MAY 2005

gamedeveloper

T H E   L E A D I N G   G A M E   I N D U S T R Y   M A G A Z I N E

» **I ♥ THE HIVE MIND**
CREATING ILOVEBEES:
EXCLUSIVE INTERVIEW

» **JOYPADS ATTACK**
MAPPING CONTROLS
FOR SLOPPY THUMBS

» **WHAT WOULD J DO?**
MICROSOFT'S ALLARD
ON XNA, XBOX LIVE

POSTMORTEM:
INDIE POWER!
RIDING THE FBI WITH
ALIEN HOMINID

gamedeveloper

28

FEATURES

10

19

37

POSTMORTEM

DEPARTMENTS

COLUMNS

COVER ART: COURTESY OF DAN PALADIN, THE BEHEMOTH

HOMINIDAL MANIAC

EVERYONE LOVES A GOOD UNDERDOG STORY AND as it happens, this month's postmortem (page 28) provides a prime example of just that. Repeatedly receiving rapturous applause at the 2005 Independent Games Festival Awards, where it won multiple awards, The Behemoth's 2D action platform console title ALIEN HOMINID is a great example of a game whose alternative development funding and vision can result in a charming title that doesn't need to sell a million copies to break even. Is this true indie power? Well, any game in which you can jump on the head of FBI agents and ride them around like horsies must be doing something right.

WASPISH CHARM

Continuing the theme of alternatives, 4orty 2wo Entertainment's "alternate reality game" often known as Ilovebees, after the web site at the center of its mystery, is one of the most intriguing experiments of the past few years and was rightly honored with an Innovation Award at this year's Game Developers Choice Awards. It's not a conventional video game by any means, but rather a tangled maze of invented personalities, constructed web sites, and collaborative puzzle-solving, promoting Bungie's HALO 2 via mysterious audio files unlocked by answering specific real-world payphones. Our rare behind-the-scenes interview with several of Ilovebees' puppet masters (page 10) explains the whole story.

Our final major feature for this issue is a delight. Neversoft co-founder and TONY HAWK'S PRO SKATER co-creator Mick West draws on his unique knowledge of game making to discuss the thorny problem of control schemes and joypad interactivity (page 19). Say you press a controller button to make your game character jump over a chasm, but instead he plunges into the abyss (and the resulting metaphorical abyss of despondency); it's easy to become frustrated and blame the game for these failures. West's article addresses the conundrum intelligently using concepts like controller interpretation, so you can map your game controls to provide the in-game response that the player intended—not what his or her sloppy thumb mistakenly signaled.

ALLARD NO DULLARD

Say what you will about Microsoft's J Allard, he certainly knows how to talk up a storm. When we caught up with him at GDC this year (page 37), we steered clear of the obvious next-generation Xbox line of questioning, since cheeky guardedness has been Microsoft's modus operandi until the console's unveiling. But many of the less-asked questions are the more interesting ones, from Microsoft's attitude to the costly acquisition of U.K. developer Rare, to the significant first-party efforts needed to get Xbox Live off the ground, through to his company's true attitude to handheld gaming.

GAME SAY HOW MUCH?

How much should next-generation video games cost? Multiple publishers are indicating that flagship next-gen titles may cost $55 or even $60 at retailers in North America, up from the conventional $50, and this is causing anything from uncomfortable glances to significant consternation, even within the development community. And when sister web site Gamasutra.com asked, "Do you think that retail prices for next-generation games need to increase?" as its Question of the Week, the vast majority of professional game respondents were against the idea.

For next-gen titles that may cost far, far north of $10 million to create, what's the alternative? Perhaps one answer is to further develop multiple pricing tiers and carefully balance development costs based on those tiers, making only a few top-price titles, and tailoring budgets carefully to the target market and game genre. The rising price of retail games is a contentious topic all around, and we've explored it further in Heads Up Display (page 4).

On the other hand, what's another ten bucks between friends? This is particularly the case when new games that aren't smash hits (and many that are!) are swiftly reduced in price; some stores such as Fry's Electronics are also beginning to regularly reduce prices on new, otherwise full-price games. Yet, we all know we'd pay an extra $10 for the tens of hours of gameplay in the next GRAND THEFT AUTO or HALF-LIFE, and therein, my friend, lies the rub. It's not a comfortable feeling, but inflation may be finally about to hit the games biz.

S!

Simon Carless

MELTDOWN 2005
GATHERING THE SUPERHEROES OF GAME DEVELOPMENT

WOW! Use this secret code 73647 for a limited-time, super-discounted registration offer!*

JULY 26 & 27   SEATTLE, WASHINGTON

Register Now at
WWW.MICROSOFTMELTDOWN.COM

*Limited-time, super-discounted offer good until May 20, 2005.

Microsoft®
DIRECTX®

THE COST OF GAMING

WITH DEVELOPMENT COSTS EVER INCREASING AND THE ADVENT OF NEXT-GENERATION CONSOLES, many companies have begun to question whether current standards for retail prices fit a forward-thinking model of game development.

In particular, Activision CEO Robert Kotick mentioned in a March 2005 conference call that the company intends to raise wholesale game prices by as much as $10 for next-generation hardware. Should wholesale prices increase, distributors and retailers would unlikely bear the brunt of that cost, and as such, the inflation would most likely fall to the consumer, raising retail prices into the $55—$60 range for blockbuster titles. A representative from THQ also stated in a conversation with *Game Developer* that the company expects its early next-gen titles to hit just such a range.

Industry response to the issue varies, but one thing remains certain: Budgets, teams, and times, are changing. —*Brandon Sheffield*

FIRST PERSON

THE ONLY GAMES LIKELY TO INCREASE IN PRICE ARE THOSE THAT ARE premium or AAA games. These games are delivering an increasingly superior game-playing experience to the consumer, and consumers have shown that they're willing to pay for great entertainment. Ultimately, we'll see that these top-quality games continue to capture the consumers' dollar, but it's likely that lower quality games will suffer from lower sales. It's survival of the fittest, if you will. The consumer will benefit because the pressure on publishers to produce quality over quantity will mean that players will receive a more satisfying game experience.

—*Anita Frazier, Entertainment Industry Analyst, The NPD Group*

SOME GAMES WILL RISE IN PRICE, BUT NOT ALL, AT LEAST not over the long term. [But] another part of the equation that needs to be factored in is whether consumers will accept paying more for games. At some point, the backlash may be so great that there will be an overall decrease in the price of retail games.

The games industry has not figured out all the factors of the consumer buying decision. Raising the price will expose some new factors and bring to light others that were not thought to be important. The good news for all of us is that the number of gamers is increasing and the variations in their tastes are widening. Until this tapers off, whatever we do will garner some level of success. We don't have to grow up quite yet.

—*Jesse Helton, vice president of Software Development, High Voltage Software*

VIVENDI'S RADICAL PURCHASE

LOS ANGELES-BASED VIVENDI Universal Games in late March formally purchased Radical Entertainment, a development studio it has worked with in the past on titles such as THE SIMPSONS: HIT AND RUN.

VU Games will not only acquire Radical's team—the 200-plus employees will remain in their Vancouver location—but also acquire the studio's development technologies, including Pure3D, Radical's multi-genre, multi-platform game engine. Additionally, Ian Wilkinson, Radical's founder, has been named president of Radical Entertainment under VU Games.

"The purchase of Radical Entertainment reinforces our commitment to strengthening VU Games' creative talent and internal console development capabilities," said Bruce Hack, CEO of VU Games in a statement he made at the time of the acquisition. "Radical's focus will remain on entertainment-based console games for current and next-generation platforms."

The publisher, whose recently released titles include Blizzard's WORLD OF WARCRAFT and Valve's HALF-LIFE 2, reported an 17 percent loss in revenue in 2004 compared to 2003, though the company at large (Vivendi Universal) reported a 5 percent increase.

Financial details of the purchase were undisclosed as of press time. —*Jill Duffy*

PSP LAUNCHES IN NORTH AMERICA

WHEN SONY'S NEW PLAYSTATION PORTABLE handheld gaming system debuted in North America on March 24, 1 million units were shipped to retailers across the continent. Overall, the launch was extremely successful, though not immediately a complete sell-out. Sony has confirmed that over 600,000 PSP units were sold in the first week, around 100,000 units ahead of the Nintendo DS's first week numbers, and it also enjoyed solid sales afterward.

The PSP's $250 bundled package comes with a soft carrying case, wrist strap, screen cleaning cloth, battery, sampler disc, and 32MB Memory Stick Duo Pro; unlike Sony's PSP launch in Japan, the North American one doesn't give consumers the option of buying a bare-bones game system without all the accessories, although Sony may vary the bundles in the future.

Additionally, the first batch of units shipped in the U.S. arrived with a free copy of the film *Spider-Man 2* on Sony's proprietary UMD format.

The PSP launched with 17 games available of which seven were first-party titles from Sony: APE ESCAPE: ON THE LOOSE, GRETZKY NHL, NBA, TWISTED METAL: HEAD-ON, UNTOLD LEGENDS: BROTHERHOOD OF THE BLADE, WIPEOUT PURE, and WORLD TOUR SOCCER. Other games that debuted in time for the North American launch were Ubisoft's LUMINES, Konami's METAL GEAR AC!D, Namco's RIDGE RACERS, Capcom's DARKSTALKERS CHRONICLE: THE CHAOS TOWER, Koei's DYNASTY WARRIORS, Electronic Arts' NFL STREET 2: UNLEASHED, NEED FOR SPEED UNDERGROUND RIVALS and TIGER WOODS PGA TOUR, and Activision's TONY HAWK UNDERGROUND 2 REMIX and SPIDER-MAN 2.

—Simon Carless & Nich Maragos

EURO DEVELOPERS CALL FOR GOVERNMENT SUPPORT

THE EUROPEAN GAME Developers Federation (EGDF), a group whose members include developers in the United Kingdom, France, Germany, Denmark, and The Netherlands, recently put out a call to its members' governments asking for financial support for regional game development.

The group maintains that European-centric, and indeed country-specific game content has long been popular in specific localities, but increased industry consolidation, higher development costs and competition from elsewhere in the world is making it more difficult for the developers to continue making culturally unique video games.

"If we want to maintain cultural specificity in video games in Europe we need (government) intervention of the sort applying to other creative audio visual sectors," said EGDF board member and TIGA CEO Fred Hasson.

In the U.S., individual states, such as Louisiana and Hawaii offer tax incentives in order to entice potential game developers to open studios in their regions, following successful feature film initiatives in the same regions. The EGDF is presumably hoping that its members' governments will make similar moves.

—Brandon Sheffield

ANARK GAMEFACE

Build great UIs in half the time

Anark Gameface, the first professional solution for creating 3D game user interfaces and front ends. Cut your development time in half by building a bullet proof UI pipeline.

Who wins?

- **Programmers**: Code the pipeline once and never touch the UI again.
- **Artist**: Use a professional authoring tool with real workflow.
- **Producers**: Save money and reduce schedule risk.

Anark Gameface Bundle Includes

- **Anark Studio**: Proven interactive 3D authoring tool
- **Anark Format SDK**: All the tools you need to import Anark UI projects into your existing game engine.

For more information go to www.anark.com/gameface.

[SKUNK WORKS]

3D STUDIO MAX 7

BY MICHAEL DEAN

3D STUDIO MAX HAS UNDERGONE MANY iterations with surprisingly few hiccups along its evolutionary path. Max's strength is still based on its very solid core, a well fleshed-out feature set, reasonable learning curve, good community support, and its firm entrenchment in development studios all over the world.

Autodesk (formerly Discreet) has routinely chosen the path of least resistance when it comes to releasing new versions. Max 7 is no different. The release newly adds or improves some excellent features while fully integrating others that were previously unavailable or available only as a separately purchased plug-in. But at its base level, Max is still the same.

This is not to say that the most recent upgrade isn't worth every penny. There are excellent workflow improvements to complement the bullet-point sales pitch features. A new paint-selection tool is an excellent addition to the pre-existing selection tools, especially when working with dense geometry. For level designers and artists, a new "walk-through" camera movement has been added, which allows the creator to navigate an environment in a similar manner as he or she would move in a game engine. There are numerous UI enhancements and improvements,
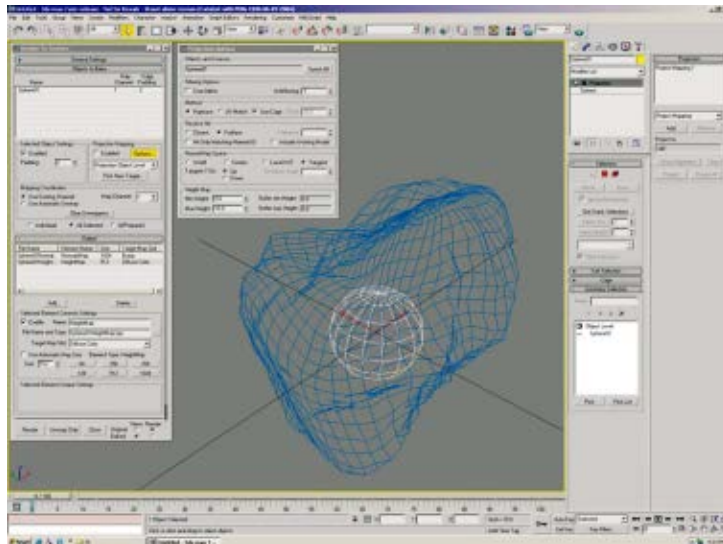
Project mapping coordinates and other vertices channel information between any two objects in Max 7.

including a dockable or floating snaps toolbar, more intuitive pop-up dialogue management, and additional options that make many tools more customizable and potentially simpler to use.

The feature set has (of course) been expanded upon and improved. While most of the expansion is based upon existing Max functionality, there are great new additions that improve workflow and overall ease-of-use.

CHARACTER STUDIO

It's not a new tool, but the latest version of Character Studio is now consolidated into Max's base toolset, though any character animator accustomed to working in Max has likely always used Character Studio. With its non-linear animation feature recently added, it's a fairly robust animation system, which still allows new animators to turn out decent (if somewhat canned) results with minimal effort.

However, many artists feel that Character Studio is too limiting, so they often opt for custom rigging and skinning systems. Max's basic toolset has been a bit tedious to use when creating custom rigs in the past; but with the addition of new tools, such as Mirror Bones, some efficiency has been added, and the Biped/Physique one-two punch is finally getting some competition from Skin and Bones.

NORMAL MAPS, EDIT POLY, AND UVW

The ability to add a perceived high level of detail to very low detail geometry is useful to almost every aspect of game art, from real-time environment and character pieces to full-motion video. In 3DS Max 7, the generation and final rendering of normal maps is fully incorporated. The process may seem a bit roundabout for the uninitiated, but it proves to be very efficient and fairly straightforward after you use it once or twice. You can view and edit scenes and characters in real time with the DirectX 9 graphics mode selected, and it visually reacts just as the user would expect, even with a custom lighting setup. The normal maps are also renderable for production output with Max's default scan-line renderer and Mental Ray.

The aging Editable Mesh object and Edit Mesh modifier are becoming less and less a part of the modeling workflow, with the introduction of Edit Poly, which has simplified the use of the more sophisticated Editable Poly object. Many tools that were formerly available just on Mesh objects are now available on Poly objects as well. Additionally, there are several unique features available to Editable Poly models that were not integrated into Editable Mesh objects. Of particular note is the inclusion of the Paint Deformation subset of the Edit Poly modifier, which brings Autodesk's own version of Maya Artisan or Zbrush-style of editing to Max. Even without this, just being able to jump into a Poly model anywhere within the modifier

3D STUDIO MAX 7

STATS
Autodesk/Discreet
10 Duke St.
Montreal, Quebec
H3C 2L7 Canada
514-393-1616
www.discreet.com

PRICE
$3,495 USD

MINIMUM SYSTEM REQUIREMENTS
Intel PIII 500MHz or AMD. Dual Intel Xeon or dual AMD Athlon or Opteron. 512MB RAM and 500MB swap space. Graphics card supporting 1,024x768 16-bit color with 64MB RAM. Windows-compliant pointing device (specific optimization for Microsoft Intellimouse). Wacom Intuos or similar pressure sensitive tablet recommended for vertex paint. CD-ROM drive, Internet Explorer 6, DirectX 9.0c, OpenGL.

Operating Systems. XP Professional (SP1), Windows 2000 (SP4).

PROS
1. Great community support and widespread studio integration.
2. Huge suite of tools.
3. Max does it all.

CONS
1. Interface becoming more difficult to navigate.
2. Many older tools are no longer the best choices for the production pipeline.
3. Max does all things well, but its competitors do some things better.

stack, select loops and rings, turn faces, and select open edges is a fabulous addition. With the improvements to the functionality of Poly objects, it's doubtful that many artists will opt for the Editable Mesh over the Editable Poly at any point of the modeling, texturing, or animating process. Modelers in particular will find this a great addition to their workflow.

Max 7 adds some UVW Mapping enhancements which pump up the efficiency of the already-solid modifier. Seams are now clearly indicated in both the modifier dialog and the scene window so that it's simpler to match up textures with corresponding geometry. Selection painting and edge loop selections are also part of the standard tools within the modifier and facilitate complex UV modifications.

