CMP
United Business Media

PRODUCT REVIEWS LUXOLOGY'S MODO 202 DARWIN DIMENSIONS' EVOLVER

NOVEMBER 2006

game developer

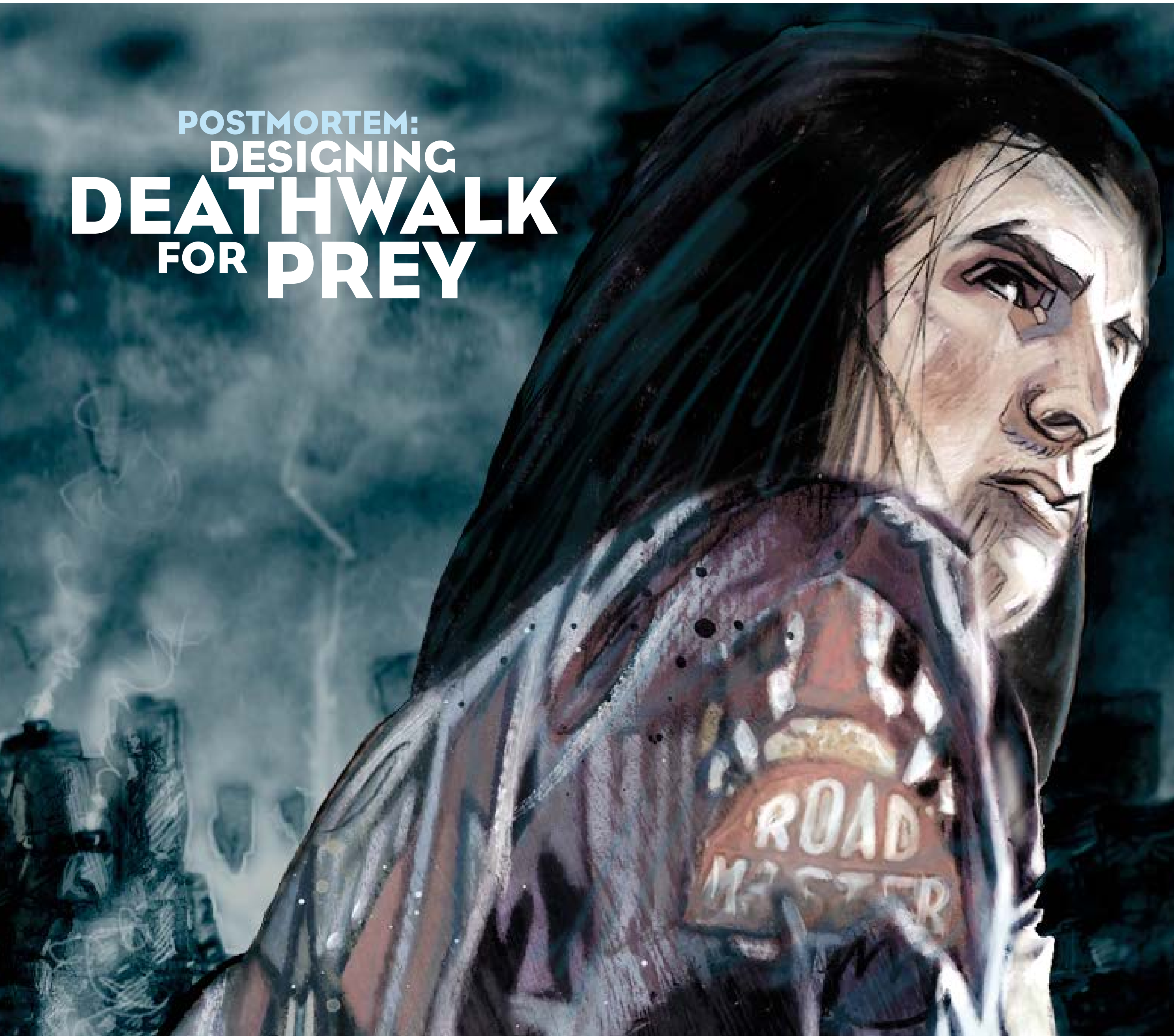
THE LEADING GAME INDUSTRY MAGAZINE

LUCK IN GAMES
RICHARD GARFIELD ON RANDOMIZING FUN

SURFACE TENSION
PHYSICS TO FLOAT YOUR GAME'S BOAT

LOOKING ASKANCE
ARTISTS: FEAR NOT THE RISE OF THE SCANNERS

POSTMORTEM: DESIGNING DEATHWALK FOR PREY



gamedeveloper FRONTLINE FINALIST '05

"ReplayDIRECTOR is one of the best tools I've found for our test department here at Crystal Dynamics."

***Chris Bruno**, QA Manager, Eidos Interactive*

BUGS ON ICE

RECORD. REPLAY. FIXED.

ReplayDIRECTOR™ gives you **Deep Recording.** This is much **more than just video capture**. Replay records every line of code that you execute and guarantees that it will Replay with the same path of execution through your code. Every time. Instantly Replay any bug you can find. Seriously.

DEEP RECORDING. NO SOURCE MODS.

download today at www.replaysolutions.com
email us at info@replaysolutions.com

REPLAY SOLUTIONS
1065 E. Hillsdale Blvd. Suite 307 Foster City, CA, 94404 - Tel: 650-472-2208 Fax: 650-472-2210

CRYSTAL DYNAMICS eidos XBOX 360 nVIDIA. XBOX

gamedeveloper

CONTENTS

NOVEMBER 2006
VOLUME 13, NUMBER 10

FEATURES

POSTMORTEM

DEPARTMENTS

COVER ART BY HUMAN HEAD STUDIOS

COLUMNS

GAME PLAN

RIGHT OF REPLY

IT'S BEEN MORE THAN 10 YEARS SINCE *GAME* *Developer* first hit gamemakers' mailboxes. The magazine, now run by CMP Technology under the erstwhile command of yours truly, has weathered the storms of generational transitions and innumerable game hardware and software wars, all the while attempting to deliver inspirational and motivational information to game developers the world over.

