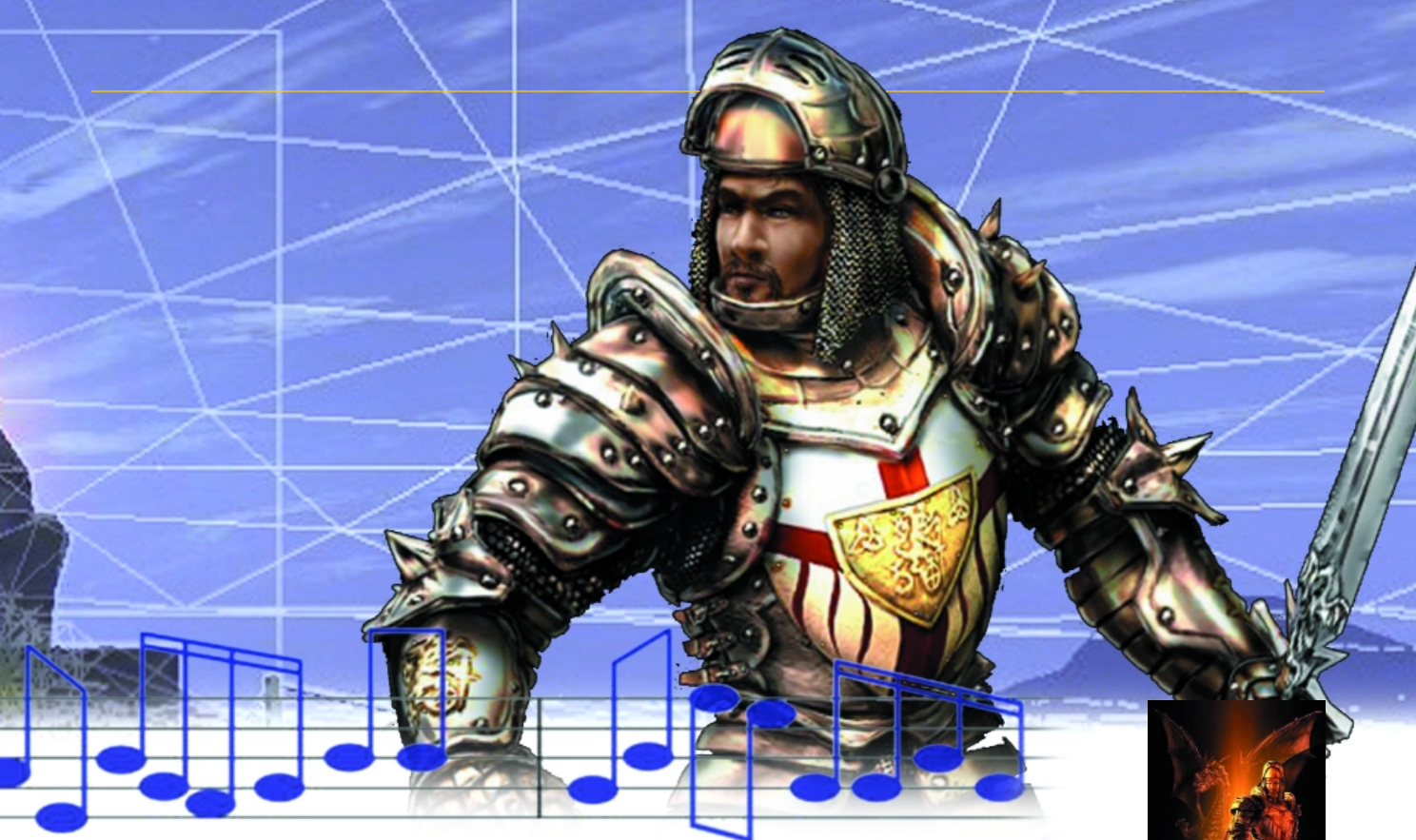
more than a string of prearranged and cross-blended MP3s, you need to get into a very complex music system. Technology has to allow for inaudible blending and good timing between different musical themes.