Animators will find the new Parameter Collector an indispensable tool. It allows rigs with specific functionality to be grouped together into one dialog. For example, a character animator is now able to animate the bones of a hand into position and drive that positioning by telling the Parameter Collector that the bones rotated in such a manner are the equivalent of a fist, with the opposite end of the scale being an open hand. The animator is able to assign these contrasting positions as values within a Collector dialog, and can easily control it via a slider or spinner. Multiple controllers can be manipulated at once, allowing unlimited variations with less tedium and repetition. The fact that you don't need to

know how to script to use and keep the tool simple and organized is a huge plus.

FEAR OF AGING

As with any software in version 7 of its evolution, Max is beginning to show its age. Some might argue that Max is aging like a fine wine; although, some of the tools have by now turned to vinegar.

Many of Max's tools are becoming obsolete, yet are still somewhat intrusive. Menus are over laden with tools that, while useful, are having a hard time staying easily accessible. Autodesk might consider eliminating this problem by stripping down some of the tools and menus that are now outmoded by newer, better options. Rollouts are becoming so heavy with buttons, spinners, and toggles that one can't help but feel there needs to be more consistency between the different components so that these interface elements can be better grouped into a standard, customizable toolbar that's simpler and quicker to access. For example, Character Studio was amazing when it first arrived on the scene, but is now being used more out of a need to conform to how things have been done in Max for the past several years than as the best choice for character animation.

Luckily, the think-tanks behind Max seem to be slowly addressing these types of issues by systematically improving features with the best potential for growth, while allowing systems that are becoming obsolete to fade into the shadows. Hopefully, the next step will be a complete elimination of these systems, more improvements and functionality to the tools that are best suited to positive evolution, and a streamlining of Max as a whole to concentrate on its strongest aspects.

Even in its current incarnation, with such a rich feature set, brand name familiarity, and conservative revision history, it's easy to see how 3DS Max has remained at the top of the heap for so long. Of course, it has always been the Jack-of-all-trades, master of few, but it handles almost any task that an artist can throw at it with relative ease.

MICHAEL DEAN *is a freelance artist based in Austin. He is currently working as an artist with Microsoft. Email him at* **mdean@gdmag.com**.
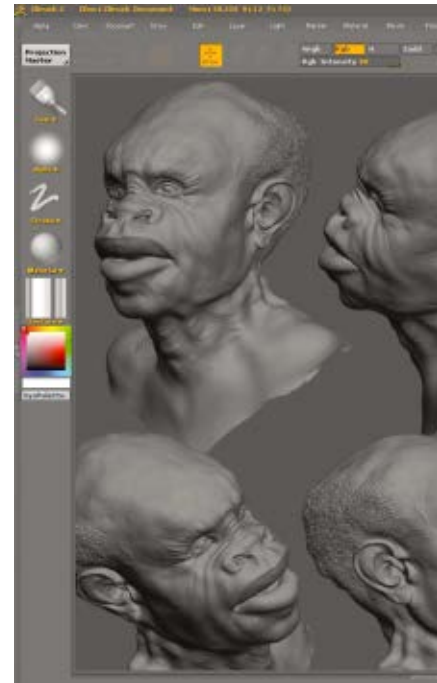
ZBrush 2's interface is as slick as its normal mapping capabilities.

ZBRUSH 2
By Tom Carroll

There once was a wonderfully succinct slogan from a not so little company called BASF that ran something like this: "We don't make the [product], we just make [it] better."

This clever little catch phrase relates equally well to Pixologic's powerful and innovative ZBrush 2 (Z2) software package. Pixologic certainly didn't invent polygonal modeling, 2D texture maps, or 3D animation, but (with the possible exception of the animation part) it sure does make them better—and depending on your application, a lot better.

For instance, if you're a developer who wants to create relatively inexpensive normal maps for next-gen console games, then pay attention. Normal mapping is where it's at; even the producer of FULL CONTACT TETRIS wants it and his game is still just a bunch of blocks. Z2 can spiff up a low-res model faster and easier than any other package, bar none.

A PIXOL BY ANY OTHER NAME

ZBrush debuted in 1999 and in a few years found favor with a core group of 3D frontrunners and converted a significant number of 2D fence-sitters. The former recognized how much farther they could

---

ZBRUSH 2

🖌 🖌 🖌 🖌 🖌

STATS
Pixologic Inc.
320 West 31st Street
Los Angeles, CA 90007
1-888-748-5967
www.ZBrush.com

PRICE
$489

RECOMMENDED SYSTEM REQUIREMENTS
Windows 2000 (min: 98). Fast PIII with optional multithreading or hyperthreading capabilities (min: PII with MMX, 200MHz). 512MB (min: 256MB). 1,280x1,024 monitor resolution (32 bits) (min: 1,024x768 (32 bits)).

PROS
1. Normal maps of complex objects in seconds or less with one click.
2. Zillions of handy tools for making or adding resolution to simple meshes.
3. Flawless transitions between various subdivision levels. Eat your heart out, Max and Maya.

CONS
1. No on-screen warning when trying to create a normal map from the highest subdivision level (if you've been clicking the button to the resonant sound of one hand clapping, you can truly appreciate the value of a little nudge).
2. What's with that boxy lasso tool? Could anything be less user-friendly?
3. Must always flip normal maps in Photoshop (or in ZBrush with an option in the Preference palette) prior to applying them back onto simplified geometry.

---

allowable adaptive skin density.

Multi-resolution Mesh Edition. Whenever a mesh is divided (quadrupled, that is) Z2 retains the previous subdivision levels. The dramatic difference is that at any time the artist can return to a simpler level to make changes. Upon returning to the higher level, simple changes are retained and are now part of the more complex mesh.

Visibility Controls. As with other modeling packages, it's critical for the user to view objects she wants to work on and exclude others. This is even more vital in the high poly world of ZBrush. New Ctrl+Shift+Click and Ctrl+Shift+Drag functions make this much, much more intuitive.

Displacement and Normal Maps. Most game developers know that both displacement and normal maps are critical to the success of next-gen game projects. Z2 works well with both. For displacement maps, the user simply needs to import a simple mesh, make changes at a higher subdivision level, import the original mesh to become the lowest level, and click on the Dispmap button. (You're going to see more of this "get a powerful result from simply clicking a button" thing all over the place in Z2.)

THE NORMAL CONQUEST
Normal maps are the *sine qua non* for next-gen game developers because they allow the real-life details that game players expect from today's games to be transferred to relatively simple meshes. A low-poly mesh can be made to look amazingly complex by simply applying a normal map to it that contains data from a high-poly equivalent containing fully-modeled 3D items, such as screw heads, rivets, wrinkles, fur … the list goes on.

Z2 is the perfect tool for making simple surfaces more pleasingly complex; and extracting normal map information is just a button click away. Check out the sidebar for the rundown on how to do it, but here are a few extra tips that will promote good Z2 use.

First, regardless of what diffuse map you're using, be sure to completely UV your model prior to importing it into Z2. The work you do in your modeler transfers with 100 percent accuracy. Hint: Use the procedural checkerboard texture if you need to. While Z2 provides ample opportunity to adjust UV maps at any subdivision level, it just makes sense to

check out your mesh's UVs from all angles, then take the finished result into Z2.

The second tip is to create the low poly in-game model, but before taking it to Z2, add small bits of definition in areas where the geometry will need to hold up when it's up-res-ed in Z2. A low-poly edge here or there can mean a lot when the model's poly count is expanded, possibly exponentially. Once the normal map has been created, delete these edges on the original model (or reload an earlier version saved before edges were added), then apply normal maps to it. With much less work, the simple model will now inherit the complex properties of the juiced-up one.

Finally, it saves time and trouble to break up complex meshes into individual sections prior to dividing them in Z2. Do this in your individual 3D package, but cap the end(s) of each piece inside Z2. This enables you to preserve (and control) the hard edge, or crease, that you need to be sure the normal maps seam up again once the finished model is reconstructed and the maps applied. ✖

TOM CARROLL *is a 3D environment artist with Rockstar San Diego. Contact him at **tcarroll@gdmag.com**.*

Bobby Milly is thanked for his input on this review.

take their existing 3D work. The latter found they could use its intuitive functionality to make 2D images with a 3D look and feel.

ZBrush fanatics jumped on this band-wagon because of something Pixologic pioneered: the Pixol, a deep pixel imaging technique whereby the creator paints in three dimensions onto the canvas. And painting with pixols can be subtractive as well as additive. You can add primitives to the mix, blend them into the existing depthscape, or create unique new works with them.

What was a unique technique then is still rather intoxicating today, only under Z2, it's even better. Here's a quick overview of new features:

New ZSphere Functions. Remember chemistry class and all those magnetic balls the teacher used to simulate valence bonding between atoms? Well, ZSpheres are kind of like that. They're the building blocks ZBrush provides for modelers who want to make a quick, efficient mesh without having to start off with single shape primitive. (Raise your hand if you can model Vin Diesel's head starting with nothing but a cube.) Among other things, Z2 improves the ease by which ZSpheres can be positioned, helps the user to select them more easily and determine what relationship one has with another, provides a front and back direction for each sphere that's easy to discern, and doubles the

How to make Z2 Simple Surfaces and Normal Maps

1. Create a low poly shape of any kind in the 3D package of your choice (the only caveat being that you must be able to export in either the .obj or .dxf file formats). In deference to FULL CONTACT TETRIS (and because we dissed the lowly cube earlier in this review), let's make it a cube.

2. Export the cube as an .obj file (still arguably the most common translation format in the industry).

3. Import it into Z2 (Tool/Import).

4. Click on the Divide button (Tool/Geometry/Divide). Click on it several times to create several subdivision levels.

5. Watch your cube become a pleasingly rounded, high-poly construct.

6. Now, use any and all of Z2's tools to modify the cube's shape, adding (or subtracting) detail.

7. Return to the lowest level of detail (Z2 won't create a normal map if you stay in the highest subdivision level, but returning to the lowest level provides the most drama, and video games are all about the drama).

8. Click on Create Normal Map.

9. At the left side of the screen, a small window will open up to display the normal map. Right click on it to export it to a directory of your choosing.

Voila! It's that easy, and the actual creation of the normal map literally takes no more than a couple of seconds. Like any complex software package, however, you get out of it what you put into it.

about me

honey

The Covenant transmission?

HONEY

At this time, I am unable to sell my honey because of an infestation of varroa

grope:
!probe extern

My honey is mixed with... the same. All that processing destroys some of the subtle floral flavors of real honey

Honey has healing... too... used it on wounds before the germs can...

truth

of

my

most

And I,

comprehensive

oriented

vision,

I

am

now

Margaret's Honey
napa valley, ca

>> alex handy

INSIDE THE HONEYPOT

THE MAKING OF HALO 2'S ILOVEBEES
VIRAL MARKETING PHENOMENON

THE INTRIGUE THAT SPIRALED OUT OF THE WEB site ilovebees.com, starting on July 16, 2004 and continuing through December, refers to a game for sure, but a game that defies easy explanation and categorization.

Ilovebees was a one-time only game experience. It started when a select few notable alternate reality gamers received jars of honey in the mail. Suspended inside the honey were bits of paper, pieces of an anagram that formed a clue when assembled properly. Other players heard about the game through an ilovebees.com link on HALO 2's preview trailer (Ilovebees was also a HALO 2 promotional tool). At its heart, Ilovebees is an unfolding story, and those who choose to participate in the game must figure out what that

story is. Banding together on forums and other online avenues, the participants collectively solved riddles and pieced together clues tied to the storyline. Players had to put their heads together especially when certain clues pointed them to physical GPS coordinates, for example, to take an incoming phone call on a payphone. Fielding and wielding clues gathered from phone calls, html files, JPG images, and so forth, individuals reported what they discovered to the other players on the forums who then collectively solved the next riddle.

The newly formed, independent company behind this creation, 4orty 2wo Entertainment, is equally ethereal. Founded by BattleTech series creator Jordan Weisman—whose FASA Studio was acquired by Microsoft in 1999, and who also

ALEX HANDY, *formerly of* Game Developer *magazine, is a freelance writer, and he's certain that there is a clue around here somewhere.html. You can contact him at* **alhandy@gdmag.com**.

From left to right: Bob Fagan, Jane McGonigal, Susan Bonds, Elan Lee, and Sean Stewart.

founded popular HeroClix tabletop game maker WizKids—4orty 2wo Entertainment builds alternate reality games as marketing tools.

Its employees claim many disciplines and work far from the head office, although a number of them were previously cloistered inside Microsoft when working on both ilovebees and a previous entertainment experience, the alternate reality game promoting the Spielberg movie *A.I.,* sometimes known as Cloudmakers.

Game Developer spoke with the company's bee-keepers—Jordan Weisman, Joe DiNunzio, Susan Bonds, Elan Lee, Jane McGonigal, Jim Stewartson, Bob Fagan, and Sean Stewart—when they were in San Francisco in March to receive a Game Developers Choice Award for innovation. In the following interview, the 4orty 2wo Entertainment crew publicly discuss their game phenomena in detail for the first time.

Alex Handy: What is Ilovebees.com?

(Entire staff becomes quiet.)

Jordan Weisman: Ilovebees is a five-and-a-half hour radio drama broken into one-minute increments and transmitted to payphones around the world, which have to be answered at the proper time and proper location by fans who have figured out the clues in advance. [This drama was then] re-assembled on the Internet.

Sean Stewart: It's *Short Cuts* meets *War of the Worlds*, a braided anthology that tells the story of six characters.

AH: Was this designed as a game or marketing tool first?

Sean: When we're doing a marketing campaign, if there is a world—like when we did the Cloudmaker thing for the film, *A.I.*—our job is to get people invested in that world, to make them interested in the kinds of stories that happen in that place. One of the tag lines is to try to make that place part of the cultural imagination of the audience.

Our job is to tell a fascinating story that makes people say, "I wonder what it would be like to be there, to be in the world of HALO, to be in the world of *A.I.*" That's our primary mission, so in that sense, the line between marketing and game and story goes away. What we do is let you walk through the back of the wardrobe into HALO 2.

AH: Have you discovered the Apple factor, the way to truly achieve irrational dedication?

Susan Bonds: It's good that you recognize that Apple people are irrational. (Laughter.)

Jordan: A lot of this goes back to the very first project we worked on together, which we refer to as the five big experiments, that we performed as part of the creation of the *Artificial Intelligence* campaign several years ago. The goal of those five experiments was to find a new model for marketing where the marketing was really quality content, and that quality content created an emotional and cognitive bond to what the marketing was being purposed of. In many ways, this was an extension of what we saw happening on the Internet. The Internet was creating a kind of try-before-you-buy mentality.

People expect to receive some value in advance of having to hand over their cash. And so this whole concept was about how to create that emotional kind of attachment and demonstrate the value and quality of that experience before you buy the video game or a ticket to the movie.

Sean: When you say a word like "marketing," what it means to most people, or at least what it means to me, is tricking people into buying some piece of crap that they don't want. So, let's reverse that equation and say, "I, the consumer, have a remote control. I can skip your commercial, I can flip past your ad in the magazine, I don't want to be tricked into buying crap I don't need. My time, my attention is valuable. If you want me to spend my attention on having anything to do with you, gimme a show."

So, we've embarked on the really bad bad bad idea of rewarding people a lot for paying attention to your thing. Instead of just saying, "Bounce makes my clothes feel clean and fresh," you actually have to really give them something for their time and attention.

Once people have worked hard enough, they have to convince themselves it was for some reason. And our players work really hard. And they are allowed to. The people who do the stuff on the web … it's just not a passive medium. Really, even a hypertext novel is just a choose-your-own-adventure game; you don't get to touch it and make it do things. On the web, you want to find a kind of storytelling where not only do the players do what they want to do on the web, they also run around and find things and then gossip about them afterwards. That's the web. Exploration made the story possible. Without the user, it doesn't happen. If no one cracks the code or answers the phone [in Ilovebees], it doesn't happen.

AH: How do you plan something like Ilovebees?

Jordan: It starts with what we've dubbed the deconstructed narrative. In this case, we're telling a story we write. Sean writes a really wonderful story filled with great characters because ultimately, that's where the emotional attachment comes from—characters you really care about.

Then, in essence, we often don't tell that story. We take that story apart. We analyze and create the evidence that would have taken place had that story happened. We bury all the evidence in organic puzzles. As the audience discovers each of those pieces and [shares] them with the larger community, the community reconstructs the story. You start with this huge spectrum of what could have happened, and everyone debates. Over time as the community gets more and more elements, that debate narrows and narrows and narrows, and eventually the audience comes to a conclusion, a consensus of what the story was. And they've recreated the story, but now its a very

Genemation
The Face Creators

GenHead v2.0
Convert 2D images to 3D
animated heads

GenCrowd v1.0
Generate thousands of synthetic
photo-realistic animated 3D
heads by age and ethnic group

Contact
Incubator Building
Grafton Street
Manchester
M13 9XX

+44 (0) 161 275 5139

john@genemation.com
www.genemation.com

Jane McGonigal busily adds labels to bottles of honey.

direction happens, when one of the players actually affects the narrative of the campaign, which is something that happened a lot.

Sean: Part of the reason for game loyalty is that in exchange for making the players jump through all these hoops, you let them drive the car. They get to affect things, and you have to be responsive to them affecting things. This is going to sound very kitsch, but part of that fanatical loyalty you get from players, on some level, reflects the fact that on some level we try to be fanatically loyal to them. We try to have a samurai commitment to their experience. We're going to let them push back and we're going to let them tell the story, and we're going to honor what they do.

AH: Does that mean you're on-call 24 hours a day? There are people in the game industry working 70 or 80 hours a week ...