But where will we be after 10 more years? And how can we ensure we cover the right material for our extremely varied audience, while maintaining our wider readability? I've been mulling over this issue for some time and now seems like a good time to ask for more feedback, since you are the people we make the magazine for.

STRIKING A BALANCE

One thing we've been striving to get right over the past few years is the balance of articles that are suitable for the many disciplines that make up video game development. Those with longer memories may recall that *Game Developer* in the late 1990s was much closer to being a game version of *Dr. Dobbs' Journal* (another CMP publication), which is to say it was very technical and much more programming-focused.

But as the make-up of the industry—and of *Game Developer* readers—has broadened, and as artists, designers, and managers filed into our readership numbers, we realized that we needed to counterbalance our technical pieces with columns and features that discuss other matters.

Trying to appeal to everyone is a tall order—but what's your take on it, readers? Would you prefer fewer State of the Industry features and more technical features? Are you yearning to read more practical business analysis? What makes you turn the pages of *Game Developer*?

POSTMORTEM PATCH

If there's one thing *Game Developer* is known for, it's the monthly postmortem articles that dissect the production of major games. But in today's day and age, with PR agencies (most of whom we love dearly) carefully vetting articles to include relatively positive spin, it's sometimes tricky to acquire true, unmasked "what went wrong" admissions directly from developers.

In fact, some of our most memorable postmortems of recent months have been written by independent developers, who have let loose with the unvarnished truth. We're also pretty keen on the postmortems that discuss one specific innovation in a game, such as this month's investigation of DeathWalk, a feature in 3D Realms' and Human Head's PREY (pg. 30).

But what universal truths are you trying to discern when reading a postmortem? When do you really care about what a postmortem has to say? Would you rather read a good postmortem of a game you don't know or an average one from a game you do? Comments are welcome.

GAME DEVELOPER PLUS?

Finally, you may have noticed that the girth of *Game Developer* magazine fluctuates significantly from month to month, depending on what's happening in the game world and what big show is approaching.

Given the current state of the print market for consumer game magazines, we're doing very well as a smaller professional trade magazine. It helps that we reach key decision-makers, too.

Still, as long as you are an eligible North American developer, you're given the magazine for free, and this sometimes acts as a strike against the amount of editorial we'd like to run, due to the economics of our business model. And occasionally, we daydream about alternatives: for instance, would you pay for an enhanced version of the magazine with more editorial if a free version were also still available?

The more feedback we receive about *Game Developer*—its content, presentation, circulation, writers, contributors, and editors—the more options and insight we'll have to deliver the best material to our entire readership.

We love *Game Developer*. We hope you do, too. And we want to find ways to continue to differentiate it, so that it's just as fresh in 2016 years as it was in 1996. Send your thoughts to editors@gdmag.com. We'll appreciatively take them into account and perhaps print some in forthcoming issues. ⁙

S!

Simon Carless
Editor-In-Chief

gamedeveloper
www.gdmag.com

CMP Technology, 600 Harrison St., 6th Fl., San Francisco, CA 94107 t: 415.947.6000 f: 415.947.6090

EDITORIAL
EDITOR-IN-CHIEF
Simon Carless scarless@gdmag.com
MANAGING EDITOR
Jill Duffy jduffy@gdmag.com
FEATURES EDITOR
Brandon Sheffield bsheffield@gdmag.com
ART DIRECTOR
Cliff Scorso cscorso@gdmag.com
CONTRIBUTING EDITORS
Jesse Harlin jharlin@gdmag.com
Noah Falstein nfalstein@gdmag.com
Steve Theodore stheodore@gdmag.com
Mick West mwest@gdmag.com
ADVISORY BOARD
Hal Barwood Designer-at-Large
Ellen Guon Beeman Microsoft
Andy Gavin Naughty Dog
Joby Otero Luxoflux
Dave Pottinger Ensemble Studios
George Sanger Big Fat Inc.
Harvey Smith Midway
Paul Steed Microsoft

ADVERTISING SALES
DIRECTOR OF SALES
Steve McGill e: smcgill@cmp.com t: 415.947.6217
GLOBAL SALES MANAGER, RECRUITMENT & EDUCATION
Aaron Murawski e: amurawski@cmp.com t: 415.947.6227
SR. ACCOUNT MANAGER, SOUTHWEST, CONTRACTORS, & MARKETPLACE
Jasmin Davé e: jdave@cmp.com t: 415.947.6226
ACCOUNT MANAGER, EAST COAST, U.K. & EASTERN CANADA
Cecily Herbst e: cherbst@cmp.com t: 415.947.6215
ACCOUNT MANAGER, NO. CALIF., NORTHWEST, ASIA & WESTERN CANADA
Nick Geist e: ngeist@cmp.com t: 415.947.6223

ADVERTISING PRODUCTION
ADVERTISING PRODUCTION COORDINATOR Kevin Chanel
REPRINTS Cindy Zauss e: czauss@cmp.com t: 516-562-5000
Julie A. Douglas e: jadouglas@cmp.com t: 516.562.5092

CMP GAME GROUP
VP, GROUP PUBLISHER APPLIED TECHNOLOGIES Philip Chapnick
VP, STRATEGIC MARKETING Michele Maguire
DIRECTOR OF MARKETING Tara C. Gibb
CONFERENCE DIRECTOR, GDC Jamil Moledina
SENIOR CONFERENCE MANAGER, GDC Meggan Scavio
EXECUTIVE WEB PRODUCER Peter Leahy
EDITOR-IN-CHIEF, GAMASUTRA.COM Simon Carless
ASSISTANT EDITOR, GAMASUTRA.COM Frank Cifaldi