Such interactivity can be achieved by not using prearranged and premixed tracks, but rather loops and samples which are strung together in real time. In the case of Microsoft's DirectMusic (the system we finally decided on), each note is a sample and played exactly when the MIDI score demands it to be played.

As a musician, my goal for the *GOTHIC* soundtracks was to create an orchestral score inspired by those of *Braveheart* and *Conan* that would work as a whole but also adapt to different situations.

It's well known that many projects neglect the audio component. Existing systems are half-heartedly evaluated, so this Postmortem is meant to shed some light on things to keep in mind and common sources of errors. I had to start over

KAI ROSENKRANZ | Kai started working for Piranha Bytes while taking his final school exams, composing the music for *GOTHIC* at home as a freelancer. He threw his degree plans overboard to help the team during the final stages of the original *GOTHIC* game. He also took on creating visual and sound effects in addition to his work as a composer. Now 22, Kai was most recently in charge of the audio component (among other things) for *GOTHIC II*. He welcomes feedback and ideas at kai.rosenkranz@piranha-bytes.com.



again so many times I'd be glad if someone were spared this gruesome fate because of this article.

What Went Right

1 ● **Choosing DirectMusic.** As I mentioned, we decided to use DirectMusic (which nowadays is called DirectAudio) for *GOthic I* and *II*. Other alternatives we considered were a self-programmed opus based on MOD files, and using MP3 files as puzzle pieces controlled by a multitude of parameters. Especially the latter option might have found a lot of uses, but our decision for DirectMusic proved to be a sound one.

DirectMusic had numerous advantages. The system contains the tools to create interactive music. The music system's structure classifies the background music as different data types, which can be organized and edited via Producer, a blown-up all-around editor.

So-called DLS files (DownLoadable Sounds) contain the wave samples proper (single strings and brass notes, choir chords, percussion beat, and the like). In addition, the DLS files also contain the instruments which control the allocation of wave files to the spectrum of notes on a virtual MIDI keyboard. They assign

the triggering and envelope of samples to keystrokes.

DirectMusic's styles contain the composed musical components. Each style contains one or more bands where the DLS file instruments are arranged in parts (similar to MIDI channels). For example, MIDI channel 1 will use the instrument "bass," channel 2 "strings," and channel 3 "brass." *GOthic I* used 16 different instruments, while *GOthic II* has 32 parts. The most important components of a style are the patterns, which contain the notes proper and the corresponding controller events (similar to a MIDI file). Each style can contain any number of patterns, which represent specific musical puzzle parts.

The styles' patterns are grouped along a timeline (again, similar to a MIDI file) in segments and brought into a chronological sequence. You can also force the system to pick a random pattern from a certain pool to create some variation.

DirectMusic contains all these elements. The Producer, with its functional overkill, might look a bit scary at first glance, but it combines all the editing possibilities I needed in one tool. Just the keyboard recording caused me a few sleepless nights, but more on that later.

COMPANY DATA

PROJECT TITLE: *GOthic II*

DEVELOPER: Piranha Bytes

PUBLISHER: JoWood

NUMBER OF FULL-TIME DEVELOPERS: 13

NUMBER OF EXTERNAL STAFF AND

CONTRACTORS: 4 translators, approx. 40 testers

PROJECT LENGTH: 11 months

RELEASE DATE: Germany: October 29, 2002

USA: Q1/Q2 2003

PLATFORM: PC

AVERAGE DEVELOPMENT HARDWARE:

500–1400MHz PCs with 512–1024MB RAM,

GeForce 3s, and 40GB hard drives

DEVELOPMENT SOFTWARE USED: Visual

Studio C++, SourceSafe, 3DS Max 4, Adobe

Photoshop, Microsoft Producer, UltraEdit

NOTABLE TECHNOLOGIES: Bink,

Miles Sound System

PROJECT SIZE: 3MB game code, 7MB engine

code, 3MB tool code, 10MB script code

NUMBER OF SAMPLES USED IN-GAME: 510

NUMBER OF SAMPLES USED FOR

CUTSCENES: 1,350

MUSIC-SPECIFIC SOFTWARE USED:

Soundforge, WaveLab, Microsoft DirectMusic

Producer

NUMBER OF MUSIC ELEMENTS

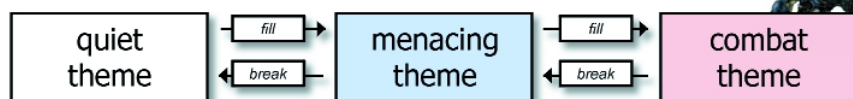
(PUZZLE PIECES): 80+ transition patterns

2. Combination of game logic and music system. The features I just described are a small part of what DirectMusic offers in terms of possibilities, but they were enough to cover everything we needed to for the internal music system for GOTHIC I and II. We wrote down a detailed requirements specification in the preliminary stages, enumerating what the system should do and what it didn't have to do.

GOTHIC II's internal music system allows me as a composer to trigger certain segments depending on the day/night cycle, location (town, dungeon, swamp, bandit's camp, and so on), and situation (quiet, menacing, combat). In order to combine this system with the game logic, I split the music system into a low-level part and a high-level part. The low-level part contains all routines for initializing the DirectMusic port and exchanging data between the engine and music system via that same port. The high-level part is constantly (every few frames) checking if the player's situation changed (depending on time, location, and game situation), and passes the result on to the low-level part. So while the high-level part is closely cooperating with game logic, deducing the "threat factor" of a given situation from coefficients such as enemy superiority and number of hit points, the low-level part just receives a composite segment name (for example, `swamp_day_combat.sgt`) and feeds the new score part to DirectMusic.

3. Harmonizing transitional system. One major challenge with interactive scores is avoiding audible transitions between different themes. DirectMusic provides the option to define the point of transition between two segments. The available options are "immediate," "grid," "beat," "measure" (or "bar"), and "end of segment," meaning that the transition will be made instantly or after the end of the next chronological unit, note, cycle, or segment. In most cases I use `Transition_SubType_Measure` because having interference at a measure break is easiest to perform.

In addition, DirectMusic offers a range



This minimalistic approach to using the music system proved quite useful.

of transition types I put to unintended use. These features usually serve a different purpose within DirectMusic, but I converted the functionality into something that would best cover my demands. For example, there is the option of assigning one of the characteristics "Intro," "End," "Break," or "Fill" to a pattern. Originally these "enhanced" patterns served as intros (before a segment started), ends (after ending a segment), to introduce a break, or a fill. In the context of the GOTHIC music system these transitions have achieved a different significance, however.

Essentially, I use the transition "End" to signify a change of location (the low-level music system can force DirectMusic to use certain modes of transition). An "End" pattern always contains a logical musical conclusion to a segment with a relatively neutral ending, so as to be able to introduce the next theme. For a change of the day/night cycle I use "Intro" transitions. The remaining transition types "Break" and "Fill" are used when a game situation changes. This minimalist approach to using the system has proven quite useful.

4. Use of orchestral sound libraries. Once we had established the technical background for dynamic in-game music, I could go on to create the actual data. I started my work as a composer and sound designer along the structure DLS file -> Style -> Segment. The first task was to create samples for an orchestral soundtrack. As with most things in life, there is an expense factor; although you can get high-quality sound sample libraries — for example, a decent strings CD — for something around \$180, it pays to buy high-quality samples if you want a more authentic-sounding orchestral sound. GOTHIC II used eight sample libraries at a cost of more than \$2,000, and those

were just the financially accomplishable parts of my wish list. The quality of orchestral sounds constitutes the main musical difference between GOTHIC I and II — the GOTHIC II budget enabled me to buy some additional libraries.