Jordan: A lot of us have considered slowing down and getting into the video game business. (Laughter.)

Jane McGonigal: I do think that the way the story is being told, in an interactive way, gives you opportunities and also challenges. It requires you to constantly be ahead of the wave, but it also gives you the opportunity to respond. If you get some dead ends, you can immediately go down a different road, so I think you have a lot of opportunities to react in real time and keep it very effective.

Jordan: Another concept we have is what we call the hive mind. Our theory was, if we posed the question in the right way, and inspired a group of people—a small group initially—to try to find the answers, they would organically start to enlist a larger and larger group. We carefully tried not to introduce a competitive nature, but a collaborative one—and that group would become larger and larger and more and more collaborative, and as that group grew, it would come to quickly represent every knowledge base, and every skill base you could imagine, plus, a virtually unlimited amount of time, energy, and resources. And we were right, but we were off by one order of magnitude. We had around three million people on the *A.I.* project, and about three million people on ilovebees as well ... Damn, there's nothing they can't solve, there is nothing they can't solve instantly.

Sometimes we intentionally tried to slow them down, to give them things that would be very difficult to do. I could go on and on with stories about the amazing things the audience accomplished. And part of that is the amount of infrastructure they built to facilitate that community.

personal thing because they've built it piece by piece. And so it's not just a passive experience at this point. The deconstructed narrative is really important.

Elan Lee: It ties into your earlier question of "have we discovered this new way of marketing?" The way we look at it, there are two ways to tell a story. There's all the very traditional ways—you put it on the screen, you put it in a book. Or like Jordan said, you hide it and then sort of thumb your nose at everyone and say, "I know something you don't know!" That excites and encourages and empowers an audience in a way that traditional advertising and traditional storytelling is not able to, especially if you're willing to put in all the backend, if you're willing to reward players for digging, make it so that every once in a while their shovels hit a gem. It's a really rewarding experience, one that encourages them to continue digging and start developing those fanatical behaviors.

AH: How about the trans-media nature of Ilovebees?

Sean: The idea that the fiction is trans-platform, that's what we're getting at. The story can spill. ... Normally, you go to a theater to see a movie, and your phone rings. It's gonna be something in the real world. In this case, we're trying to spill out and say, "This story exists in the world you exist in." Your world comprises what you see on TV, who you talk to on the phone, what you look at on the computer, the people you meet in the street ...

Jordan: ... the email you get, the faxes you get, the phone calls you get, the phone calls you make, what you see on the billboards, what you see in the want ads in the newspapers, live events in clubs—the campaigns we do have used all of those techniques to glue them all together.

Sean: Trying to jump the story off the platform.

Jordan: It's the ultimate breaking of the fourth wall.

Jim Stewartson: That's especially reinforced when the other

AH: It sounds like you were conducting sociological experiments instead of marketing campaigns.

Jordan: All games are about socialization, from chess and checkers to EverQuest and everything in between. They're all ultimately about social interaction. We knew we were playing on a very broad social scale, so it wasn't so much game theory as social theory that had to go into it. What is this community? How do we encourage the community in the right way? How do you guide the community? I remember right when we launched the first *A.I.* campaign there was some guy who was pretending to be us, and we were three days into the campaign. That guy pretending to be us went online and said, "You guys are all cheating. You're collaborating, and you're not supposed to. Everybody's supposed to solve it individually." What do we do? Do we go online and say that it's not us, that's not true, or do we

just let the community sort it out? And we decided to step back and not do anything, and the community debunked the guy.

Elan: It was a hard choice to make.

Jordan: Oh, it was! We were scared to death. But they debunked him, what—within 24 hours?

AH: So how does this all change the traditional concept of storytelling?

Jordan: I remember my very first phone call to Sean. I asked him, "Sean, do you know what a role-playing game is?" because in essence, that's what we do. We're the game masters and the three million people are our players. In a classic role-playing game, you are their friends and their antagonists. You are the entire world, from the player's perspective. Not only are we the bad guys, but we're also the characters they love. We're the girls they want to date, the guys they want to meet. It's the role of game master to player base.

AH: Tell us about accomplishing the technical aspects behind all these fake servers, web sites, and such.

Jim: It was an amazing learning experience. Elan and Sean were kind enough to let me in on the early brainstorming, and my instincts were just 100 percent incorrect every single time. What was really great was just going along through the process and seeing how these things actually work, because they'd actually done it before. By the end, I think I had a pretty good sense for what to do and could contribute substantially to the design.

Sean: You should probably get him to talk about the games he creates on his own: server games, where he needs to come up with ways to thwart potential hackers and misdirect them and make them think certain systems are running when actually they are not, and that the servers are located in places they are not. There's quite a bit of art that goes into that.

Jim: Oh boy, yes. One of the most nerve-racking things about this experience for me was being on IRC and lurking and watching hundreds and hundreds of people talk about how to break our web site. One of the loveliest things about that was to see the other people in the community say, "No, don't do that. You're crossing a line. Go as far as you can using normal and legal activities, but don't break it. That's just not part of the rules." It worked remarkably well. They policed themselves.

Jordan: Don't underestimate the technical sophistication and the appreciation the audience had for the technical sophistication of what you accomplished. After we did A.I., there were a bunch of knock-off campaigns, and they totally underestimated the capabilities of the hive mind. So the technology was literally just ripped apart in minutes. Jim's task is to make the stuff really strong and really clever, and the audience respects that and frankly can't break it.

AH: What was unique about the theme in Ilovebees centering around the launch of Halo 2?

Elan: Ilovebees is particularly interesting because you had these two very different groups of people: the ARG (alternate reality games) community and a whole bunch of rabid Halo 2 fans. They were very different communities. We got all kinds of videos of these very different people [while they played Ilovebees]—older

A group of Ilovebees players assembles near the San Francisco phone booth where they successfully answered a call related to the game.

ARG guys and younger Halo 2 guys—and they were working well together and became friends. It was really cool to see.

Susan: Honestly, there was some concern on Microsoft Game Studios' side about how we're making people "get up out from the safety of their home and their computer and go to a public street and answer a pay phone! And oh my gosh! There's going to be people racing and competing with each other."

Sean: Understand, we wouldn't have designed it this way for Grand Theft Auto. (Laughter.)

Jordan: I think it's also worth noting that we couldn't have created this without an enormous amount of faith on the part of Chris Di Cesare and the Microsoft marketing group, and on the part of Bungie. Sean and the gang definitely had to earn their respect for us to play significantly in their world, which is something they don't take lightly. Their faith in us was an enormous enabler to do this right.

Sean: "Here's our half-billion dollar franchise, bring it back with no dings …"

Susan: I think that's a really good point about Bungie, because they [were] obviously having a lot of anticipation about Halo 2 and a lot of people who are very interested in that story. To give us a framework to live within a very rich IP, and give us trust in no small respect due to the caliber of people who were here and writing—that was a great leap of faith for them, to let someone take their story, and it was almost like a prequel story to the game that was coming out.

Sean: They had some script approvals of the radio dramas, but the reality of this thing was that they were just not going to be able to control everything that we did.

AH: What's next for 4orty 2wo Entertainment and the alternate reality genre?

Elan: I think there's something very cool that's starting here. There's a lot of momentum behind this right now. The question I get asked all the time is: "Where is this going? What happens next?" It's something I have thought a lot about and never had a really good answer to; but I think what's really cool is that while we have no idea exactly where this is going, there seems to be a lot of people that want us to find out. ✖

Let's shed some light on mobile Java 3D development.

```
private Vector3D dir = new Vector3D(-3511, 731, 878); // Light
vector
private final int dirIntensity = 4096; // Light intensity
private final int ambIntensity = 1755; // Ambient light intensity

...

light = new Light(dir,dirIntensity,ambIntensity);
effect = new Effect3D( light, Effect3D.NORMAL_
SHADING, true, null);
...
g3.renderFigure(figure, 0, 0, layout, effect);
```

Java 3D™ doesn't just have the power to move the mobile gaming experience into a new dimension. It also has the power to do the same thing for your business. Because once a consumer's seen the light, there's no turning back.

No one is more committed to Java 3D than Sony Ericsson. And no one is more committed than Sony Ericsson Developer World to get your mobile Java 3D project on the fast track from mind to market. That's not just because we've got an unparalleled line of Java 3D technology-enabled phones on the market, but because we've got everything you need to dive into the development of mobile Java 3D games. From great tools and responsive tech-support to exciting go-to-market opportunities for high-quality 3D games from key development partners.

Go to www.SonyEricsson.com/developer/java3d to get enlightened on mobile Java 3D development.

With Java 3D in Mascot Capsule Micro3D v3, light is really easy to set up. A direction vector and an intensity value are all you need. You'll find more tips, tricks and code at: www.SonyEricsson.com/developer/java3d.

Sony Ericsson

www.SonyEricsson.com/developer

PUSHING BUTTONS

INTELLIGENT SOLUTIONS FOR AMBIGUOUS PLAYER CONTROLS

CONTROLLER IMAGES BY LAWRENCE WRIGHT

HAVE YOU EVER BEEN PLAYING A GAME AND, WHEN YOU pressed a button to make your character do something, the character either did something else or simply did nothing at all?

A core aspect of programming a game is transferring the player's input from the gamepad buttons to the character's actions, an aspect of the job that seems straightforward enough: you just map buttons to events. However, due to the different ways that players press buttons and perceive events, problems of ambiguity arise, which can lead to frustration and a feeling of unresponsiveness. For example, the player thinks she has hit the correct button or buttons at the correct time, but the intent of her button-pushing is ambiguous and cannot be resolved satisfactorily with a simple mapping.

What are the physical and perceptual reasons for this ambiguity? Different players produce different input, so developers need methods for analyzing what's actually occurring, a strategy for resolving input ambiguity that allows your game to feel responsive and intuitive.

CONTROLLER CONCEPTS

On a typical gamepad, such as the standard ones used with PlayStation 2 and Xbox, there are four buttons, all of which are operated by the player's right thumb. In this article I concentrate on these four right-thumb-controlled buttons.

One of these buttons is the "primary" button, the one used to trigger the primary action in the game. For example, in Mario games, the primary action is "jump." On the PlayStation 2 joypad, the primary button is X. On Xbox controllers, the primary button is A.

In this article I refer to the buttons as laid out on the Xbox joypad, mostly because each button is clearly demarcated by a letter: A, B, X, and Y. Moreover, the diamond shaped layout is the most common (see Figure 1). I also refer to the right shoulder button, or R1 for short.

MAPPING INPUT TO GAME EVENTS

You can map player input to events in a number of ways, but basically it boils down to a set of rules. Each game event, such as jumping or shooting, must satisfy a number of conditions before it can be triggered. In pseudo-code, we can express this as: If (conditions met) then (trigger event).

I'm going to use two examples here, both based on a simple Mario-like 2D or 3D platform game.

Example 1: Jumping. If our primary button, A, maps to the jump

MICK WEST *was a co-founder of Neversoft Entertainment. He designed, implemented, and debugged the skateboarding player control for the Tony Hawk's Pro Skater series. Email him at* **mwest@gdmag.com**.

FIGURE 1 This square arrangement of buttons on the Xbox joypad is the common arrangement on current consoles.

action, the rule could be: If A pressed then JUMP. But you don't want the character to be able to jump while he's in mid air; so a more reasonable rule would be: If A pressed and ON GROUND then JUMP.

Example 2: Super Jump and Ground Pound. Super jump and ground pound are actually two moves performed with the same buttons. In our game, pressing R1 causes the character to crouch. Pressing A while crouched causes the character to jump extra high.

If the player presses R1 while in the air, the character will do a "ground pound" where he rapidly slams down into the earth. (Both of these moves are similar to moves in SUPER MARIO 64.)

We can express these actions through three rules:

If R1 pressed and ON GROUND then CROUCH.

If R1 pressed and IN AIR then GROUND POUND.

If A pressed and CROUCHED then SUPER JUMP.

So far, this seems pretty straightforward. However, there are numerous problems with this simple implementation, as we'll see in a moment.

Making player controls work is inevitably a fiddly and complex task. In order to provide a feeling of simple intuitive control, developers will actually have to write some rather convoluted ad hoc code.

DETECTING PLAYER INPUT

The simplest way to monitor input from a joystick is to query its current state: "Is button X pressed right now?"

This method works fine in racing games, for example, where acceleration and braking are continuous events. But for one-time events, such as jumping and shooting, you actually want to detect when the player initially presses the button. The game needs to respond at the moment a button's state changes from "released" to "pressed," that is, when the button is triggered.

In addition, you often want the game to measure how long a button has been pressed, and sometimes how long it has been since a trigger event has occurred. You might also want to be able to flag the trigger event as having been handled to limit the player from continually reusing it.

Here, I'm going to assume a fairly simple implementation of a joypad button interface that provides the information listed above. This is also what I've implemented in the sample code available on gdmag.com.

PRECISE VS. SLOPPY THUMB

In playing a vast majority of games, the player rarely moves his or her thumb from the vicinity of the X, Y, A, and B buttons, so even when not engaging a button, the player rests their thumb over them, ready to press them as needed.

There are two basic ways people can hold their thumbs over the buttons: Precise Thumb and Sloppy Thumb. When implementing your control scheme in a game, it's important to consider both types of thumb positioning.

A Precise Thumb player uses the tip of his thumb to press each of the four buttons. When moving from one button to another, the precise thumb player lifts his thumb off the button and presses it down on another button, as in Figure 2A.

A Sloppy Thumb player uses her whole thumb to operate the four buttons. There are many variants, but basically, the Sloppy Thumbs use their thumb tip only to operate the uppermost button (Y). Sloppy Thumb players operate the side buttons, X

and B, with the sides of the thumb, and the A button with either the middle of the thumb pad or the bone at the first joint in the thumb. (See Figure 2B.) So, in order to press a different button, Sloppy Thumb players usually just tilt their thumbs.

GameCube controllers were built around the expectation that most players use some kind of Sloppy Thumb technique. There's a large central primary button with three other, smaller buttons surrounding it. This layout encourages players to keep their thumbs squarely over the primary button and hit the other buttons with the edges and the tip of the thumb.

This radical difference in thumb position and button layout can lead to perceived control problems that only arise on one platform and with just one tester. But remember, one tester out of your pool could equate to many thousands of players encountering the same problem if you're making a reasonably popular game. If you have very few testers and one of them says, "It feels wrong when I do this," then pay close attention. One tester in a small pool represents a good chunk of your potential audience.

THE CAUSES OF AMBIGUITY

Ambiguity Cause 1: Imperceptible state changes. The player runs his character toward a chasm. When the character reaches the edge, the player presses the jump button. But the character does not jump. He plunges to his doom.

Why didn't the character jump? In the internal model of reality that the game stores, the character had already run off the edge of the cliff, although this was imperceptible to the player when she pressed the jump button. To the player, it looked like the character was right at the edge, and a jump looked perfectly possible. This case shows how a discrepancy between the perception of time (to the player and to the game) can cause player frustration.

On one frame, on the ground, pressing A makes the character jump. On the next frame, in the air, pressing A has no effect. To the player, these two frames of the game appear identical because they are only just 0.016 seconds apart in reality. This same type of problem can also occur when jumping just before landing, or just before hitting a wall (for a wall jump).

Ambiguity Cause 2: Change in button function based on state. In Nintendo's SUPER MARIO 64 DS, when playing as Mario, pressing A to jump then R1 (or the Z button on the Nintendo 64 version), triggers a ground pound. Pressing R1 before A triggers a backflip. Pressing them both at the same time causes either a ground pound, a backflip, or a normal jump, seemingly at random—the user has no control. The player can press these two buttons simultaneously over and over, and never figure out how to control each of these three actions properly.

This problem also shows up in Mario when using the long jump action, by running and then pressing A and R1 simultaneously.
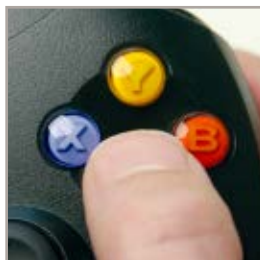
FIGURE 2A Precise Thumb players rest the tip of their thumbs over the primary button.
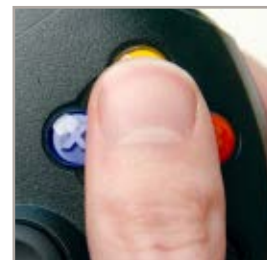
FIGURE 2B Sloppy Thumb players rest their thumbs over all the buttons, and press buttons by tilting the thumb.

>80%

25 of the top 30 media & entertainment companies trust Akamai.

They trust Akamai to deliver more than content. To deliver billions of downloads daily and give users what they want when they want it. Why? Because Akamai provides instant scalability to handle spikes in traffic, delivering positive user experiences and increased customer loyalty.
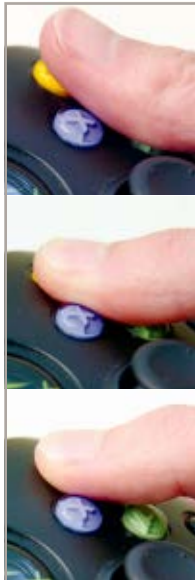
Start learning more and get your copy of *Online Games: 10 Tips to Increase Play and Profitability*. Call 888-340-4252 or visit www.akamai.com/gaming

Akamai
The Trusted Choice
for Online Business

FIGURE 3 Precise Thumb players move their thumbs from A to Y by lifting the thumb off A and placing it on Y.