CIRCULATION
CIRCULATION COORDINATOR Miguel Mendiolaza e: mmendiolaza@cmp.com
CIRCULATION ASSISTANT Michael Campbell e: mcampbell@cmp.com
CIRCULATION ASSISTANT Andrea Abidor e: aabidor@cmp.com

SUBSCRIPTION SERVICES
FOR INFORMATION, ORDER QUESTIONS, AND ADDRESS CHANGES
t: 800.250.2429 f: 847.763.9606 e: gamedeveloper@halldata.com

INTERNATIONAL LICENSING INFORMATION
Mario Salinas t: 650.513.4234 f: 650.513.4482 e: msalinas@cmp.com

CMP TECHNOLOGY MANAGEMENT
PRESIDENT AND CEO Steve Weitzner
EXECUTIVE VP AND CFO Adam Marder
SENIOR VP, AUDIENCE MARKETING & DEVELOPMENT Bill Amstutz
SENIOR VP, CMP INTEGRATED MARKETING SOLUTIONS Joseph Braue
SENIOR VP AND GENERAL COUNSEL Sandra Grayson
SENIOR VP, CORPORATE MARKETING Lisa Johnson
SENIOR VP, CORPORATE SALES Anne Marie Miller
SENIOR VP, MANUFACTURING Marie Myers
SENIOR VP, COMMUNICATIONS Alexandra Raine
VP, AUDIENCE DEVELOPMENT Michael Zane
PRESIDENT, CHANNEL GROUP Robert Faletra
PRESIDENT, CMP ENTERTAINMENT MEDIA Tony Keefe
PRESIDENT, BUSINESS TECHNOLOGY GROUP Jeff Patterson
SENIOR VP, GROUP DIRECTOR, ELECTRONICS & SOFTWARE GROUPS
Paul Miller
SENIOR VP, GROUP DIRECTOR, COMMUNICATIONS GROUP,
Stephen Saunders

BPA INTERNATIONAL
Game Developer is BPA approved

CMP
United Business Media

WWW.CMPGAME.COM

Idea:
Assemble the ultimate team to create a John Woo sequel that you can play.

Realized:
The line between video game and film is officially blurred. When faced with the challenge of creating a video game sequel worthy of John Woo's movie "Hard Boiled," the team at Midway chose to use Autodesk® 3ds Max®, Autodesk Maya® and Autodesk MotionBuilder™. Using this 3D arsenal, artists are able to create graphically stunning characters, backgrounds and animations with amazing production efficiency. Get the full story on this next-generation masterpiece at autodesk.com/stranglehold

AUTODESK® 3DS MAX® AUTODESK® MAYA® AUTODESK® MOTIONBUILDER™

TOKYO RISING

JAPANESE SHOW EXPLODES IN POST-E3 HAZE

THE THRONGS CRANE THEIR NECKS IN CHIBA prefecture's Makuhari Messe to see the latest trailer for Metal Gear Solid 4, as others busily snap photos of flyer-toting booth babes, while dodging hasty Western journalists, fighting in vain through the crushing crowds. The Tokyo Game Show is upon us once again, and Sony, Microsoft, and to a lesser extent Nintendo, are all making bids for global dollars.

In the wake of the E3 downsizing announcement, all eyes were on TGS as the biggest consumer game show in the world. Though the affair has always been open to the public, with the first day exclusive to business and media, it has never before reached the epic proportions of attendance as it did this year. Across three days a total of 192,000 fans and business professionals filled the convention center on the outskirts of Tokyo, a truly noticeable increase over last year's 176,000.

Sony's PlayStation 3, playable for the first time by the Japanese public at large, drew the biggest and most eager crowd who waited in two-hour lines for hands-on time with games. Topical buzz at the show surrounded the machine itself more than the games (such as Gran Turismo HD, Lair, and Resistance: Fall of Man) though, as the mockup unit on display was constantly surrounded by photographers.

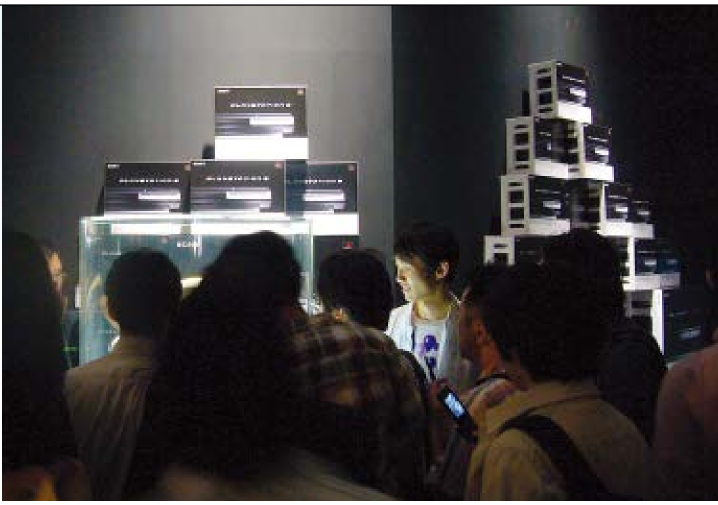
Sony executives also announced a nearly 20 percent drop in price for the 20Gb-sporting system in Japan, emphasizing that a lower price point was essential to promote sell-through. This brings the price to 49,980 yen, or roughly $418. HDMI support was also announced for the lower model. As of press time, no announcements have been made regarding potential price cuts for the rest of the world.