There's one important point I should mention: the published game contains the DLS files, and since the raw samples can be extracted via certain tools, I couldn't use the sample libraries' original, unadulterated files. Most libraries provide the samples to be used as part of an arrangement only; the raw materials in themselves may not be extracted and reused. So it is necessary to alter each and every sample somehow. Storage space necessities forced me to convert samples and shorten the loops anyway, which in the end turned out to be enough alteration.

5. Close cooperation with the team; no freelancing. Game audio still seems to prefer freelancers. The many presentations at last year's Game Developers Conference gave me the impression that the need for cooperation between the audio professional and the rest of the team remains severely underestimated.

The advantage of working in-house over using freelancers is significant, though. Because I work with my 12 Piranha Bytes teammates on a daily basis, I can instantly align my work with the project's desired styles and effects. Especially during the phase when we were introducing the new music system, it helped a lot to be on location. For one thing, I as the musician was able to implement most specifications myself. Furthermore, I got a detailed overview of the system, which helped to keep later bugs and errors limited to a small number, which helped to save programming time.

Perhaps most importantly for me as a composer, the team's regular feedback on style helped a lot to ensure my music's

success. Especially when creating interactive music, cooperation is a necessity; music and the rest of the game move together even more closely than in the case of linear background tracks. An autonomous MP3 track may create an atmospheric if self-contained layer of sound, but interactive music tends to cling to the ups and downs of the game just as eagerly as players themselves.

What Went Wrong

1 ● **Endless optimizing of samples, not enough composing.**

Obviously I was far too engrossed in my self-imposed exercise of optimizing each and every sample to create the solar system's most authentic and at the same time shortest strings loop. The decision to use DirectMusic was accompanied by yet another gigantic task: to create an orchestra optimized for storage use, complexity, and authenticity. After finishing this part of my job, there was hardly any time left for composing, so in the finished product several locations had to share a common theme pool.

I learned the hard way how much time it takes to create a DLS orchestra — more time than you'd ever believe you needed. All musicians who want to work with DirectAudio should be very generous with themselves when calculating their time.

2 ● **Huge memory requirements, a lot of computing time when mixing.**

The limits DirectMusic imposes on a composer should not be underestimated. Because DirectMusic is mixing the complete orchestra at run time in a separate thread, all samples needed for a segment to be played have to be in active memory. My first test orchestra might have made the London Symphony Orchestra green with envy — not because of its quality, but the sheer manpower! Unfortunately, it also used 160MB of my RAM. I had to trim it down by using shorter loops, getting rid of all samples which weren't absolutely necessary (for example, the G#6 of a transversal Irish pan flute), and other means.

To create the sound of a pompous orchestral theme in real time, DirectMusic sometimes has to mix more than 100 wave samples at one time. Of course the system uses existing sound hardware, but you still ought to make sure you're not using the whole orchestra at all times. I managed to get a working compromise by using a few tricks of the trade (creating combo samples strung together from recurring melody parts and chords, for example), but interactive music does hamper a game's inner technical workings more than MP3 playback ever would.

Fortunately, I could really whoop it up while composing the score for the cutscenes. I could be a lot more brachial

and ostentatious than with the in-game music — the interactive music system was sound asleep while the video soundtrack was playing.

3 ● **Limited musical freedom.**

Aside from the performance and memory considerations I just mentioned, other factors limited my musical freedom more than I would have liked.

While linear MP3 or CD background music might switch to a completely different style every 10 seconds or so (not that I'm not saying that should be done), in the case of DirectMusic you have to live with the limits imposed by your very own DLS orchestra.

To illustrate the difference, in *GOTHIC II* there is a Mayan temple (which has no explicable relevance to the overall plot) that sticks out in the way of style. I imagined African flutes and percussion, strange vocal patterns, and other less-orthodox instruments for this location. Unfortunately, the orchestra that I had trimmed down to 25MB memory usage didn't contain any of those.