FIGURE 4 Sloppy Thumb players move from A to Y by tilting the thumb forward, accidentally pressing X with the side of the thumb.

FIGURE 5 Precise Thumb players move from A to X cleanly.

Sometimes instead of a long jump, the character does a ground pound by accident. This is not the player's fault, who appeared to do everything right.

At this point, some game developers will say, "I'm pretty good at Mario, I don't notice those problems." That may be true, but to get good at it, you had to learn that to do a consistent long jump you had to tap R1, wait 0.05 to 0.18 seconds, then tap A. This is something players must learn by trial and error, failing repeatedly when they first play the game. To a beginning player, this just isn't any fun. Controls that do not do what you want them to do are frustrating. Just as bad are inconsistent controls—sometimes the character does one thing, and sometime another, without any perceptible logic to the player. To these people, the game "just feels wrong." Ignoring the needs of these players when developing games and mapping player controls means that you're snubbing a large part of your audience.

Ambiguity Cause 3: Accidental button presses with Sloppy Thumb. When Sloppy Thumb players move their thumb from one button to another, it's likely that they may hit some other button accidentally, particularly when going between opposite buttons. For example, when switching between A and Y Precise Thumb players fully lift their thumbs off A and then press Y (Figure 3) in a nice clean motion. Sloppy Thump players tilt their thumbs forward, rocking it from A to Y and

quite possibly momentarily pressing X or B (Figure 4).

Ambiguity Cause 4: Release and press vs. press and release. It's common in games for an action to involve releasing one button and then pressing another in rapid succession. The player can do this in two ways. Let's say the player needs to hit A then X. A Precise Thumb player releases A, then presses X (Figure 5), whereas the Sloppy Thumb player presses X while still engaging A, then releases A; for a brief period of time, both A and X are pressed (Figure 6).

This can lead to problems if you have a different event triggered by A+X or if your event tied to A relies on no other buttons being pressed.

UNAMBIGUOUS SOLUTIONS

In dealing with all these problems, the most important tool is one that gives you the ability to see exactly what's happening, and more importantly, what has happened. When you're tasked with deciphering control problems, you have to be able to figure out the precise sequence of events that caused that problem.

To measure and track these problems, you first should implement a simple log file. Every time an event of interest occurs, print out that event along with the time and any other useful information. Then when the control problem crops up, examine the log file to see what caused the problem. I use the "OutputDebugString" Win32 function, since there are numerous programs designed to capture and filter these debug strings.

Consider the situation in Figure 4. The corresponding output log is shown in Table 1.

The player presses A to jump (which happens in the same frame). Then the player rolls his thumb forward pressing Y and X simultaneously. As the thumb rolls forward fully onto Y, the player releases X and A simultaneously.

FIGURE 6 Sloppy Thumb players often press A and X at the same time when trying to move from A to X.

TABLE 1: Figure 4 Output Log	
Time in s	**Event**
14.218	+ Pressed A
14.218	Event Jump
14.418	+ Pressed X
14.418	+ Pressed Y
14.484	− Released X
14.484	− Released A

Now consider the case for one of the ambiguity problems we listed above, the super jump and ground pound. Pressing A and R1 at the same time resulted in inconsistent results. Table 2 shows the output for a ground pound, whereas Table 3 shows the output for a super jump.

Note that in Table 1, A is pressed 0.016 seconds before R1, so the game triggers a jump first, and then a ground pound. In Table 2, R1 is pressed 0.016 seconds before A, meaning the game triggers the crouch; then as the character is crouched, it triggers a super jump when A is pressed.

To the player, it looks and feels like he attempted to do the same thing both times: He pressed A and R1 simultaneously. There's no way whatsoever to distinguish between the two events. To him, the game responds inconsistently and ambiguously, which is annoying—and your job is to stop this from happening.

Log files are useful but can be rather dense and difficult to pore over. The lines all look alike, and the problem can be hidden in a dense flurry of events.

However, putting the information into a graphical format can make the data much easier to understand.

I added a simple "watcher" class to my log file, which records the value of a variable—such as a button up/down state, or a physics state or flag-over every frame and then displays it as a scrolling state graph across the top of the screen. There's a separate line for each variable that I watched, as you can see in Figure 7.

To this state graph, I added an event recorder that recorded events (jump, land, fall, super jump, late jump, and crouch) and

TABLE 2: Ground Pound Output	
Time in s	Event
21.199	+ Pressed A
21.199	Event Jump
21.215	+ Pressed R1
21.215	Event Ground Pound
21.232	Event Land
21.249	Event Crouch
21.432	- Released R1
21.432	Event UnCrouch
21.465	- Released A

TABLE 3: Super-Jump Output	
Time in s	Event
33.778	+ Pressed R1
33.778	Event Crouch
33.794	+ Pressed A
33.794	Event Super Jump
33.961	- Released A
33.977	- Released R1

FIGURE 8 Timing graph for tapping A and X with Precise Thumb (A) and Sloppy Thumb (B).

displayed them on the graph as a vertical line, labeled with the event. Finally, I added the ability to scroll through and zoom in and out of the graph using the joypad when the game is paused. So, whenever some control problem occurs it's very easy to pause the game and scroll over and zoom into the area on the graph that caused the problems.

Take a look at a simple example, sans events. Figure 8A shows the state graphs for the A and X buttons that correspond to a precise player tapping A and then X. A raised line indicates that the button is pressed. The lines along the bottom represent 0.1 second intervals. So, we have a series of button presses for each button, each lasting about 0.1 seconds, separated by times where no button is pressed for about 0.1 seconds.

Figure 8B, however, shows the same situation but with a Sloppy Thumb player. Here, the presses last much longer 0.2 or 0.3 s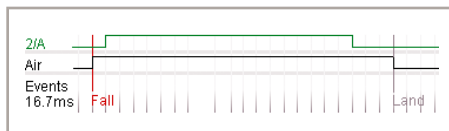econds, and the button presses overlap, with A and X depressed at the same time. Note, though, that the total time between the first press of A, and the last press is about the same in both examples.

Let's look at the late jump problem. The player runs to the edge of the cliff and jumps—well, tries to jump, but the character falls to his doom. Figure 9 represents this scenario. I've added a watcher on the "Air" state so the transition between ground and air is visible. The graph shows that the player actually hit the jump button just 16.7ms after leaving the ground. There are two major problems here: First, the amount of time is too short for any human to recognize, and second, the frame display showing the character clearly off the edge of the cliff has not even been rendered yet! The player is not going to—and should not have to—understand that the game's internal representation of his position indicates he can't jump. He's just going to be annoyed that it looked like he should jump and did not.

What can programmers do to fix this problem? The slightly non-intuitive solution is to allow a jump to occur for a short period after the character has left the ground. Typically, this period will be 0.2 seconds or less, unless

FIGURE 9 This jump attempt failed by just 0.016 seconds.

it's a 3D game in which case you'll probably use a longer time, as the visual representation is less precise. We can express the new rule as If button A pressed and IN AIR and ON GROUND 0.1 seconds ago then JUMP. Allowing a late jump removes the uncertainty.

Figure 10 shows this solution in action. A is pressed two frames after the player starts to fall, but the jump is still permitted. And generally, this late jump will look just fine. It might look a little unusual if the player jumps at the very extremity of the late jump window, and you have to make sure your animation and sound don't glitch when the character momentarily goes from grounded to falling to jumping in the course of 0.1 seconds. But players will tolerate a little oddness now and then far more than they will tolerate unresponsive controls.

For the second problem, the super jump versus ground pound. Figure 11 illustrates that A is pressed and causes a jump. Then, on the next frame, R1 is pressed in the air so we immediately get a ground pound and then land again, all in the course of 0.033 seconds.

The reverse case is shown in Figure 12—the player hits R1 0.016 seconds before A, so the character crouches, and then super jumps. Again, the difference between the sequence of events in Figures 11 and 12 is just 0.016 seconds, one-sixtieth of a second, not enough time to blink.

To fix this problem, let's first decide what pressing A+R1 should actually do. Since the character is on the ground, the player is probably trying to do a super jump. So we can add another rule similar to the late jump: If A+R1 pressed and IN AIR and ON GROUND 0.1 seconds ago then SUPER JUMP.

RESOLVING AMBIGUITY

I've presented some specific solutions here for two examples of ambiguity by adding additional "disambiguation rules" to your player-control logic. To achieve truly intuitive and responsive player controls, this type of disambiguation rule is going to be your primary tool.

Disambiguation rules are very application-specific; they also can end up being rather complicated. But if you detect a control ambiguity and you can add a rule, then add it. The logic you're implementing with this solution is not elegant. Inevitably, there will be a high degree of ambiguity that can only be resolved by a similar degree of complexity. Your code and data will need constant adjustments to make this work.

To fix a control ambiguity problem, first find out exactly what's happening. Then ask yourself,
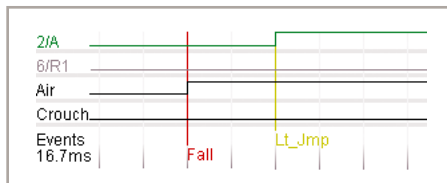
FIGURE 10 In the late jump in action, the character jumps even though it has been falling for a couple of frames.
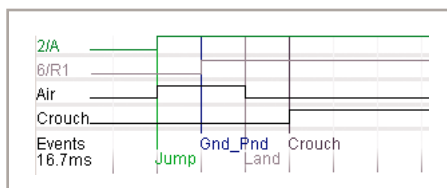
FIGURE 11 The player tries to super jump by couching then jumping, but accidentally jumps one frame before crouching, causing an instant ground pound.
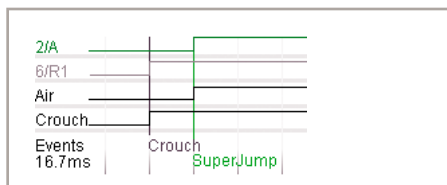
FIGURE 12 With just a fraction of a second difference in input, the player successfully performs a super jump.

"For this state and this input, what should happen?" Add disambiguation rules as necessary to make the correct result happen. Test. Repeat.

DISAMBIGUATE NOW!

Like game designers and programmers, testers too need a method of approaching ambiguous controls. Teach them not to accept the slightest annoyance, and instead figure out exactly what they're doing to trigger it so they can record their results.

An ideal framework for player control development would allow you to record the game state and inputs for the duration of the problem, and then replay it, editing the rules on the fly until the problem is resolved. Your audience will thank you for it. ✖

Desire takes many forms.

Having unparalleled control and mastery over your 3D world may not be the first thing that comes to mind but let's be honest, it's right up there.

![3Dconnexion logo] 3Dconnexion

Introducing SpacePilot™, the Intelligent Motion Controller. Forged from the highly successful SpaceBall and SpaceMouse, SpacePilot represents a giant leap forward in intelligent interface devices for professionals. Featuring adaptive sensing technology that automatically gives you the functions you want, when you want them and an array of extendable keys that put unlimited functions within a fingertip's reach. Master your 3D world today – learn more at: www.3dconnexion.com/newspacepilot

A LOGITECH COMPANY

POSTMORTEM

Riding The FBI With ALIEN HOMINID

ARE YOU A BURNT-OUT SHELL OF YOUR FORMER SELF DUE TO INHUMANE working conditions as you toil away in your cube at GloboCorp Games? Are you afraid to be the first to leave work for the day because you'll end up on The List? Does the word "crunch" make you break out in a cold sweat, even if you are just talking about breakfast cereal? If so, leave work early today and join us on a romp through indie heaven and hell as we bring you into the studios of The Behemoth and the making of ALIEN HOMINID.

ALIEN HOMINID is the debut title from The Behemoth. A PlayStation 2 and GameCube console title (with an Xbox version having now gone gold in Europe), ALIEN HOMINID allows players to race across the earth in frantic recovery of their spaceship. Construction of the console version began on April 1, 2003 and the game shipped on November 17, 2004. As a traditional 2D side-scrolling shooter with a twist, ALIEN HOMINID has received numerous awards, including Independent Games Festival 2005 awards for Technical Excellence, Innovation in Visual Art, and the coveted Audience award.

ALIEN HOMINID began life as a Flash game built by Dan Paladin and Tom Fulp. It was rolled out on Newgrounds.com (Tom's "other job") in August 2002, and to date has racked up more than 10 million views on Newgrounds alone.

Soon after its release, a link to the Flash game landed in the inbox of John Baez, who was working as an artist at Gratuitous Games on an Xbox port of Raven's SOLDIER OF FORTUNE II for Activision. John immediately realized that whoever brought the title to consoles would be a hero.

At the time, Dan was with Presto Studios, having recently wrapped up artwork for WHACKED!, an Xbox Live launch title Presto was developing for Microsoft. In the fall of 2002, Presto decided to shut down operations after WHACKED! shipped, and Gratuitous hired the remnants of its art department. Suddenly, the artist for ALIEN HOMINID was just down the hall.

A few months later, the owner of Gratuitous Games decided to shut down his company. Gratuitous had made a name for itself by doing ports for Midway, Activision, and Crystal Dynamics, but each year it was harder to get quality work, because many of the big publishers were using their in-house studios. Once SOLDIER OF FORTUNE II went gold, Gratuitous would close.

Not wanting to go back into the tight job market, they decided this was a perfect opportunity to make a console version of ALIEN HOMINID. Dan sent Tom an email that basically said, "Some guys from work want to take a lot of money out of the bank and make an ALIEN HOMINID console game." Tom decided to humor him and replied, "Okay!"

Eighteen months later, ALIEN HOMINID was on store shelves. It's one of the first web games to cross over to consoles.

WHAT WENT RIGHT

1 SMALL, EFFICIENT TEAM. Part of the success of ALIEN HOMINID was that we kept our team very small throughout development. A worldwide, multi-SKU console title on the current generation of hardware will often take upwards of 25 people to make. Our small team size meant everyone had to do double or triple duty, but it also meant that our risk was reduced because we didn't have a bunch of guys standing around waiting for meetings or needing to be told what to do.

Aside from Tom, all of us had worked on the production side of game development before, so our learning curve was relatively flat and we could concentrate on creating the game instead of learning new software. This, combined with the fact that we had all worked together at Gratuitous is probably the number one reason we succeeded.

Being small meant that each person had more responsibility than was probably healthy, but we grew into our new roles well. For example, John's background was in architecture and level building, but in the new company, he would be the strategist, money guy, business guy, and toilet bowl scrubber. Dan would move from being an artist to being an art director, now with the added fun of reversing roles with his old art director.

In the end, our core team of eight full-time developers and a handful of part-time friends brought back to life the garage atmosphere most people in the game industry thought died out a decade ago.

TOM FULP is the creator of the Internet entertainment portal Newgrounds.com and co-founder of The Behemoth. Tom was the programmer behind the original web version of ALIEN HOMINID.

JOHN BAEZ is a nine year industry veteran. As co-founder of The Behemoth, John is in charge of running its day-to-day operations and was the producer on ALIEN HOMINID. He is also a licensed architect in California.
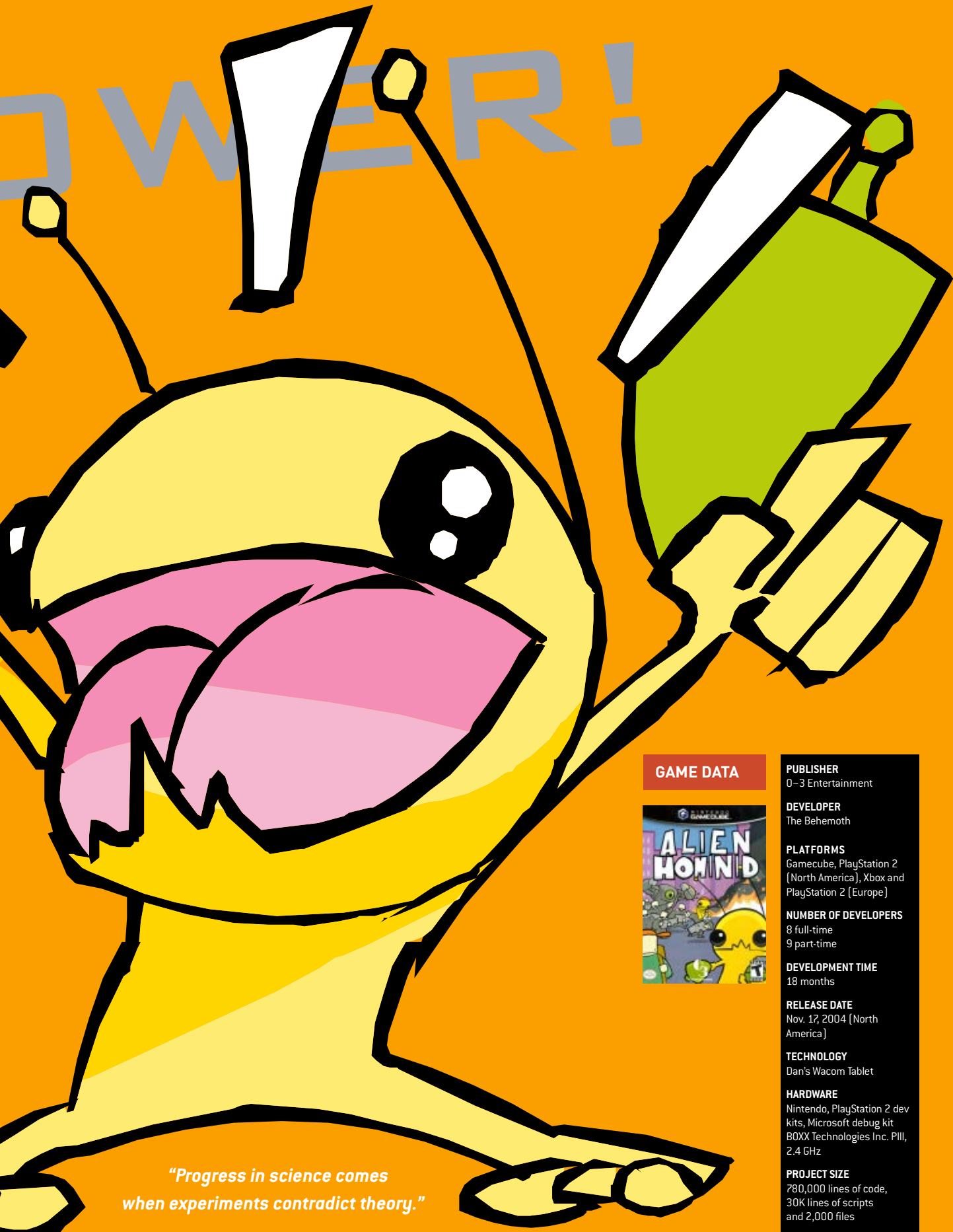
> *"Progress in science comes when experiments contradict theory."*
>
> *~ Richard Feynman ~*

The Behemoth sticks it to the man, ALIEN HOMINID **style.**

2 OUR OWN TECHNOLOGY & IP. Being small and self-funded meant we couldn't spend money on middleware. Knowing this from the start let the programmers rip into the engine, aware there would be no one to turn to if things didn't work out. It was also key for us to simultaneously develop for as many consoles as we could since we had the experience from our Gratuitous Games days and it would be a shame to squander that advantage. Publishers took notice when they saw it running on three consoles, which was a big bonus later on. At the same time, we knew we would not be able to develop our own in-house content creation tools, so we had to build our engine knowing that we would be using off-the-shelf content creation software like Flash and Photoshop.