Microsoft had a very solid TGS showing as well, with fans most anxious to get a glimpse of Mistwalker's Lost Odyssey and Epic's Gears of War. The company also unveiled a new trailer for Halo 3, which was disappointingly short on gameplay footage, but generated interest nonetheless. In order to foster excitement around its Xbox Live service—which like the Xbox 360 itself has yet to take a significant foothold in Japan—Microsoft distributed free Xbox Live points, useful toward the purchase of Xbox Live Arcade titles. These cards granted the user a mere 100 points, a token gesture, but a gesture nonetheless. Perhaps reduced prices and pack-in deals toward the end of the year may turn the tide in the company's favor.

Nintendo followed its typical path of eschewing the consumer-oriented show, only allowing Wii software to be demonstrated at specific booths by company representatives. The company is clearly banking on the price and branding to sell the unit, rather than significant pre-release buzz on a popular level. DS games were playable at the show, but only in the booths of the games' respective publishers.

The 2006 Tokyo Game Show centered around anticipation, as the next generation finally rolls out in full. Each major console manufacturer is taking a rather different route to success, making this Christmas holiday an incredibly important one for not only the big three, but the game industry at large.

—Brandon Sheffield



192,000 people flooded the Tokyo Game Show to lay their eyes on a PlayStation 3 mockup.

GAME WEEK GRACES LONDON

A SLIGHTLY DAMP LONDON IN EARLY autumn was the venue for the 2006 London Games Festival, a conglomeration of get-togethers that combined key public and industry events throughout the English capital during the first week of October, including the London Games Summit, presented by ELSPA and TIGA, programmed by the CMP Game Group; Game Developers Conference London, also presented by CMP; the London Content, Outsourcing and Middleware Market, presented by TIGA; the London Game Career Fair, presented by Gamasutra.com and GamesIndustry.biz; as well as BAFTA's British Academy Video Games Awards.

The BAFTA Awards, which are open to video games released worldwide in the past year, honored Ubisoft's Ghost Recon Advanced Warfighter as best game and for best technical achievement.

A notably large variety of winners were recognized, with several titles picking up more than one award: Sony's LocoRoco took two awards for best character and best children's game. Shadow of the Colossus won in the categories artistic achievement and best action and adventure game. Other notable winners were Nintendo's DS title Brain Training in the innovation category and LucasArts' multi-platform game Lego Star Wars II for best gameplay.

At the Game Developers Conference London, keynote speaker Jamie Macdonald, vice president of Sony Computer Entertainment Worldwide Studios, tackled one of the toughest challenges of modern game development in a speech titled "Developing for a Networked Experience." He homed

The child-friendly LocoRoco won two BAFTA awards.

BROADBAND BULLSEYE

NEW DEVELOPER, PUBLISHER CARVES MULTIMEDIA HEADWAY

GAME DESIGNER JON VAN CANEGHEM (MIGHT AND MAGIC) and former Electronic Arts executive Dr. Lars Butler have faith in the power of broadband. They believe in it so resolutely that they've built a new game development and publishing company, Trion World Network, devotedly solely to delivering content with it.

Van Caneghem and Butler insist that, because the future lies in broadband, media companies need to rethink their strategies for providing entertainment, as well as the substance of the entertainment. How video games are developed and distributed today, says van Caneghem, "is a byproduct of old architecture." Trion, he adds, is taking multimedia and going straight back to the drawing board, building new architecture that will allow for various hybrid types of entertainment, delivery methods, and connectivity. "All devices can tie into some game world, and that can only happen if you build the architecture right," he says.

Trion is not restricting itself to games. With a multitude of partnerships, from game development studios to corporate media giants (think Paramount), the company plans to do business in not one type of content, but three: games, online, and traditional media (television and film).

Funded by Tier 1 Silicon Valley and venture capital firms DCM and Trinity Ventures—rather than a major game publisher—the company already has two major offices in Redwood City, Calif., within sight of EA's headquarter campus, and Austin, the U.S. epicenter of talented online game developers. Says Butler, "It has not been a challenge for us to raise a lot of money." Nor has recruiting talent been a challenge: Trion has already signed on a number of expats from EA, NCsoft, and Sony Online, hoping to get the most out of hiring veterans who understand connectivity, online delivery, and good game design.

Jon van Caneghem (l) and Lars Butler (r) want to change games, one broadband customer at a time.

The company is still many months away from giving consumers (or journalists) anything tangible. So it's vagaries and promise for now. According to Butler, the technology itself won't even be ready until sometime in 2007.

—Jill Duffy

WII REMOTE TO LIGHTEN CODERS' LOADS

NINTENDO AND AILIVE HAVE announced the release of LiveMode for Wii, a product that allows the Wii remote to learn via artificial intelligence. The software is poised to ease the difficulties of programming for Nintendo's new input device; officials claim it will allow developers to train the remote through example rather than code. In the hands of developers, Nintendo hopes this tool will further simplify the development process for Wii, already the most affordable new console to develop for.

The announcement specifically calls out independent developers as a market for this product and has priced it to match, at a mass-adoption license fee of $2,500 per seat.

AiLive will offer tutorials and demos of the product. Its previous product of note was LiveCombat, which taught AI characters using the example of a player-controlled team leader. The LiveMode product seems to build on that concept.

—Brandon Sheffield

CALENDAR

Asia Pacific Entertainment and Media Summit
Sofitel Los Angeles
Los Angeles
November 6 and 7
Price: $159–$349
www.apemsummit.com

Montreal International Game Summit
Palais des congres de Montreal
Montreal
November 8 and 9
Price: $150–$495 CAD
(exhibition-only rates also available: $25–$40)
www.montrealgamesummit.com

eGames & Entertainment Expo
Melbourne Exhibition Centre
Melbourne, Australia
November 17–19
Price: $13–$38 AU
www.auexhibitions.com.au

Sigrad Conference
Högskolan i Skövde
Skövde, Sweden
November 22 and 23
Price: contact organizers
www.his.se/sigrad06

Game Connect: Asia Pacific 2006
Brisbane Convention and Exhibition Centre
Brisbane, Australia
November 30–December 2
Price: $100–$350 AU
www.gameconnectap.com

in on some of the next-generational problems common to all gamemakers: the increasing cost of production, a shift from package-centric to network-centric gaming and development, and strategies for success.