Somebody whose job it is to deliver completely prearranged MP3s won't have a problem achieving stylistic diversity. The number and size of raw samples used don't play a role in the finished product. I, however, had to decide whether to add more samples to my already stately orchestra (using those 25MB of active



The author created musical accompaniment for the transitions between night and day.

memory just for background music) to reach the desired artistic goal, or just not reach it. In the end I decided to add those new samples, and to this day I'm hoping the programmers won't find out about it.

The DirectMusic documentation claims that it's possible to load new DLS files (samples and instruments) for certain segments during play. In theory those files should automatically be removed from memory after a segment's end, but this never worked as it should have. A possible workaround might be to use self-programmed resource files and corresponding loaders instead of the prescribed loading routines. We chose to use the default components mostly for time reasons, while a personal solution would have been better to circumvent the removal-from-memory error.

4 • Elaborate orchestral arrangements, terribly small sequencer. The Producer comes with a sequencer of its own. This sequencer isn't nearly as good as, say, Cubase or Logic, but it's been tailored to fit the other parts of the interactive music system. Those who don't want to part with their favorite sequencers can integrate premade MIDI files as patterns into styles.

Work with the integrated sequencer was, however, nothing but problematic — especially where recordings of note events via the MIDI interface were concerned. For example, I couldn't manage to get rid of a bothersome half-second delay. On top of that, melodies weren't played back during recording so I had to do "deaf recordings," being able to control what I had done only after the recording. This was the reason why I entered most notes by hand, via mouse.

Some preliminary research might have helped, but since most of my time went into the creation of the orchestra, I had no time left for experiments. Later on I discovered that the music system programmer had encountered the same half-second delay and had discovered his own workaround — so much for improved team communication.

5 • Older version of DirectX, no way to use new effects. Both GOTHIC I and II use DirectX 7. I would have loved to see a newer, more up-to-date music system for GOTHIC II, but since we had only 11 months for the game's development it just wasn't possible. This also meant I was stuck with the DirectX 7 version of DirectMusic.

In newer versions of DirectX, DirectSound and DirectMusic are combined into DirectAudio, with several profound consequences (most of them positive). Possibly the best contribution for musicians — something the new DirectAudio structure can attribute to the former DirectSound — is the option to use real-time effects with single tracks. With DirectX 7 it was possible to apply a spongy reverb effect to the finished mix, but a sophisticated blend of effects for each instrument was out of the question, and something I sorely missed. I've only had the opportunity to test DirectAudio during a brief evaluation phase, so the bulk of my experience comes from using an older version.

Looking Ahead

If you're used to a lot of latitude in the use of samples, you'll feel deprived of your artistic freedom when using a dynamic real-time music system (especially if the corresponding resource management is flawed, as was the case with both GOTHICS). If you ever tried to create an orchestral arrangement with a sampler weak on RAM, you'll know what I'm talking about. You're forced to use minimalist compositions, and each additional drum beat requires a cost-ben-



This mysterious location demanded a bunch of special samples. A cost-benefit analysis was necessary.

efit analysis. This kind of work will hamper the flow of your opus for sure; on the other hand, the system's interactivity does open up new vistas. To create music which will grow to fit even the most unpredictable of game situations and still be stylistically sound is an incredibly exciting task.

Our company's own finance manager mumbled something about using a real orchestra for our next project. Because of Piranha Bytes' financially conservative outlook, I avoided asking him about it — he might, after all, change his mind. Should things turn out, and should I ever find myself in the lucky situation of facing a real orchestra, I'd leap at the challenge. 🎵

DOWNLOADS

Download MP3 samples of GOTHIC II's music as well as sample code from the game's interactive music system from the *Game Developer* web site at www.gdmag.com.

ACKNOWLEDGEMENTS

Thanks to Nicole "Jaz" Schuhmacher for her generous translation help.

The Visible Woman: The Art of (Shameless) Self-Promotion

What does it take to succeed in the gaming industry? The answer isn't simply "make great games." The industry's explosive growth has led to a colossal wall of noise. In order to set yourself apart from that noise, some degree of self-promotion is in order. It's not enough to believe that you're good at what you do; you need to convince others as well.