In the beginning of development, we experimented with 3D rendered versions of the characters. They looked really slick, and much more "professional" than the original Flash version. Dan prepped a 3D cel-shaded version of the alien, which also won our hearts for a moment. As we progressed with this style, we found it was totally lacking the charm and character of the original. Once the decision was made to drop 3D all together, we developed a tool that allowed us to easily incorporate Dan's vector art, hand-drawn with a Wacom Tablet.

3 SELF-FUNDED. From the get-go, we knew it would be hard to get anyone to fund our risky game venture. Although we had been approved to develop through the console manufacturers' unsigned developer programs, many of the people we talked to insisted a 2D game would never be approved and we were wasting our time. Others believed we were too small to pull it off, or we didn't have enough business sense because we were down-in-the-trenches workers, not up-in-management guys with good connections throughout the industry. Naive or stubborn, we believed in the project and knew it could make money, so we assumed the risk of funding it ourselves. The initial "seed capital" came from John mortgaging his house, which had seen a lot of appreciation during the San Diego real estate boom.

We thought this initial loan would cover the complete development of the game, which we roughly estimated at nine months. However, as the game continued to become more and more rich, we decided to give it more time to grow organically. Josh Barth, Scott Fadick, and Chip Burwell, critical programmers on the project, all kept their wages far below market rate to help

keep the lights on, and Tom started dumping in money to continue funding and keep us strong. At times, this put a lot of strain on the team, but since we could see the game getting stronger on a week-to-week basis, it made it all bearable.

As much as we believed the game would eventually pay for itself, we realized we needed to create some alternate revenue streams to keep ourselves going, so it came time to explore merchandising. The great thing about owning your intellectual property is you can do whatever you want with it. We decided to make t-shirts, skateboard decks, and figurines. In order to maintain a high level of quality control, we decided to bypass middlemen and produce everything through our network of friends. The figurines were sculpted by Clint Burgin, a local artist in San Diego. Clint had a friend who made regular trips to China, so we paid him to be our factory liaison.

Ultimately, we discovered that by self-funding we had stumbled upon any number of revenue streams. Since our North American publisher had no presence in Europe, we decided to license the game on a per territory basis. Initially, we accepted mail and check orders only for the figurines and other merchandise, but closer to the console release we launched our online store and worked out the kinks of the fulfillment. By creating a multi-SKU title that we sell online with cross-promotional merchandise, we've avoided the traditional developer/publisher relationship of a single, worldwide revenue stream.

4 SELF-PROMOTED AND CULTIVATED FANBASE. One thing that gave us all great confidence in the success of ALIEN HOMINID was the existing fan-base on Tom's site, Newgrounds.com. The Flash version already had over five million downloads when production began, and Newgrounds entertains millions of unique visitors every month. Tom was able to actively market ALIEN HOMINID online, so even without paid advertising, we still moved units. We were also able to conduct polls relating to the game. To our astonishment, we had over 65,000 people take a lengthy poll which queried everything from how many times the person had played the prototype and where they bought games to what gender they thought the alien was.

As the owners of our intellectual property, we were free to promote ALIEN HOMINID anywhere we wanted. By this time it was July 2004 and we had just received concept approval from Sony, so Comic Con 2004 was going to be the first official unveiling of the PlayStation 2 and GameCube versions. We set

Alien Hominid features quite a few minigames, including this retro-styled PDA themed retro platforming game.

[Other competitive solutions require multiple supporting packages to handle our assets. With **Alienbrain**, everything we need is in one package.]

Christopher Bretz Art Director, Secret Level

Starts at $690

Brainpower

Avid Alienbrain Studio 7.1 has the intelligence to help your team work faster and smarter.

Its intuitive interface tells you more than the what, where and who of an asset. It tells you what an asset references, what it's referenced by, and whether it's buildable, it's optimized, or it's redundant. It knows what platform an asset is for and what should happen to it next. It even knows an asset's location in a texture map, its resolution in level three, and how much damage it inflicts when eaten. Alienbrain knows. You know. Your pipeline knows.

ASSET MANAGEMENT FOR CREATIVE TEAMS

Free eval copies available at www.avid.com/alienbrain

alienbrain STUDIO

www.avid.com/alienbrain

Avid computer graphics

up a big, beautiful booth at the show and promoted the nearly finished game to 87,000 attendees. We even made the cash register sing by selling figurines, t-shirts, and skateboard decks. The hit of the show was the yellow alien antenna headsets we gave away to the crowd. We only brought 5,000, but should have brought at least five times that many; the demand was way beyond our expectations. The event won over a lot of new fans and landed us in several magazine spreads and on TV.

5 A NEARLY FINISHED GAME=MULTIPLE OFFERS. Back in 1998, John attended the Game Developers Conference in Long Beach, Calif. In one of the sessions, the lecturer commented about how he started his company in his basement and had done a tremendous amount of work on a shoestring. It was a huge feat, and he recommended that no one in the audience follow in his footsteps. People refer to this as the "don't try this at home" talk. However, in case someone did want to attempt it, he recommended that you nearly finish the game before looking for a publisher so that there is less risk on their end. The speaker said he had received multiple offers for his game due to this foresight. Well, lo and behold, five years later, John found himself cold calling publishers with ALIEN HOMINID stuffed in his backpack. Far beyond the mythical "vertical slice," our pitch package included a robust copy of the game running on any of the three consoles the publishers cared to see it on. We also developed a slick presentation that would ultimately be refined over the next few months as we visited with more than 20 publishers.

While some of the top tier publishers were quick to tell us, "Thanks, but no thanks. Your game is fun but won't sell a million units," we kept at it. Finally, a few months before E3 2004, we started to receive serious offers. In a last attempt to get in front of as many publishers as possible, we took ALIEN HOMINID to the first U.S. Game Connection held during GDC 2004. Run by a non-profit group of fashionably dressed French guys (Lyon Game), the concept sounded great even if it seemed expensive. John was able to fly in on Tuesday morning, grab a cab to the Connection, check in, set up, and display the game running on all the consoles to eight publishers, pack up, jump back on a plane, and be back in San Diego by dinner time. The feedback we received helped us move forward with our negotiations and helped us begin meeting European publishers.

WHAT WENT WRONG

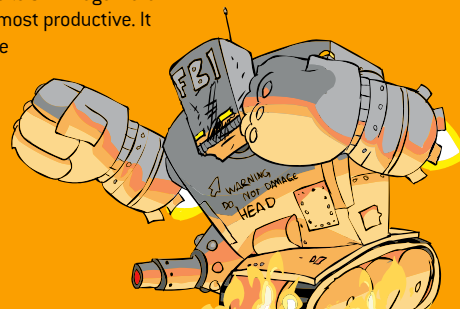**1 WE NARROWED OUR CHOICES TO JUST TWO PUBLISHERS.** Looking back, almost a year to the day that we signed our contract, we realize now that we should have done a complete and thorough due diligence on each and every one of the publishers who made us formal offers. It was something we

really didn't know how to do beyond the cursory checks and it seemed to put a shadow on the negotiations when everything was looking rosy at the start of a relationship. But these days, when developers and publishers are closing their doors left and right, it is critical for a developer to really get to know their future publisher. Forget about being nice. Roll up your sleeves, put a pair of rubber gloves on, and find out who you are really dealing with because it takes a lot of cash to publish, distribute, and adequately market a game. This is especially critical if the developer is bringing new IP to the table or if your publisher is just getting established. You can't launch a franchise from scratch without extensive marketing.

For us, the results of a good due diligence alone would have probably reduced the field considerably and would have allowed us to further pursue offers that were beginning to form. Ultimately, we decided that one publisher was pretty much like another when in fact they are not. Add to this a big helping of being relieved to have any offer at all (let alone four) and the belief that we knew what we were doing when in fact we were pretty clueless. Those are the lessons you learn with your first deal, which is what makes you stronger for the next deal.

2 DISTRIBUTED STUDIO DIFFICULTIES. When we started with ALIEN HOMINID, Tom was living in Philadelphia and the rest of the team was in San Diego. Over the course of development, Dan moved from San Diego to his hometown in Cleveland, then to Philadelphia to work directly with Tom as he programmed gameplay, then back to his hometown in Cleveland after the game shipped. Matt Harwood, our music composer, was in New York, but all of the source files and console coding was done in San Diego, which meant we had to depend on our Internet, VPN connections, and Source Safe to work from all over the country. To put it gently, the experience was horrible. It took forever to check content in and out of the system, and remote computers often dropped the connection mid-transfer. For security, we used SST for communication instead of AIM or MSN. I don't know what the problem was, but SST was slow and lagged too much. We plan to use different source control and messaging tools on the next project.

During development, Tom made frequent trips to San Diego, staying for several weeks at a time for crunch sessions with the team. The trips to San Diego were definitely the most productive. It was great to be able to shoot ideas back and forth in realtime and feel the

Figurine models went through many phases, from initial clay molding, to a repositioning based on Dan's corrective sketches, to resin, to the eventual finished product.

POSTMORTEM

The complete ALIEN HOMINID team, taking a break after curing cancer.

excitement and energy in the room. We would all get completely pumped and would work insane hours. Not having to deal with an Internet connection, Tom was able to get infinitely more work done in a short amount of time. Having always considered himself to be a champion of telecommuting and distributed offices, Tom now admits it is best when you stick everyone in the same room together.

3 **OFF-SITE CONTRACTORS.** On the topic of sticking everyone in the same room, this is especially true for short-term contractors. We were a small team with a lot of content to produce, so bringing in friends as content creators seemed like a great idea. We sought out some of the best online talent we could find to assist with development, but in the end, a lot of artwork was redrawn and code was rewritten. Their work was done early in the project and as it lengthened, we decided to stop using contractors. These were great artists and programmers, but we didn't manage the sudden team growth well and in the end wasted a lot of time and money having things developed that weren't put to proper use. When everyone is already incredibly busy working on their own things, it's hard to maintain good communication with people who are sharing the workload from distant cities. In the future, we plan to keep all content creation local, whether it be local to the San Diego or Philadelphia office. We don't intend to rely on anyone we can't meet for lunch.

4 **SCOPE CREEP.** Forget story boards and design documents! Aren't outlines for high school English class? Dan and Tom are both pretty impatient when it comes to planning. Their ideas come as a stream-of-consciousness river full of mud, rocks, and the occasional gold. Making ALIEN HOMINID was a matter of panning for that gold on a daily basis. Planning ahead was like panning in stagnant water; months later, you're stuck with the same nuggets that were there when you started. Everything great about ALIEN HOMINID comes from our constant desire to add new things and to let the design grow organically. However, that was also a big problem and caused a lot of frustration for the rest of the team on many occasions.

The hardest part of ALIEN HOMINID was calling it "done." The programmers built a development system that made implementing new features very easy, but new features always introduced new bugs and there came a time when we couldn't afford to have any more new bugs. We called this time "content lockdown." We actually had several content lockdowns through the course of development. This was the equivalent of a parent saying, "Playtime is over, time for bed!" You always know you can stay up for 15 more minutes. John was the parent while Dan and Tom were the kids, full of sugar with toys all over the floor, not ready for bed. The thought of no longer being able to add neat things to ALIEN HOMINID caused stomach aches and insomnia for many nights. It is the crux of anyone who creates something; you're

never completely satisfied with your work and there is always room for improvement. In the future, we all hope to stick more closely to a plan, for John's sanity, if nothing else.

5 TESTING, TESTING, TESTING. As our Alpha deadline loomed on the horizon, we decided we should begin a routine testing regimen. The other publishers we had worked for in the past always had their own quality assurance teams and did not require assistance from the developer; however, we knew that our publisher was still ramping up and time was getting critically short.

Originally, we used Dan's fiancée who could school anyone in just about any game, but when they broke up, we knew we'd have to get a full-time replacement. We decided the testers should be located in San Diego with the console team, so we ran Monster.com ads and quickly had a stable of testers to choose from.

Our publisher brought in their testers in August, but we soon realized the game wasn't getting the punishment it deserved, as our internal tester Emil was able to log more bugs than the whole crew of testers at our publisher. Testing would haunt us after the launch of the game, even though the game had gotten a clean bill of health from us, our publisher, and Q/A at both Sony and Nintendo.

After ALIEN HOMINID hit retail, a humorous bug began appearing on user forums. It seems that when a single player exhausted their stock of lives and began to use the second controller and Player 2's lives, one boss would exit stage left in search of Player 1. In a way this bug was kind of charming, as if the boss wanted Player 1 to come back to enjoy another pounding. Ultimately, it was frustrating for only a few players, but it was unnerving for us.

ADVANCE TO NEXT LEVEL!

In the end, we chalk up a lot of our successes and problems to being a new developer. We learned a lot about planning, hierarchy, marketing, testing and more. We went far beyond where any sane developer would go by manufacturing our own merchandise, opening our own online store, and doing our own marketing and PR. Want to know where to buy the best industrial 1-inch button making machine or where to find the best walk-around suit manufacturer? Ask us. Need a trademark lawyer in Europe? We've got one.

Yet these lessons came with a price beyond the hard knocks of learning. When "quality of life issues" is the buzz term of the day, being independent makes our lives that much more stressful because there is no one here to give us a paycheck every two weeks or load us up with benefits. Just imagine going to your spouse and saying, "Honey, I want to start a video game company and the hours are going to get worse, not better. I won't be drawing a paycheck anymore and we'll have to liquidate our savings accounts. And one more thing—I need you to sign the mortgage on the house." Believe us, the only reason this game ever made it to the shelves is because of our wives, kids, and the girlfriends who stuck by us. They never once stopped believing in our crazy dreams. ⁘

Concept model for a polygonal ALIEN HOMINID.

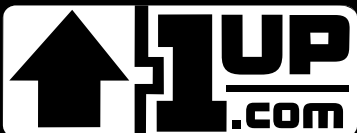"OUR 1UP BLOG IS GREAT FOR KEEPING FANS UP TO DATE ON WHAT WE'RE DOING AT WIDELOAD GAMES. IF YOU GOT ANY CLOSER TO STUBBS, HE'D EAT YOUR BRAIN!"

[ALEX SEROPIAN, PRESIDENT, WIDELOAD GAMES

CO-FOUNDER, BUNGIE SOFTWARE]

HTTP://SEROPADOPE.1UP.COM

1UP.com

Games.Blogs.Clubs.Home.

'SOFT LOVE

J ALLARD TALKS
XNA, HANDHELDS, CREATORS

SPEAKING TO *GAME DEVELOPER* **AT MARCH'S GAME DEVELOPERS CONFERENCE**
in San Francisco, Microsoft's corporate vice president and chief XNA architect
J Allard was ebullient as ever about his company's incursions into the game
market. This month looks to be the jumping-off point for the next Xbox
hardware, and as such, Allard was relatively tight-lipped about it. But he was
happy to spiral into much more interesting tangents when discussing
development methods (particularly XNA Studio, Microsoft's integrated, team-
based development environment tailored specifically for video game
creation), as well as the kind of developers Microsoft is partnering with, and
the ever-intriguing handheld market, orating in such an animated fashion that
we almost felt compelled to nod through sheer charisma alone.

*Game Developer: Are you happy with developers making games in XNA Studio
for any [console] hardware?*

J Allard: Sure!

GD: That's the one-word answer?

JA: I mean, why not? It's a stupid strategy to say, "Let's go solve problems for
a small set of people in this little corner of the universe." The reality is, the
future is polymorphic IP. It's cross-platform game development. Different
developers and different publishers are going to have different expectations
of business plans, and we have to enable those. The biggest pain point in
game development today is workflow. We can contribute materially to helping
solve this—no other company can.