Sony's E-Distribution Initiative, the rough equivalent of Xbox Live Arcade, was one particular focus of Macdonald's analysis. Its job? "To drive the direct delivery of content to consumers through PS3 and PSP's Network Platform," targeting new developers through lower barriers of entry. "We're talking about shortform works of content, and we really want to encourage innovation," Macdonald said. EDI was devised to allow developers "to work on those great ideas that they've had but have been told will never work as a triple-A, 20-hour game."

At the related London Games Summit, an event tailored more to the business minds of games, the commencement address was given by Lord Sainsbury of Turville, England's Parliamentary Under Secretary of State for Science and Innovation.

Lord Sainsbury, whose position falls under the Department of Trade and Industry, stressed that the U.K. government is working strongly with the game industry in the U.K. to create the "best possible conditions in this country for your industry to innovate and grow." He noted the financial strength of the game industry in Europe and the U.K., saying its worth now surpasses that of the film industry. "The computer games industry is economically much more important (than film)," he said. "It is the innovation and creativity that has allowed this sector to grow."

A comprehensive digest of the London Games Festival and its various events is available on *Game Developer*'s sister web site Gamasutra.com.

—Simon Carless and Jill Duffy

OUR RATING SYSTEM :

★★★★★ EXCEPTIONAL ★★★★ GREAT ★★★ FAIR ★★ POOR ★ UNFORTUNATE

LUXOLOGY'S MODO 202

By David March

LUXOLOGY

★★★★★

STATS

Luxology
1670 South Amphlett
Blvd., Suite 214
San Mateo, CA 94402
www.luxology.com

Price

$895 ($395 upgrade)

System Requirements

- 1Gb RAM
- 100Mb available hard disk space (3Gb required for all content and integrated training materials)
- OpenGL-enabled graphics card
- Monitor resolution of 1,024x768 or greater
- DVD-ROM drive (for support materials)
- Internet connection required for product activation
- Adobe Photoshop CS or later required for modo ImageSynth

Macintosh Requirements: Mac OS X 10.3.9 or later; Macintosh G3, G4, G5 or Intel processor.

Windows Requirements: Microsoft Windows 2000 or Windows XP; Intel Pentium 4 or AMD Athlon processor.

Pros

1. Unparalleled subdivision modeling abilities and workflow.
2. A slew of new tools including a great painting tool.
3. New renderer.

Cons

1. Wish list 1: animation and rigging.
2. Wish list 2: particles with a cloth and hair system.
3. Wish list 3: physics system.

WHAT DO YOU GET WHEN YOU ADD A bunch of cool features and an efficient SDS modeler to an existing tool that's rock solid? Modo 202.

The team that's driving modo's development is definitely listening to and thinking about what the majority of the user base wants. After waiting more than a year for this release, modo enthusiasts will be extremely pleased with the new updates from Luxology. Users saw major updates in the release of modo 201, and then two months later, after final tweaking, bug-fixing, and throwing in a handful of additional features, modo 202 was sent out as a free upgrade to license holders.

Modo has impressed me with many improvements in a number of areas, most notably modeling tools, workflow, painting tools, and a new renderer. Game developers will be impressed by the Power 2 grid, which lets them create assets using a unit grid (each line of the grid is set to a power of two, just like texture resolutions), and the Object Baking tool, which bakes detail from high-resolution objects into lower-poly count models. UV tools have been revamped as well, giving developers yet another reason to closely evaluate modo, and maybe adopt it permanently.

RENDERING

One of the biggest updates to modo is the new rendering engine. The renderer is bucket-style, and supports global illumination, subsurface scattering, anisotropic blurry reflections, motion blur, depth of field, and displacement. If your machine supports multithreading, you will see a dramatic increase in performance with modo 202.

Worth checking out is a tutorial video called "Render Project," which explains how to set up a scene and how the Fresnel option affects the scene. Modo also has a new preview renderer that allows you to model in near real time. This feature is like the surface preview window in LightWave 3D. Although you can configure your working environment any way you would like with this option, there is a reliable default triview that supports a preview window, camera, and perspective view.

This image, created in modo 202 by Jacques Defontaine (and courtesy of Luxology), makes use of the tool's sub-surface scattering ability.

Another newly added feature is the render to region option. This feature is great if you're happy with your entire scene except for maybe some bad shadow or glow in the corner of your render. You can render away on that bad spot without having to redo the whole scene over and over.

UV IMPROVEMENTS

Newly added to 202, UVs that overlap are marked in red. It's a pain to zoom in and find small overlaps in a tightly packed map, so this feature makes perfect sense, especially for creating normals. Also in 202 the UV Pinning update adds a whole new level of control over what UVs you do and do not want relaxed.

A few more of 202's great updates include object-to-object detail baking for displacement mapping or normal mapping use; a new game creation unit system; and the all-useful image invert command, which lets you flip your image inside modo.

MODELING

New modeling tools were added to modo's "best of hybrid" style subdivision-polygon modeler, including a new Solid Sketch tool. This tool is great for quickly laying out a base figure to fill out your work area. Think of it as a page for quick concept sketching, using clay-like connectors in an organic fashion to get your basic modeling idea down.

It reminds me of how one might create organic concept art using Zspheres in Zbrush. The Solid Sketch tool is a great feature for laying down basic forms to get you started on any organic form—or even something that would be hard edged and mechanical.