Feel sick to your stomach yet? Evoking images of Machiavellian managers, delusional ass-clowns who overestimate their genius, and ambitious personalities who have no problem stepping on your spine on their way to the top, the idea of self-promotion leaves most people cold.

Don't let the stereotype blind you to the reality. Self-promotion matters, whether you're a freelancer out in the world or a staffer snug in a company cocoon. Opportunities come and go in your office, and you may not even hear about some of them if coworkers aren't aware of your capabilities. Companies themselves change: they get acquired, merge, and sometimes close their doors. Self-promotion improves your chances of emerging from such circumstances in a favorable position.

Though self-promotion might be a thorny issue for most game developers, it's especially so for women in this male-dominated industry. When it comes to self-promotion, female game developers face both internal and external barriers. Standard in-your-face marketing models employing more aggressive tactics aren't appealing to most women. So what to do? Play to your strengths. Here's how:



Make career management a priority. In the 80-hour chaos of crunch time, it's easy to forget about the big picture. Don't. Become your own agent. Take charge of your career. Watch the industry and your place in it. The best way to do that is to...

Show up. This one is so obvious and yet so easy to forget. Attend conferences, join mailing lists, and attend local meetings. Be where people can see you. Pitch a panel topic for

continued on page 71

continued from page 72

next year's Game Developers Conference. Write an article for a magazine (say, this one) about your accomplishments or ideas. Speak your mind at design meetings; advocate your own ideas. You say you're too shy? Not experienced enough? Not smart enough? Tell all those voices in your head to zip it; you have a career to cultivate. Nobody hands out "It's time to kick ass" cards — you just have to do it.

Let your gender be your asset. Taught early on to collaborate rather than compete, women are often quick to give praise where praise is due. This is good news for you. Let your female colleagues be your proponents; you can be the same for them. In this regard, your gender can be one of your greatest professional assets.

As you move through your career, make a serious effort to keep track of old friends and coworkers. If your former producer has landed a dream gig at Maxis, get in touch, congratulate her, and let her know you're out there — who knows, in six



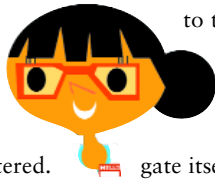
months she may be hiring for your dream project.

Plug in. Independent, strong-willed young women often disregard the good-old-girl network; the thought of a career "handout" doesn't suit them. But capitalizing on your connections doesn't lessen your talents and capabilities. If anything, it raises the stakes: if your colleague is the one that opened the door, the last thing you want to do is let your friend down.

Recruit a mentor. Find higher-ups who think highly of you. This is no time for false modesty; plenty of people think you're great. Approach one or two (or ten) of them and ask if they'd be interested in developing a mentoring relationship with you. Odds are, they'll be flattered. And, sooner rather than later, you'll be able to repay the favor.

Come prepared. This is one idea I wish I had taken seriously years ago, when I was working those 80-hour production weeks. At the end of a project, all I wanted to do was

take my archive discs and throw them down a deep hole. In retrospect, I should have set those goods under glass, because I needed them later. You will too. When a project wraps, do yourself a favor by saving box art, screen grabs, awards lists, glowing client feedback — you name it. Write up a project description, or ask for copy from the marketing department. While you're at it, ask for pithy quotes from your coworkers — quotes you can later use in your portfolio or on your web site. Burn that material on a disc and forget about it — until the day you need it. Let your work speak for you.

At the end of the day, it's your work that matters most. Cream does rise to the top. If you strive to represent yourself well both in your work and among your colleagues, your good reputation will naturally propagate itself through the connections you cultivate in the industry. 



SUSAN O'CONNOR | Susan is an interactive writer based in Austin, Tex. She has worked on over a dozen game titles. Her clients include Acclaim, Disney, ESPN, Hasbro, Nickelodeon, and Pixar. You can find her online at www.susanmary.com.