We have Visual Studio as an asset, we have this Team Server work and all
this stuff that our storage team is putting together, we have the
relationship with the middleware companies, we have this first party—let's
take all the talent and the technology we have, put it together and solve
one of the number one development problems for the industry! If we do
that, it'll help our platforms too. Is it equal? I don't think it's equal. I'm sure
we'll put some extra stuff in ours that we have lying around. You know, I

think it would be a mistake for us to turn our back on the rest of the
problems the industry is facing.

Something I didn't focus on so much, but which is very important to those
that read and click, is that XNA Studio isn't intended to be holistic one-size-fits-
all. It's not a Swiss army knife. It's really a foundation for workflow and the way
you stitch together these off-the-shelf tools. It's more like your toolbox, with a
handle and a couple of standard pieces in it. Then you go and you buy all the
tools you want, or you build the tools you want and you stitch them together.

You'll go author your textures in TIFF, and then you'll be able to put in a
transform for Windows, and you'll be able to write custom transform for
Xenon, and jam it through the build pipeline, and say, "Build me the Windows
version, build me the Xbox version," and it just works! That could be an off-the-
shelf part with a third-party middleware provider, or it could be your own
code. I mean, we're not trying to recreate the wheel. There's lots of
middleware, but it just doesn't snap together well.

*GD: With all the consolidation these days, first parties seem to be at a
premium. What's Microsoft's feeling about the acquisition of Rare?*

JA: I love those guys. I mean, the soul of everything we do with Xbox is
centered in gaming. … We want to create the ultimate canvas for the best
artists in the world. The Rare team is just phenomenal. They've given us great
feedback on the platform all along, and the great thing about being a first
party and being Microsoft is that we can be very patient.

HALO was four-and-a-half years in development. And remember at E3 2001,
people were saying, "You guys really blew it. You're bringing crappy PC ports
over, the frame rate sucks, you wasted your money on Bungie." … Be patient.
It'll pay off. I think it's a great relationship because the Rare wizardry, the

---

creativity and the force which is Rare, the soul of Rare, requires patience. It's a perfect relationship.

GD: *How important do you think wireless will be in the next generation of console [as emphasized in Nintendo's GDC keynote]?*

JA: I said it at least once [in my GDC keynote], but it doesn't change what developers do. … My point about wireless: As the most high-tech devices in the living room, why are game consoles the only things with cords? Kind of dumb. I think that's something we have to fix for the users, but it doesn't change what the game creators do. So I really wanted to emphasize that the leg that we did show on Xenon is really about the stuff that matters to the game creators. Think about achievements. Think about connectivity. Think about self-expression. These are going to be the hallmarks beyond the 16x9 visual stuff. Those are going to be the hallmarks of the next generation. That's what I really wanted to spark in people.

GD: *What inspired you to pursue Japanese game creators, specifically those that have splintered from their original companies, such as Tetsuya Mizuguchi, Yoshiki Okamoto, and Hironobu Sakaguchi, and are now making next-generation Xbox games?*

JA: They're awesome!

GD: *Do you see this effort as a good way to rebreak into the Japanese market? It seems very focused.*

JA: Let me be really transparent about our agenda. Our agenda is to go redefine how people have fun. If we're the leaders in that, there's probably some money in it. In order to be the leaders in that, you've got to partner with the best people in the world. I've met with Sakaguchi since 1999, sat down with him, said, "It's an honor to meet you," some nice pleasantries and gave a lot of compliments about the work he's done. And actually meeting him for the first time, I told him what our plan was. He said, "That's nice. Here's what's wrong with it."

In 2000, I showed up in Tokyo and had another meeting. He said, "That's very nice. No." I said why not? And I listened. Our strategy with Japan—and frankly anyone who doesn't believe in what we're doing, in the creative community in particular—is to listen and understand why they don't feel like we're on the right track. And after six years of meetings, it's great to hear him finally say yes. Part of the reason he said yes is because there's a little bit of Sakaguchi in what we did in 2000, 2001, and 2002, because we heard it.

I've got no big ideas—J doesn't have any big ideas! J gets to go talk to the greatest game creators in the world, hear what they have to say, simplify it a little, make it into a couple of basic concepts, deliver the tools so that people can go realize it with less effort—that's my job! My job is to enable these guys. They're the rock stars of the world. [I bring] the rock stars to our platform and make sure the platform is appropriate. You know, making sure the rock stars have the best instruments, that's all we can do. They make the music. The music is what sells.

GD: *What's been the biggest challenge for you in terms of getting Xbox Live adopted by developers and publishers?*

JA: I think we've been remarkably consistent with our vision and values, which is what drives us. I mean I think that's the long-term success strategy. But sometimes it takes a while for your actions to catch up. A lot of people are going to judge you based on your actions, not your words.

With Xbox Live, a lot of people said, "I love everything you're saying, but I question whether people are going to buy it. I question whether people are going to be comfortable with voice. I question whether it's going to be a lot of work from a development point of view." We had to do the heavy lifting in first party. We had to launch MECHASSAULT. We had to partner up with THQ to do MOTOGP. We had to show people what the metaphor was. We had to drive some early adoption, and we had to go do XSN Sports. [With the] later adopters, we announced [EA's Xbox Live support] last year at E3, and they're like, "Hmm,

you kind of paid off what you said and what your vision and values were to build an online community that [Microsoft was] going to be able to manage, that you were going to be able to scale, that you were going to be able to make work for developers, that wasn't going to give control over. You really did what you said you were going to do. Okay, now we're ready." Kind of like Sakaguchi.

GD: *Is the move toward [Xbox] Live and customization also a move toward updatable games?*

JA: There's just such a wide spectrum of opportunity. There are two opportunities that I center on that I get most excited about. I'm very excited about opportunity number one, which is how you decorate and give the [game] a little more life after you've launched retail. That's what we're doing with HALO 2. We're listening to the community, we're finding out what's going on, we've made a few changes, we're tuning it up. No matter how much testing you do before you release an online experience, you didn't do enough. Once it gets in the hands of six million people, they're going to discover something that you didn't think of. I don't care how good the Bungie guys are. We've learned a little bit. I think there's the tuning up of the experience, and adding some customization with things you know that work, and so people are really into the Warthog, so the next vehicle you release is more like the Warthog, and less like the tank, and that's good!

There's another thing which I think is particularly interesting as we shift into the stratosphere of $10 million game budgets. You're a publisher. You've got 50 million bucks. You can do five game concepts because they're $10 million each. What are you going to do? You're going to do three sure things. You're going to go chase GTA, TONY HAWK, and HALO, and steal the best ideas from that. The next thing you're going to do is something that's a little out there, like RESIDENT EVIL meets ZELDA. You'll try that thing, it sounds a little wacky, but then you've got $10 million left: What do you want to do? I want to spool up five wacky ideas. I want to go for broke. And I'm not willing to spend $10 million because the risk/reward is really not there. So I think about the concept of broadcast gaming, if we can create that channel. Now, I create just a level or two. That's pretty exciting because now I can put five hooks in the water and see where the fish are biting.

That's what TV does. TV didn't kill film. TV augmented film. How do we augment this $50, 50-hour, two-year in development experience with something that takes three or four months and a couple million bucks to pilot? You can go one of two ways, so you set those two hooks out, and you get some nibbles. One of those nibbles might say, "Hey let's go throw $2 million at it every six months and do something episodic, and it'll be more like TV." And maybe it's sponsored, or maybe it's $5 a month, or whatever. Or this other one says, "Hey, let's put this into full production. We'll go figure out how to put $10 or $15 million behind this thing because we know that this is a winner but it's more of an epic-like thing." So we get to test it. We don't have a test market in the console space right now.

So the [stage] I don't want to get to is people shipping half-assed games that they're going to finish after they ship. That's, you know, the bad spot. But I think there are two magical spots where I really want to focus some of our energy. The creative community is going to come up with something I haven't even thought of. And I love them for it.

GD: *Does Microsoft have any attitude toward handheld consoles?*

JA: Well, what's [Microsoft's] trajectory? I love our trajectory. What's served us so well in our trajectory? We've said the same thing, done the same thing for five years. We've listened to developers, we've enabled their imaginations, we're driving this online community, which we think is really, really important. We're focused on cost, we're focused on quality, we're focused on building a team, we're focused on the next generation, we're not distracted. So I'll never say never because someone could've asked Gates 10 years ago if he'd ever make a TV console: He could have said, "no," or he could have said, "I'll never say never." He didn't think he was going to at the time. Right now we're just saying, from our point of view, handheld would be a distraction. I think a handheld is, for any console leader, a distraction, potentially an expensive one. ❖

game developers choice awards

5th Annual Game Developers Choice Awards

Visit www.igda.org/awards to learn more.

Congratulations to the Recipients!

BEST GAME
Half-Life 2
Valve Software / Vivendi
Universal Games
*Ken Birdwell, Gabe Newell,
Jay Stelly*

AUDIO
Halo 2
Bungie Software / Microsoft
Game Studios
*C Paul Johnson, Marty O'Donnell,
Jay Weinland*

CHARACTER DESIGN
Half-Life 2
Valve Software / Vivendi
Universal Games
*Ted Backman, Dhabih Eng,
Bill Fletcher, Bill Van Buren*

GAME DESIGN
Katamari Damacy
Namco
Keita Takahashi

TECHNOLOGY
Half-Life 2
Valve Software / Vivendi
Universal Games
Yahn Bernier, Brian Jacobson

VISUAL ARTS
World of Warcraft
Blizzard Entertainment
*Sam Didier, William Petras,
Justin Thavirat*

WRITING
Half-Life 2
Valve Software / Vivendi
Universal Games
Marc Laidlaw

NEW STUDIO
Crytek
Far Cry
Avni Yerli, Cevat Yerli, Faruk Yerli

INNOVATION
Donkey Konga
Namco / Nintendo
Hiroshi Igarashi, Hiroyuki Onoda

I Love Bees
4orty2wo Entertainment /
Microsoft Game Studios
Elan Lee

Katamari Damacy
Namco
Keita Takahashi

LIFETIME ACHIEVEMENT
Eugene Jarvis: Legendary arcade
game developer

FIRST PENGUIN
Richard Allan Bartle, Ph.D:
Founding father of Multi-User
Dungeons (MUDs)

MAVERICK
*Matt Adams, Ju Row Farr
and Nick Tandavanitj:* Founders
and leaders of Blast Theory,
an internationally renowned
interactive media artists' group

COMMUNITY CONTRIBUTION
Sheri Graner Ray: Revered
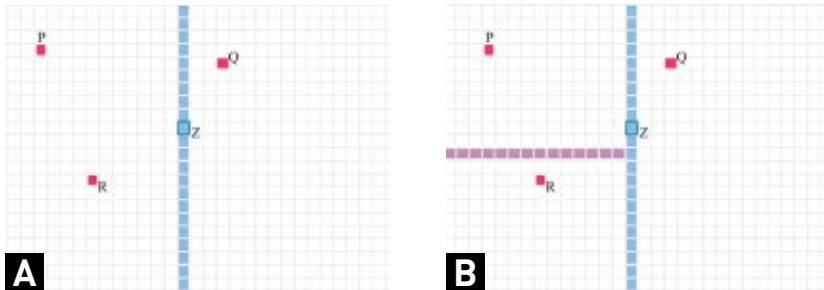advocate of women's interests
in game development

>> THE INNER PRODUCT

OPTIMIZING PATHFINDING V: PRECOMPUTATION

IN THIS SERIES I'M INVESTIGATING TILE-BASED pathfinding, a problem which the game industry generally treats as solved: just use A*.

Last month I demonstrated how it's possible that a more computationally-efficient algorithm than A* might exist for tile-based pathfinding, although it's pure speculation. As the pathfinder explores the map and discovers obstacles, there's enough information that it might use to more wisely guide the rest of its pathfinding.

Walking through the algorithm, thinking about how much work it does in discovering obstacles and drawing conclusions about them, it was obvious to me that the algorithm could do a lot better if it didn't even have to investigate to locate those obstacles, if it just knew all along because some data structure represented them. Knowing about obstacles in advance means making use of "precomputation." The pathfinders I've presented in previous columns allowed you to change the map after every pathfind, because they did not use precomputation. As soon as we introduce precomputation, comparing the performance of pathfinders that do and do not precompute will be tricky. Precomputing pathfinders are, in practice, solving a different problem since they cannot easily cope with the map changing. For now we'll ignore that, but next month I'll discuss how to fairly compare them for practical use in games.

GRAPH ALGORITHM PERFORMANCE

Algorithm text books measure performance of graph operations with big-O notation using the variables V for vertices and E for edges; for example, Dijkstra's single-source shortest paths algorithm using a binary heap runs in $O((V+E) \lg V)$ time. In pathfinding, the number of edges is usually proportional to the number of vertices. For tile-based pathfinding, E is no more than 4V, or 8V with diagonal edges.

SEAN BARRETT *develops independent games in Oakland, Calif., when he's not consulting in the game industry. Reach him at* **sbarrett@gdmag.com**.

Because of this strict proportion, we can simply use V everywhere in big-O formulas.

Since I'm working with large grids, it's sometimes more useful to think in terms of the grid dimensions. With a square grid of dimensions $n \times n$, $V=n^2$. We can also incorporate the length of the path, since certain algorithms are constrained by that length. Let's say the length is p.

Now let's look at graph algorithm performance in these terms. Breadth-first search takes $O(1)$ time per node, so exploring the entire graph takes $O(n^2)$ time. If we stop after finding a path of length p, we can't explore more than p^2 nodes, so computing a path of length p takes $O(\min[p^2, n^2])$ time. To allow edges with different weights, we have to use Dijkstra's algorithm, which takes $O(\min[p^2, n^2] \lg \min[p, n])$ time. Knowing the costs are bounded integers and using the multi-list priority queue (see "Optimizing Pathfinding Part I: Low-Level," December 2004) allows reducing the big-O time to that of breadth-first search. A* best-case performance is better than that of Dijkstra, but its worst case is the same, and our interest in precomputation arises from the cases where A* performs poorly.

ALL-PAIRS SHORTEST PATHS PRECOMPUTATION

The most obvious way of precomputing pathfinding is simply to precompute all possible paths. For any start node A and end node B, we can precompute the path from A to B. Assuming the average path length is $O(n)$, storing all these paths would require $O(n^5)$ storage. However, we don't need to store the entire path from A to B, just the first direction to move from A to B. If this leads us to C, we can then find the first step on the optimal path from C to B and use that. Because optimal paths are appropriately substructured (a section of an optimal path is always optimal), this suffices.

Another way we can store the precomputation information is to store the distance from the current node to the goal node; here, "distance" means the sum of weighted edge costs across the entire path. By starting at the end and working back, an

FIGURE 1 Discovering a path from the distances from a source; this example disallows diagonal movements. Starting from the 15 at the top right, an optimal path to there from the source can be found by repeatedly moving to any square with one lower distance than the current one.

optimal path is easily recovered, as shown in Figure 1. In the general case, one must take steps where the cost of the edge traversed is the difference between the total movement cost to the start for each of the two nodes.

In terms of big-O performance, storing the distance is generally superior. If we store directions and we want to know the distance, it takes us $O(p)$ time to determine it. If we store distances and want to know directions, it takes us $O(1)$ time to compute the direction at each step, and $O(p)$ time for the entire path. Storing directions can't do any better. Even if we stored the whole path, it would still take $O(p)$ time to do anything with it. In terms of practical storage costs, the difference can be significant. We can store the precomputed directions in 4 bits each (padded from 3), requiring $0.5 \times n^4$ bytes, whereas storing distances that fit in 2 bytes requires $2 \times n^4$ bytes—four times as much storage. Because the amount of storage we'll be considering here is quite large, a factor of 4 might be significant.

In this particular case, however, both are entirely impractical. For a 256×256 tile map (much smaller than what I've been tackling in previous columns) storing precomputed directions will require 2^{31} bytes, or 2GB. Spending 2GB on allowing paths on a

FIGURE 2A Any path from P to Q must cross the line of blue tiles in this divide-and-conquer strategy.
FIGURE 2B Any path from P to R must cross either the blue tiles or the purple tiles.

256×256 grid to be computed in $O(p)$ time hardly seems like a useful tradeoff.

Looking back at big-O notation, we can spend $O(n^4)$ space to get distance in $O(1)$ time and paths in $O(p)$ time. Or we can use Dijkstra with multi-lists and spend $O(n^2)$ space—map costs and the run-time search map—to get distance and paths in $O(\min[n^2, p^2])$ time. Since big-O notation described an upper bound, we can omit the min[] and still have a valid result. Table 1 makes use of this when summarizing these results, suggesting the possibility of an intermediate algorithm.

There's nothing specific to grids in the table, so perhaps there's an intermediate algorithm that works for arbitrary sparse graphs. To simplify things, though, I'm going to describe an algorithm that only works on grids. (There may be some way to generalize it to non-grid maps, but I don't know what it is.)