Modo 202 also sports a Sculpt tool, which simulates sculpting in clay. You can drag across the surface and deform the geometry, creating bumps—or by holding down the control key—reversing the action. Scaling is controlled via the right mouse button.

Another improvement that I'm very happy with is the simplified way in which you can weight simple edges. In older versions, you had to go to the vertex map list under weight map and select subdivision. But in modo 202, all you have to do is go to vertex map and use the edge weight tool and drag it out in the viewport. Because I use this function frequently when subdivision modeling, I definitely appreciate modo's approach.

Smaller updates also dot modo 202's fact sheet. For example, the edge slice tool lets you bevel points in two dimensions. Also new is a really cool thicken tool, which is similar to 3ds Max's shell modifier.

Finally, there's a new Pen tool. The modo Pen is similar to a device found in the plug-in Polyboost for 3ds Max. The major difference is that for modo, it's free. It's great for redoing the relief detail on an extremely high-poly model. For example, you can draw polygons on top of your high-poly mesh with the constrain-to-mesh feature turned on. And with the free update to 202, the Pen tool has a newer, unique extension for creating subdivisions with finite control.

Personally, I find that the best way to learn all these new features is to use modo's Quicktime modeling tutorials, which can be downloaded from Luxology's web site.

WORKFLOW

Luxology has definitely spent time from the beginning making modo's interface incredibly user-friendly and customizable. Users can basically clear out the entire screen space and lay out the interface any way they want. I think this feature is really quite cool, and the possibilities seem endless.

Modo 202 also allows for customization of the keyboard mapping and mouse inputs. Although these features may seem minor, busy game artists know the power—in workflow speed—of being able to map the functions they use most often to the keys that are most easily reached. The default keys work just fine, too.

Yet another feature is the Tool Pipe, which is extremely powerful. It takes modo's core functionalities and lets you create combinations of tools and tool modifiers. For example, you could create a function that rotates with a specific falloff every time. It beats the heck out of scripting.

Other simple workflow features, like highlighting child-to-parent items to show their relationship, give modo 202 a useful sort of elegance. There's also a new browser for presets and expand/collapse usability for sub-items in trees.

PAINTING TOOLS

Modo 202's painting tools are a major added function and are exceptionally user-friendly, just like the rest of the package. Simply click and start painting away.

You can create a new image from within modo or simply import an existing image to paint on. Not only can you paint on your 3D model, you can also paint directly on your 2D UV image view on your wires with a texture. It's all pretty straightforward and reminds me of Deep Paint or BodyPaint 3D.

As far as the brushes go, you can use the presets, create your own custom ones, or even grab them from Photoshop.

The Tool Pipe also allows for interchangeable brushes, inks, and falloffs. The image ink tool can be explained as painting on a preset texture like a lizard skin onto your models with the option to blend different scales in. Modo also has masking and blending features, which are super handy when painting on your models. And since the 202 update, you can display in 3D with handles for position and rotation, which is extremely useful. 202 is also "smarter" now about the way it paints on complex UV folds and borders. Best of all, painting is supported by pressure-sensitive input devices, which is a must if you're going to paint.

It seems as if the only thing missing for modo now is an animation system, but I've heard it may be on the way.

WORTH ITS WEIGHT

In short, modo 202 is one of the best—if not *the* best—SDS modelers out there. If modelers test it and use the plethora of video tutorials that come in the help directories, I think they will be impressed on how quickly they can get up to speed.

Luxology has done a great job shipping these well thought-out instructional tools while also fixing, adding, and updating tools.

There's little room for negative feedback about modo. It works extremely well with other packages in terms of importing and exporting files. Even the "cons" I listed (see pg. 7) are personal wish list items, not failures to deliver on Luxology's part.

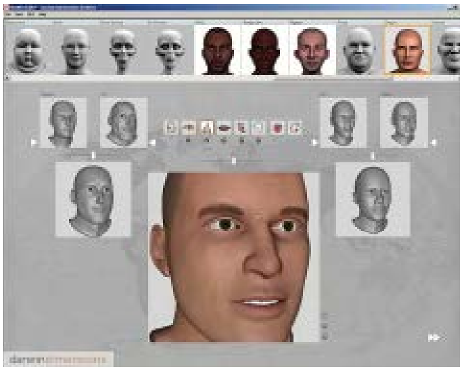
At this point I can only selfishly state I want more videos—I always want more of the good stuff. With the new painting and rendering abilities, artists really don't have to export for presentation if they don't want to. However, don't forget that modo was built from the get-go to be super friendly and work well with all your tools. For the $895 price tag, I think it's well worth its weight in gold.

DAVID MARCH *has recently joined Irrational Games in Australia as lead animator. Send comments to him at* ***dmarch@gdmag.com****.*

DARWIN DIMENSIONS' EVOLVER
By Tom Carroll

WHO DOESN'T LOVE THE LATE-NIGHT talk show game of creating hilarious offspring by crossing the genes of two celebrities? What would Tom Cruise look like with Yoda's ears and skin color? What would you get if you crossed Christina Aguilera with Dobby the House Elf (of *Harry Potter* fame)? And what might a character that was half "W" and half Tony Blair look like?

While not quite apples and apples, evolver by Darwin Dimensions allows anyone who's willing to pay $39 to combine various attributes from a long list of faces and figures to form their own custom characters. The resulting characters can be converted into a form



Darwin Dimensions' evolver uses a simple interface and a family tree-style approach to character creation.