DIVIDE AND CONQUER
Consider finding a path from P to Q in Figure 2A. Because the path is a grid and there are no teleporters, any path we find from P to Q must step through the blue tiles at least once. That means the optimal path must cross the blue tiles at least once. Suppose the optimal path goes through the blue tile Z. Because optimal paths are optimally sub-structured, the optimal path from P to Z concatenated with the optimal path from Z to Q must be an optimal path from P to Q. Therefore, if we precompute all optimal paths from the blue tiles, we can determine

TABLE 1

Algorithm	O() space	O() time
Dijkstra	n^2	n^2
???	n^3	n^1
Fully precompute	n^4	n^0

Time and space required to compute distance using various precomputation strategies.

the optimal path between any point on the left of those tiles and any point on the right. If we precompute distances, we can find the optimal distance in $O(n)$ time: For each blue tile, it takes $O(1)$ time to sum the distance to P and the distance to Q, the optimal distance from P to Q through this tile. We then take the maximum through all the blue tiles. The storage required is $O(n^2)$ per blue tile, or $O(n^3)$ total.

This doesn't handle the case if the tiles *aren't* on opposite sides of the line of blue tiles, but we can divide and conquer, as shown in Figure 2B. We recursively split each half, adding a new line of precomputed nodes down the center. These nodes only store precomputed distances for the nodes on their side of the previous partition. To find the distance between P and R, we have to find the optimal distance that stays within the left half of the grid and passes through the purple tiles and the optimal distance that crosses the blue tiles. The lowest of those is optimal.

Recursing down like this results in each tile on the board being precomputed for $2n$ "middle" tiles, so the total storage cost is $O(n^3)$ and each requires $2n$ comparisons and tests. So $O(n)$ total time is required to compute the distance, nicely filling in the gap in Table 1. However, this recursive structure is actually clumsy and difficult to process, and it's complicated to deal tiles which are "normal" at some levels of recursion and "boundary" at others. It doesn't cope well with map layouts that aren't exactly the ideal size, which turns out to be one less than a power of two. For that reason, I've come up with a more straightforward approach that's easier to measure.

REGULAR GRID SPACING
Figure 3 shows a different way of structuring the grid to allow precomputed pathfinding in $O(n^3)$ storage and $O(n)$ time. Rather that recursively subdividing, we simply place a grid of "precomputed boundary tiles" across the entire map. Although the figure shows them placed every fifth tile, the ideal layout is to have the number of rows and columns of boundary tiles be the square-root of n, that is $n^{0.5}$, or $2n^{1.5}$ boundary tiles. (This

double-counts tiles that are in both the boundary rows and boundary columns, but there are only n of them.) It also means the spacing between each row is also $n^{0.5}$, so each block of tiles in between the boundary tiles is $n^{0.5}\times n^{0.5}=n$ tiles.

Each boundary tile stores the precomputed distance for all the regular tiles in the adjacent blocks. This amounts to $2n$ tiles for each boundary tile, which, at 2 bytes apiece, requires $2n^{1.5}\times2n\times2$ or $8n^{2.5}$ bytes. Additionally, each boundary tiles stores the precomputed distance to all other boundary tiles, which requires $8n^3$ bytes, again assuming 2-byte distances.

To compute the distance between two points such as P and Q, we consider all the boundary tiles surrounding each of P and Q. For a given pair, such as Y and Z, we have precomputed the distances from P to Y, from Y to Z, and from Z to Q. The total distance of the best path from P to Q through Y and Z is thus the sum of these three. The minimum distance from P to Q is thus the minimum distance for all Ys and Zs in the boundaries surrounding P and Q. There are $4n^{0.5}$ tiles surrounding each of them, so the cost of computing the distance for all pairs is $4n^{0.5}\times4n^{0.5}$, or $16n$.

We can easily recover the path from P to Y and from Z to Q since we store the distances from Y to P's neighbors as well, so we can "follow the distance gradients." From Y to Z, however, we have no information. If we store the next boundary tile on the optimal route from Y to Z, though, we can then use that to follow the distance gradients as well. This doubles the storage cost to $16n^3$ (dropping the smaller $8n^{2.5}$ term) assuming 2-byte distances and 2-byte boundary tile ids.

Unfortunately, this is still too much storage. For a 256×256 tile map $n=256$, so the storage required is $2^{28}+2^{23}$, or about 256MB. That's maybe right at the limit of what you might afford in a next-generation PC game with the highest priority being the fastest possible perfectly optimal paths, but it's infeasible elsewhere. At 64×64, this technique would require 4MB, which is possible although the cost of building the table at run-time would still be prohibitive, and the overhead hidden by the big-O discussion might be significant at that scale.

Two notes about this algorithm: First, the above description can't find the path between two tiles that are in the same block. If we simply do a Dijkstra search in the block, we can spend $O(n)$ more time to find a direct path if there is one. Second, we don't actually have to space the boundary tiles by $n^{0.5}$; that spacing achieves optimal big-O behavior, but changing the spacing allows useful tradeoffs between real performance and storage usage.

FIGURE 3 Using a fixed grid of tiles allows a non-recursive strategy for reducing precomputed storage by storing the distances between all blue tiles, such as Y and Z, as well as distances between blue tiles and nearby white tiles, such as P and Y or Q and Z.

GOING HIERARCHICAL

We can entirely dispense with precomputing the smaller chunk of data—from the boundary tiles to the normal tiles in their neighboring non-boundary blocks—by doing a small search from both P and Q. This will compute the distances from P to all its adjacent boundary tiles and from Q to all its adjacent boundary tiles. (You'd actually search from Q using reversed edge costs.) Each of these operations takes $O(n)$ time, since there are only n tiles per block.

However, that only computes distances; to find the path, we need to "gradient follow" on the distances, and we don't have that data between the boundary tiles anymore. Since we do know which boundary tile is next, we can do a local pathfind within each consecutive block to find it. In the worst case, this local pathfind requires n steps (filling the block). If we revisit a block multiple times (zig-zagging along a border), I'm pretty sure you don't explore more than n nodes total if you use Dijkstra's algorithm. Given a path of length p, we cannot visit more than $3p/n^{0.5}$ blocks, so we'll spend at most $O(p \times n^{0.5})$ time doing these local pathfinds. This is worse than $O(p)$ and worse than $O(n)$, but is not too bad in practice. For a $1,024 \times 1,024$ map $n^{0.5}$ is only 32.

The pathfinder sketched above is essentially hierarchical. The "boundary tiles" are high-level nodes for hierarchical pathfinding, but rather than do a normal pathfind on them, as hierarchical pathfinding would, we simply fully precompute all pairs shortest-paths for them. The central problem with this algorithm is that storing the distances between all boundary tiles requires a lot of memory and time to compute. Recomputing after map changes would be expensive. If we instead only stored the distances between boundary tiles that border the same blocks, we'd only be precomputing information that was relatively local (one block apart) and we'd only need to store $O(n^{2.5})$ information. To solve a complete pathfind, we'd have to do pathfinding on the boundary tiles, using these local distances as edge weights.

Unfortunately, there are far too many edges for this to be effective as hierarchical pathfinding; each boundary tile has $7n^{0.5}$ neighbors, so a $1,024 \times 1,024$ map would have 448 edges per boundary node, which would not lead to efficient pathfinding.

If we "lossily" simplify the map, combining boundary nodes together, we can reduce the number of boundary nodes and edges significantly. Next month I'll look at the HPA* algorithm published by Botea, Muller, and Schaeffer's in 2004, which does exactly that. ✖

SIGGRAPH 2005

The smart place to be. You'll find all the data, techniques, people, and inspiration you need for another successful year of research, creativity, development, and production. **COURSES:** Upgrade your knowledge. Extend your warranty against career obsolescence. **COMPUTER ANIMATION FESTIVAL, ART GALLERY:** Encounter extraordinary images and learn from the minds that made them. **EMERGING TECHNOLOGIES, GUERILLA STUDIO:** Optimize your real-time interaction with digital machines. **EXHIBITION, RECEPTION:** Interact with an astonishing accumulation of A-list minds in the world capital of entertainment and innovation. **PAPERS, PANELS, POSTERS, EDUCATORS PROGRAM, SKETCHES, WEB PROGRAM:** Join the industry's best and brightest stars as they illuminate engineering and art, animations and equations, allegories and algorithms. Five Days Only! 31 July through 4 August 2005. For conference details:

www.siggraph.org/s2005

bring your brain

Conference 31 July-4 August 2005   Exhibition 2-4 August 2005   Los Angeles Convention Center
The World's Leading Marketplace of Computer Graphics and Interactive Techniques

Sponsored by ACM**SIGGRAPH**

SIGGRAPH2005

» PIXEL PUSHER

READ THE DARN AD!

OR, "HOW I STOPPED WORRYING AND LEARNED TO LOVE THE COVER LETTER"

Why browse your local bookstore for bad advice on how to get a job when you can get all of that, and more, right here?

AHHH! IT'S SPRING, AND THE AIR IS FULL of new life. Everything old is new again-particularly the soundtracks (and the appetizers) at those E3 parties. Everywhere you look, studios flush with greenlight funding are strutting their plumage before a new crop of art school grads, hoping to entice them into an intricate mating dance. Birds do it, bees do it, even Hollywood TDs do it! So let's do it—let's talk about job hunting.

You can cut down the unpleasantness in job hunting pretty quickly if you learn the two simple rules that govern how employers approach the hiring process. It's not much to learn. All you need to remember is 1) hiring is painful and 2) companies are desperate and pathetic.

RULE 1: HIRING IS PAINFUL
Hiring is a slow, expensive, and risky proposition. Think about it from the perspective of a company that has a slot to fill. In order to get one person, the human resources department and hiring managers need to spend a couple of weeks contacting various web sites and magazines, writing up a set of ads that adequately describe the exact job you're trying to fill—assuming management and production agree on what that is! They also spend at least a few hundred bucks on each ad.

Once all this money and time has been laid out, hundreds of applications come tumbling in, and 95 percent of them are from people who don't have any of the qualifications specified in the job ad. Nevertheless, the company needs to dedicate one staff person to a few weeks' worth of sifting through resumes and portfolios of old school work, 3D package tutorial images, and scanned-in charcoals from figure classes. When the really good candidates finally emerge from the dreck, half of them will already be hired elsewhere. To top it all off, after you've gone to all this trouble to find the right candidate, you actually have to pay them salaries—and there's probably a recruiter, a moving company, or an immigration lawyer lurking in the background with a bill as well. It's really not hard to see why companies hate hiring.

RULE 2: COMPANIES ARE DESPERATE AND PATHETIC
As horrible as hiring can be for companies, it's also an unavoidable necessity. You can't make a game without people, so you really need to find them and quickly. The power of networking is a byproduct of how adamantly companies dislike the ordinary over-the-counter hiring process. Bringing aboard that old friend or former colleague means skipping the most laborious and risky parts of the hiring process, so naturally it's popular with firms that are in a hurry to hire. We all know how this can stack the deck against folks without pre-existing industry connections.

Luckily for outsiders (and unfortunately for the potential employers) no company can fill every vacancy from the ranks of old friends and former co-workers. Companies with vacancies are constantly aware that every day an opening stays vacant is a day lost, possibly many days, if the empty seat is an important one. So any employer will eventually turn to the public forums, web sites, and recruiting firms where job-seekers congregate. It's tough for the companies because it implicates them in all the unpleasant stuff, such as the weeding through irrelevant resumes. But it's good news for the hopeful job-seeker. By the time a job posting hits the boards, the company is absolutely committed to filling that empty chair.

LIVING BY THE RULES
We all know that it's not just the companies who suffer. Job-hunting, particularly when you're out of work, is a morass of frustration and uncertainty. The injustice of it all can easily overwhelm you. You know how good you are, how passionate you are about games, and what great work you can do. The nonsense generated by the process—the resumes and cover letters, the formula interview questions, and of course the dreaded issue of prior experience—eventually starts to seem like a vast shadowy conspiracy directed right at you. "I know I can do the job!" you want to scream, "Let's just cut to the chase!"

The sad truth, though, is that the miseries inflicted on job seekers aren't just random cruelties. They are logical byproducts of the dilemmas faced by potential employers. Companies hate hiring, and at the same time they're desperate for people. The rituals of

STEVE THEODORE *is a 10 year veteran of the games industry. He currently works at Zipper Interactive, where he's been sorting through lots of resumes. It shows. Email Steve at* **stheodore@gdmag.com**.

the job hunt have evolved out of these conflicting demands.

Figuring out why employers set you to jumping through a particular set of hoops is very powerful knowledge that can help you upgrade your own job application. It's also a good way to keep yourself focused when the emotional drain of job hunting gets you down.

MYSTERIES OF THE COVER LETTER
Seeing the hiring process from the employer's perspective can take some of the mystery (and misery) out of job hunting. Take the tradition of the cover letter as an example. It's easy for a job hunter to see the cover letter as a meaningless formality. After all, having an uncanny talent for writing cover letters is hardly a guarantee of artistic or technical skill. The ability to search for the name of the last company you applied to and replace it with the name of the next hardly betokens artistic brilliance or technical savvy. Moreover, artists aren't famous for their verbal skills. Don't forget either that many talented folks don't speak English as a first language. So why do we bother with cover letters? Or, more to the point, why do the companies bother? After all, if you think cover letters are a meaningless formality, just imagine what the HR person on the other end goes through, sorting through them by the dozen.

In fact, the cover letter makes sense when seen against the backdrop of the two basic rules of hiring. The screener or HR person or producer who reads your application is trapped between two conflicting desires: to find a great candidate, and to dispose of all the no-hope candidates as efficiently as possible. The cover letter helps the screener in both directions.

On the positive side, a good cover letter lets the screener know that you've actually paid some attention to the requirements of the job at hand. It may sound strange, but proving that you have read the ad is a surprisingly important matter. Until you've been tasked with screening resumes yourself, you literally cannot believe how few applicants pay attention to the carefully thought-out, diplomatically worded prose that describes a job on offer. Place an ad for a concept artist, and 30 to 40 percent of the respondents will be modelers or level designer or animators. Post a job requiring two shipped titles, and half the respondents will be in their senior year at art school. Against this backdrop, a decent cover letter which clearly indicates that you've read and

understood the job requirements automatically tells the screener that you're in the top half of the incoming wave of resumes. Even if you don't formally meet every last detail of the criteria, using your cover letter to explain why you'd be good for the job anyway probably will get you a free pass on the first cut. The key thing is to tell the screener simply and clearly that you know what job is being offered and that you have the professional chops and personal drive to do that job.

Which brings us to the second reason cover letters exist. Remember the first rule of this column: Companies hate hiring. No matter how badly a company wants to fill that slot, most of the applicants for the slot will be completely unqualified. The employer wants to get to the handful of good candidates quickly, and that means dispensing with the bad ones as fast as possible. In this context, a cover letter offers a chance to fail—in other words, a fast way for the screener to cull out weak candidates. If you can't put together two reasonable paragraphs that tell the screener 1) you've read the ad and 2) you'd make a good match for the job on offer, your application is headed for the circular file.

It's not strictly true that a bad cover letter will sink you right away because most screeners are conscientious enough (and desperate enough to find that mythical good candidate) that they'll still check out the resume and reel. But make no mistake, if you blow the cover letter stage of the process you're set up for elimination in the next round. Anything else is a comeback.

THE AWFUL TRUTH
"Wait a minute," you're thinking, "Did he really mean to say that a stupid cover letter means more than my demos?" Well, no. No artist gets a

Dear Job Applicant,

Thank you for applying to the position we advertised. While we specifically requested only applications from candidates having four or more years of industry experience, we understand that you might not have realized which industry we intended. However we are impressed by your entrepreneurial drive and are sure that the skills you learned behind the cash register can contribute greatly to our team environment. We are somewhat concerned, however, that you may have mistaken our ad for one from another company. Since none of the four identical resumes and cover letters we received from you mentioned our company or which position you were applying for, we wanted to contact you to be sure that you were, in fact, responding to one of our ads. My assistant and I spent quite some time puzzling over this and were unable to reach a conclusion, so we thought it best to contact you and see if we could figure out what exactly you intended. After all, we receive dozens of applications a week; what excuse could we have for giving each one less than our undivided attention?

In any case, we also have a more personal motive for contacting you. We really wanted to thank you for introducing us to the fascinating world of post-industrial Thrashtronica, a musical style we had never even heard of around the office until we went out and bought a VHS player to view your samples. The nightmarish soundscape certainly taught us a few things about the meaningless void at the heart of modern capitalism! It's a good thing the turntable animations on your tape were 15 minutes long, so we were able to hear the entire piece.

Sincerely,
Big Boy Games, EOE

job with a terrible reel and a great cover letter. But don't let that fact make you forget that rule No. 1 is lurking in the background at every stage of your job application. The person sorting and screening job applications, no matter how nice and open minded he or she may be, spends most of the day looking for reasons not to spend time with a given resume or reel. To put it more succinctly, most of the screener's job is figuring out who *not* to hire. Against this context, the cover letter, and then the resume, and finally and most ultimately the demo, are all up against a pretty stark test.

It's always sudden death overtime in the hiring business. It's not because any rational person thinks a cover letter, or even a well formatted resume in .doc format, is an indication of artistic talent; it's because there are a lot of resumes and not much time, so every niggling detail can become an excuse to move on to the next item in the inbox.

If this sounds brutal, that's because it is—but it's purely impersonal. It's a fact of modern life that can be managed if you understand it and work with it. Above all, nobody is exempt, not even the greatest talents among us.

Next month I'll finish the discussion of job hunting by looking at how resumes and demo reels function in the hiring process. In the meantime, polish up those cover letters. And please, read the damn ad! ✷

S. GREGORY BOYD

>> BUSINESS LEVEL

WHY PRIOR ART MATTERS

The selfish reasons to save our interactive entertainment history.

One of many early PONG arcade clones.

SURE, GAME HISTORY IS CULTURALLY important, but what if I only care about money? Even if you ignore the cultural and educational benefits, preservation of game history—including source code, functioning game systems, and executable files—is very important. To wit, it can save the industry significant legal fees and settlement costs. This saving is dependent, though, on recognizing the value of this material beyond those aforementioned educational and cultural benefits.

Games are an important part of our culture. Each shipped game freezes in time our technological capabilities and what many of us value in entertainment. Games are also snapshot representations blending our technological advances and artistic sensibilities. For these and other so-called high-brow reasons, games are worth saving. That may not be enough to inspire game developers and publishers to contribute to game preservation efforts, however.

THE GOOD OLD DAYS

First, a little needs to be said about patents in the game industry and the litigation that inevitably follows. Patents are offensive instruments. Contrary to popular belief, a patent does not give a company the right to sell an invention; instead, a patent prevents others from selling any product or system that falls within the claims of the patented invention. Patent holders often use litigation or the threat of litigation to force competitors to stop selling infringing goods, or more commonly, to pay a license fee.

A lament I often hear as a patent

attorney is how great the "good old days" were before there was patent litigation in the game industry. But there was never such a time. Starting with PONG and for every year afterward, the game industry has had to deal with patent litigation. Ralph Baer filed the application that became U.S. Patent 3,728,480 in 1973. Baer's patent titled "Television Gaming and Training Apparatus" included prior art later licensed for Atari's PONG. Recognizing the patent's ground-breaking status, a court later wrote, "Evidence now of record reveals the phenomenon of a patent that heralded the beginning of an industry, the home video game." Not surprisingly, Baer's patent and several related patents were involved in various litigations between 1976 and 1982. Magnavox was able to collect $18.5 million in royalty payments from the patent.

After we accept that patent litigation has been and probably will always be with us as an industry, we need to plan for ways to make it as efficient and painless as possible. One key to efficiency lies in creating a usable body of prior art, which is in turn tightly linked to the preservation of game history.

LITIGATION, PRESERVATION

Roughly speaking, there are two ways to win a patent litigation from the perspective of the defendant accused infringer. The first way is for the developer to show that it does not infringe upon the patent. The second important defense is for the accused infringer to show that the patent is invalid by prior art. In other words, the defendant uses prior art to prove that the patent is not new and not deserving of patent protection.

Prior art is material that existed before the patent was filed. Preferably, this material was publicly sold or used. In the game industry, prior art typically includes source code, working game systems, manuals, and executable game files. Prior art can also come from other industries, but the closer the art is to the accused game invention, the better.

Without getting into too much legal jargon, it's easier to win using the "we don't do that" arguments than it is to prove that the patent is invalid. Invalidating a patent through prior art is a much more devastating way to win because with noninfringement cases, the patent still stands to fight another day. With invalidity through prior art, the patent is no longer viable and cannot be used against anyone else. A solid preservation system could make finding prior art and showing that shoddy patents are invalid easier.

THE HARD WAY

Currently, attorneys have to hire industry experts to help search for prior art. Attorneys and experts search databases like Mobygames.com to find leads for prior art that beats the filing date of an asserted patent. In addition, experts and attorneys comb through the academic literature to find descriptions of projects that may be able to invalidate the patent. After they have the leads, they still need working examples of the games, thorough descriptions of the games, and preferably source code. Often, they have to track down employees of defunct companies and have them search old hard drives for pieces of source code or find old game executable files through retro gaming bulletin boards. As you can imagine, the burden and costs of these efforts can be significant.

THE EASIER WAY

Even if solid preservation doesn't have a preventative effect, it should make finding prior art cheaper and therefore help make litigations cheaper, and aid cases toward finding an efficient resolution. Hopefully, it will also help invalidate patents that should never have been issued. Patent litigations cost money and can take years to resolve; a losing verdict in a patent litigation, even in the game industry, can cost the defendant tens of millions of dollars. ✖

THE PONG PATENT

Two cases over PONG: Magnavox Co. v. Chicago Dynamic Industries, 201 U.S.P.Q. 25 (D.Ill. 1977) and Magnavox Co. v. Mattel, Inc., 216 U.S.P.Q. 28 (D.Ill. 1982). In both these cases, the patent survived attacks on its validity.

S. GREGORY BOYD, *is an attorney with Kenyon & Kenyon in New York. He is currently involved in several patent litigations, including the American Video Graphics case. He is also a member of the IGDA IP Rights Committee. Email him at gboyd@gdmag.com.*

ALEXANDER BRANDON

» AURAL FIXATION

THE LINE OF QUALITY PART III: INTEGRATION

WE'RE ROUNDING OUT "THE LINE OF Quality" series with something that is perhaps in the greatest state of flux right now in game audio: the integration of sound effects, music, and voice. Since we discussed the integration of tools in the last column, we'll now discuss the workflow involved in using these tools, who should do the integration, and how it should be done.

Though you might shudder to think this, nearly 40 percent of developers still employ programmers and level designers to do sound integration work (in fact, some publishers even employ full-time audio tools programmers), and only 30 percent of developers spend any time doing actual integration design. In that 30 percent, you'll find the likes of the top brass publishers, including EA and Sony. The reason they invest in this is because they want top quality for their game sound, and they want it done quickly.

INTE-GREATNESS
There are several good reasons why integration needs to be in the hands of audio specialists, and why integration design is crucial.

Clearer goals. By allowing an audio specialist to examine design documents early on in the development process, sounds, voice over, and music can be set to a more focused specification based on the needs of the title, thus speeding up the entire process. Take a description of a non-player character (NPC), for example. If the description includes the actions of the NPC and its relationship via AI to the player as well as the world,

the audio integrator can create a spec sheet for the sounds required and schedule the work needed for a more accurate schedule that can meet a ship date, rather than just guesstimate.

Specialization. While it still isn't necessarily cheap, specialists for audio integration will usually provide less expensive and more reliable audio going into a game than a programmer doing the job. A programmer that integrates audio will certainly have other tasks in hand, and with today's production requirements (such as the audio being mixed at the proper volume and using the proper filters and ducking/submix groups), overlooking these requirements yields a more obvious breach in quality than it did five years ago. So, save money and get better quality. Oh yeah, look at that first reason again—you can do it faster, too.

Parallels to other industries. In big budget releases, the film industry always uses specialist audio engineers for audio mixing for at least a few months in postproduction. Picture a gaffer doing the job of one of these audio engineers, or a lighting engineer pushing Ben Burtt (the *Star Wars* sound designer) out of his chair to mix lightsaber sounds. Not a good idea.

A pioneering example of a sound specialist integrating sound to great success is Eric Brosius' work in the original Thief. He had control (within the limitations of the hardware) of the sounds, although he used text files for quite a few of these controls. Nevertheless, he could manipulate sound data such as volume, pitch, occlusion, and propagation with a reasonable auditioning pipeline, and the result was phenomenal. Regardless of whether the title you're working on has sound as a key design mechanic, the results will be obvious with a solid engineer designing, integrating, and mixing.

AN AURAL FIX
Comparing methods for audio integration, you'll see that doing the job in real time costs nearly half of what it would to do it in non-real time—and the quality will be much greater.

This is something to think about for your next project, which might help ease the transition from programmer to audio integrator, aside from the potential savings. I've used hypothetical numbers for salaries to calculate the savings, but they're not far from the real thing.

Having tools that can assist an integrator mixing volume in real time is becoming pretty standard now, though it's not necessarily used often. XACT and SCREAM, though proprietary to Xbox and PlayStation 2 respectively, have functionality that allows this real-time manipulation, and next-gen tools will make it even simpler to integrate at the outset.

KNOW YOUR ROLE
When non-specialists such as programmers have more responsibility for sound, it ends up being lower in quality with bad mixes (overly loud voice or music, etc.) and bad transitions (cut-off music and voice rather than a smooth fade). Imagine if the roles were reversed—most sound designers wouldn't know where to begin with programming. This is often the case with a programmer attempting to integrate and mix the sound themselves.

The programming time is far more valuable in both methods we've listed, but you spend much more in production costs on the back end. Even more valuable, your audio staff doesn't complain, and the quality of the game's sound engineering is not an issue anymore. Now go sell two million copies and win a few awards. ⋉

ALEXANDER BRANDON *has been involved with game audio since 1994 and is currently the audio manager at Midway in San Diego, Calif. You can email him at* abrandon@gdmag.com.

NOAH FALSTEIN

» GAME SHUI

THE WRIGHT STUFF

Will Wright at the Game Developers Conference 2005.

AS I WRITE THIS COLUMN MY MIND IS still reeling from the 2005 Game Developers Conference. Until the last day, I thought my favorite session was the "Emily Dickinson Challenge" in which Clint Hocking, Peter Molyneux, and Will Wright each presented their approach to how they would make a game based on the poet's works.

Then, in a separate session, Wright managed to top himself by introducing SPORE, easily the most impressive game preview I've seen at GDC in at least five years—and one of my all-time favorite presentations in the 17 years I've been attending the conference. I haven't been this excited about an unreleased game in decades.

SPORE is the working title of Wright's next big achievement, a genre-spanning hydra of a game in which the player starts as a micro-organism in a pond and evolves through the eons to eventually become an interstellar traveler with world-transforming power.

Now, however, E3 is upon us and with it, likely the formal announcement of the game. Major game news sites will undoubtedly cover the nuts-and-bolts, so here I hope instead to focus on why SPORE's underlying design innovations make it such an impressive concept, even at this early stage of its development.

DARWIN RULES!

First, as I've observed and stated in the past (see "In the Beginning," March 2005), the best games are based on Darwinian themes of survival and reproduction, hunting and gathering, individual and social success. It's hard to imagine a game that gets closer to these themes than SPORE.

THE SPORE TRIUMVIRATE

Also remarkable are the three underlying concepts in the game, each of which might seem risky in isolation but, like the legs of a tripod, support each other to form an elegantly stable system, the weaknesses of each innovation buttressed by the strengths of the others. The three concepts at work are:

1. Utilizing user-developed and shared content
2. Mixing multiple game genres
3. Playing up a fanciful core gameplay mechanism—creating and evolving beings, buildings, cities, and vehicles.

USER-MADE CONTENT

The quantity and quality of user-developed content is notoriously irregular across games. Only perhaps 10 percent of players really like to generate their own content, and only 10 percent or so of them are any good at it. But the potential pitfall of not enough content is balanced by the high compressibility of that content, allowing SPORE to ship with thousands of pre-generated elements from the developers and the inevitable early testers. The varying quality of content is perfectly balanced by a variety of gameplay styles. If one player is bad at designing creatures, perhaps he or she will be great at forming cities.

SPORE is even more balanced by the evolution-influenced gameplay. Weak or ineffective beasts designed by others can add to the fun by becoming easy conquests (and meals) for the player's creature.

MULTIPLE GAMEPLAY TYPES

Conventional wisdom says games must focus on a single, clear theme and not mix together many different types of gameplay. But the three game mechanisms used in SPORE—evolutionary change, competition, and reproduction—unify the game rather than clutter it because all three of those game genres are actually related under the skin.

Another concern might be that including multiple gameplay styles will escalate the cost of the game. However, Wright commented during his talk that using player/tester-created content in developing SPORE helped to keep costs down and minimized the initial financial risk as a result.

EVOLUTIONARY THEME

The third risk Wright took in making SPORE was to ground the gameplay in the fanciful, perhaps seemingly too cerebral theme of evolutionary change. The argument is that in his previous games, the more cerebral or fanciful ones like SIM ANT, SIM EARTH, and SIM LIFE, had only limited appeal, while everyday themes of SIM CITY and THE SIMS were widely popular. Other hit titles these days like sports games or GTA: SAN ANDREAS are also based in the real world. That makes it easier for players to identify with the characters and to bring their own real-world goals into the game.

But let's not underestimate people's ability to find human motives and behavior even in abstract or iconic representations. Once again, the other elements of SPORE support this: Having players create their own creatures instead of just using pre-defined ones assures that the players will identify with their characters. It's not "a strange creature" one controls; it is "*my* Blerk that I raised from a microbe."

By offering a variety of play styles at different levels, SPORE invites all players to find a style in tune with their personal goals or tastes. That technique of pleasing a variety of players by packaging together different gameplay alternatives is commonly seen in modern MMORPGs, and it's one I know first-hand from my design days with Hal Barwood while making INDIANA JONES AND THE FATE OF ATLANTIS.

Finally, I can't help but observe the appropriate name, since radically innovative games like SPORE only appear at unpredictable intervals of time—sporadically! ✷

NOAH FALSTEIN *is a 25-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. Email him at* **nfalstein@gdmag.com**.

GAME ON

NOW HIRING PEOPLE WITH TECHNICAL AND ARTISTIC VISION

rainbowstudios.com

relic.com

volition-inc.com

crankypantsgames.com

heavy-iron.com

helixe.com

pcpandl.com

concretegames.com

thq.com

bluetongue.com

rainbow STUDIOS

relic ENTERTAINMENT

volition inc®

crankypants games

HEAVY IRON STUDIOS

helixe

PACIFIC COAST POWER & LIGHT CO.

concrete games

thq studio australia

bluetongue.

THQ

THQ Studios... there's a place for you.

Rockstar Games is actively seeking experts in the following fields:

Rockstar North
Animators
Character Artists
Environment Artists
Vehicle Artists
Game Programmers
Level Designers

Rockstar San Diego
Technical Artist
Character Animator
Networking Programmer
Tools Programmer

Rockstar Vienna
Graphics Programmer
Tools Programmer
Level Artist
Level Designer
Character Artist

Rockstar Vancouver
Art Director
Character Modeler
Scripters
Game Programmers

Rockstar NYC
no current openings

Rockstar Toronto
no current openings

Rockstar Leeds
no current openings

Headquartered in Chicago with studios in San Diego, CA, Seattle, WA, Austin, TX, Los Angeles, CA & Chicago, IL and offices in Munich, Germany and London, UK, Midway Games Inc. is a leading developer and publisher of interactive entertainment software for all major video game systems.

Midway has released some of the most popular games in the video game industry for play on all major video game platforms including PS2, Xbox and GameCube. These games have entertained millions of players around the world.

We are on the lookout for the best and brightest. Our goal is to create the most compelling, high-quality games around. We're always searching for world-class talent with energy, commitment to excellence and unrestricted imaginations to join our exceptional teams.

WE HAVE IMMEDIATE OPENINGS FOR:

- **Gameplay Programmers**
- **Network Programmers**
- **Graphics Programmers**
- **Software Engineers**
- **PS2 Programmers**
- **Environmental Artists**
- **FX Artists**
- **Concept Artists**
- **Animators**
- **Development Directors**
- **Creative Directors**
- **Technical Directors**
- **Producers**
- **Audio Manager/ Senior Sound Designers**

We have openings on many different game teams and genres including RPG, Action Adventure, Racing, Horror, Sports, etc.

For more detailed information on available positions at Midway, please visit our website:

http://jobs.midway.com.

Artists: Providing samples of your work is essential. Please first apply with your resume online, then send sample art/demo reel with a breakdown sheet detailing your contributions to the Midway location you applied to.

Midway Locations:

- **Chicago, IL**
- **San Diego, CA**
- **Seattle, WA**
- **Munich, Germany**
- **Austin, TX**
- **Los Angeles, CA**
- **London, UK**

Benefit Highlights
- Competitive Salaries
- Competitive Bonus/Incentive Programs
- Medical/Dental/Vision/LTD/Life
- 401(k)
- Vacation/Holidays
- Referral Bonus Program

MIDWAY

NEVERSOFT

HAVE A DRINK ON US

CONFERENCE HALL BAR @ THE SOUTH HALL - NOON, TIL WE CLOSE THE BAR
WEDNESDAY, MAY 18TH. E3.

COGSWELL

The Fusion of
ART & ENGINEERING

Bachelor of Art & Science Programs:
- * Digital Art & Animation
- * Digital Motion Picture
- * Digital Audio Technology
- * Digital Arts Engineering
- * Electrical Engineering
- * Software Engineering

WASC Accredited
Financial Aid Available
Small Class Size
Tours Avaliable

ACCEPTING APPLICATIONS!
CALL NOW!

COGSWELL
POLYTECHNICAL COLLEGE

1175 Bordeaux Drive
Sunnyvale, California 94089
(408) 541-0100

www.cogswell.edu

ADVERTISER INDEX

PANDEMIC/THQ'S
DESTROY ALL HUMANS!

Artists: Pandemic Australia art team

Bink takes the grind out of game video!

The video codec for games

Bink has shipped in 2,500 games for a reason. It is the standard for video in games - it is faster (up to 4 times faster), it uses less memory (up to 18 MB less), it has a built-in audio codec, it supports every platform, multiple audio tracks, data interleaving, alpha and RLA files (for Z-depth, normal and uv per-pixel data), and much more! Now supports HD video on Xbox!

It's like having your own codec

Using Bink is like using a video codec you wrote yourself. You can, for example, play to the screen, to a 3D texture, to a back buffer, to an overlay, to a plain old chunk of memory, or whatever. Bink works the way you want it to.

RAD knows games

We have been selling middleware solutions since 1991. We can solve the problems you face, because we (or our 2,000 previous customers) have had to solve them too!

New! Now shipping for Nintendo DS and HD video for Xbox! New Japanese Manual!

For more information:
425.893.4300
www.radgametools.com