DARWIN DIMENSIONS

STATS

Darwin Dimensions
1155 University
Suite 901
Montreal, Quebec
H3B 3A7, Canada
888.392.1331
www.darwindimensions.com

PRICE

$39 evolver Basic (software only; customers can purchase geometry-only of a model for $49, or fully featured 3D humanoid for $1,995)

$4995 evolver Pro (ships with the capability to output three fully featured 3D humanoid models; additional models can be purchased for $1,000 each)

Evolver complete: Product and pricing not yet available.

System Requirements

Any Windows XP-compatible computer

Pros

1. Easy to understand and use.
2. Extremely fun to play with.
3. Simple interface.

Cons

1. Not the answer for companies that create unique characters.
2. Artists still need to clothe the characters and put hair on their heads.
3. May be too much fun for its own good.

that's recognized by most customary 3D packages (those that support Autodesk .fbx) or that can filter directly into a Maya pipeline. If you've got seven days to kill, download the free trial version of evolver from the company's web site and watch in awe (and shame) as time flies by.

THEY CAME FROM THE GENE POOL

Darwin Dimensions' stated reason for evolving evolver is to "automate the creation of a near infinite variety of high-quality 3D characters by blending and combining physical attributes derived from a vast 'virtual gene pool.'"

However, my understanding of the software's purposes reads something like this: "To make character creation less like drudgery and more like a fun game."

Regardless of the stated reason for development, the company has succeeded in making a product that creates interesting character models quickly and easily—and for just $39, you don't have to break the budget to buy a copy and have a little fun with it.

Users start their evolver experience with a screen that shows a large selection of heads as well as a few trolls and aliens thrown in for good measure. These are called ancestors.

Highlighting an ancestor icon, the user loads it into one of four boxes by clicking on an arrow button to the side of the box. Up to four heads can be loaded at any one time, but ultimately, up to 20 of them can be used to build a new character head.

Sliders make it possible to quickly combine elements (eyes, ears, noses, head shapes, mouths, and so on) to make a unique head.

If you try, and sometimes even if you don't, you can attain some very unusual mugs. But by predefining the type of character you want to build (for example: Asian female with elfin ears and ample chest) and staying within those parameters, you'll avoid the silly monkey business that inevitably results.

CORPORAL CREATION

But what's a head without a body? The next step is to repeat the same process, but this time with a large selection of male and female body types. The head you previously fashioned is now positioned atop a body. Within moments, even the inartistic among us can fashion an appropriately bodacious body.

Evolver allows users to create just heads or both heads and bodies, though no hair or clothing is available.

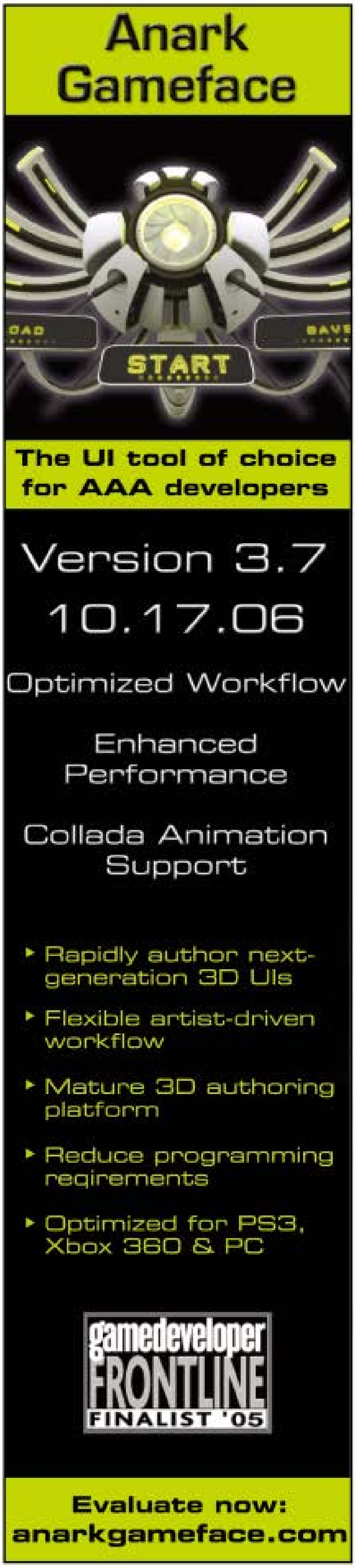
The last step is to assign a skin color to the finished head and body. This is accomplished in a similarly straightforward way by selecting and blending iconic images of Caucasian, Black, Asian, (and alien) faces.

Saving the work is as easy as hitting the Save As command and typing in a unique name for your .dde file (Evolver's proprietary file format); evolver also supports Autodesk's .fbx format.

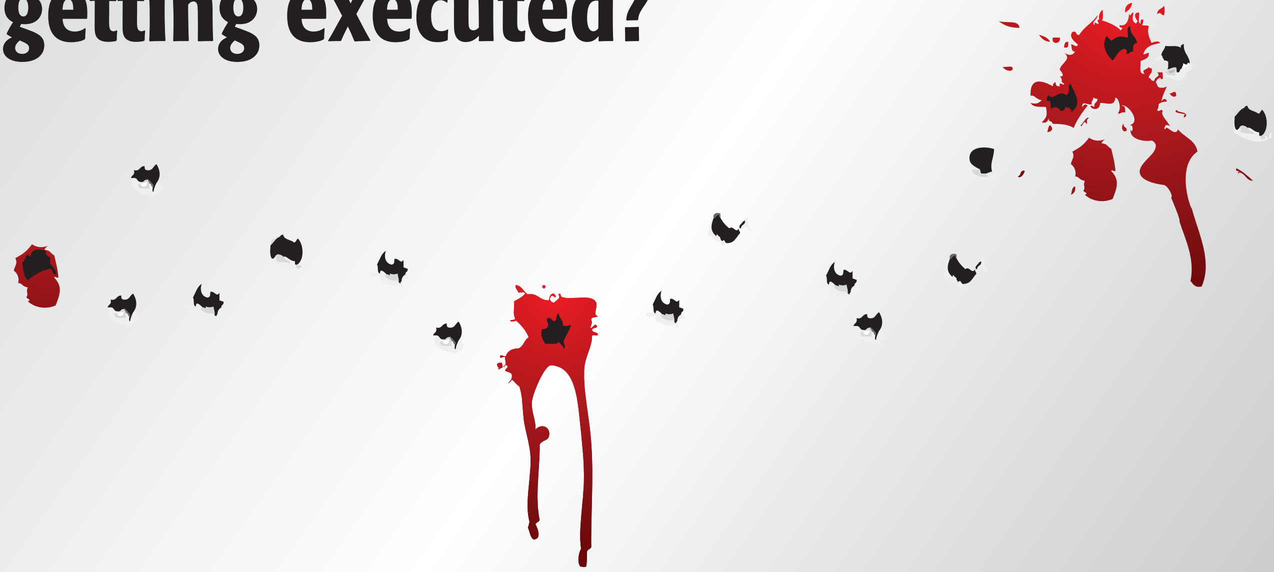
The file is then submitted to Darwin Dimensions. Whether purchased geometry-only ($49) or as a fully featured 3D humanoid ($1,995), the model is delivered to a secure FTP site for pickup. As of press time, this automated process was not operating, but the pros behind evolver were quite willing and able to process orders on a case-by-case basis while they perfect their delivery pipeline.

Evolver is absolutely *not* the solution for every company, especially for those with staffs of hungry character artists, modelers, and texturers. But it is the perfect solution for any company that does not have such a staff and still needs a wide selection of character models that have the professional quality that the game, television, and movie industries demand. ⁙

TOM CARROLL *is a video game artist and freelance writer who strives to understand only enough of his corner of the universe to be able to sleep at night. Email him at* ***tcarroll@gdmag.com****.*



Are all of your test cases getting executed?



Introducing:

TestTrack® TCM

The ultimate weapon for test case planning and execution.

Issue & Defect Management

> Test Case Planning & Tracking

Configuration Management

Automated Functional Testing

Essential Process Management

A cruel fate awaits a buggy game and its QA team. The ability to know what's been tested and what hasn't can be the difference between success and failure. TestTrack TCM puts you in control and keeps you out of the line of fire.

In TestTrack TCM you have the tool you need to write and manage thousands of test cases, select sets of tests to run against builds, and process the pass-fail results using your development workflow.

With TestTrack TCM driving your QA process, you'll know what has been tested, what hasn't, and how much effort remains to ship a quality product. Deliver with the confidence you've tested everything.

- Ensure all steps are executed, and in the same order, for more consistent testing.
- Know instantly which test cases have been executed, what your coverage is, and how much testing remains.
- Provide engineers with the detailed steps and data they need to quickly reproduce problems.
- Streamline the QA > Fix > Re-test cycle by pushing test failures immediately into the defect management workflow.
- Balance the testing effort among team members, both local and remote.

Download your **FREE** fully functional evaluation software now at **www.seapine.com/gdtcm** or call 1-888-683-6456.

Seapine Software™
Managing Process, Change & Quality
Throughout the Enterprise

GETTING LUCKY

THE MAGIC: THE GATHERING CREATOR'S STANCE ON CHANCE

I CAN FIND A BOARD OR CARD GAME FOR ANY GROUP OF PLAYERS. Game players or people who never played games, old or young, in large or small numbers, with confrontational or passive personalities—there are games out there for them all. While I weigh many factors in choosing a game, by far the most important is the amount of luck inherent to the gameplay. If the game has a lot of luck, it usually appeals to a diverse group.

Games in the non-electronic world are widely varied in luck, but computer games are a different story, as very few of them allow any real chance for a beginner to win against a skilled opponent. The number of electronic games I can play with my parents, kids, wife, or friends from outside the game industry is extremely limited.

Historically, games usually evolved in such a way as to reduce the amount of luck in them. Even chess at one time had dice. The people who are in a position to modify a game are likely to be very good at it, and the sort of modifications they will be drawn toward are the ones that showcase their talents and their friends' talents—although they, of course, are all top players.

In other words, as games evolve, they tend to become better for the experts, but not necessarily better for new or non-dedicated players. A game that illustrates this conflict is

RICHARD GARFIELD *currently teaches at the University of Washington. His first game, Magic: The Gathering, a collectible card game, was released commercially in 1993. He also consults for video game design. Send comments about this article to* ***editors@gdmag.com****.*

CHESS BOARD PHOTO COURTESY OF BELEN MENDEZ;
DICE PHOTO COURTESY OF MICHAEL CONNORS

GETTING LUCKY

First person shooters, such as HALF-LIFE 2, could use luck to allow players with greater variation in skill to compete with each other.

Settlers of Catan, one of the best-selling board games of recent years. The only consistent criticism I have heard leveled at it (always from dedicated gamers) is that it has too much luck. But it's rather possible that the abundance of luck is exactly what made the game so wide-reaching.

Enlightened players, skilled or not, will appreciate luck in their games for a number of reasons. First, they can play challenging games with a much broader audience, allowing them to easily assemble a galley of players and lure their friends, who would otherwise play something else, into the game. Second, if skilled players want to experiment and try off-the-wall strategies, the more luck a game has, the more forgiving it is—after all, no one is expected to win every time. The only cost of all these terrific benefits is that skillful players must manage to swallow their pride and settle for winning a majority of the time, rather than all the time.

LUCK VERSUS SKILL IN TRADITIONAL GAMES

GAME	AMOUNT OF LUCK	AMOUNT OF SKILL
Poker	High	High
Basketball	Low	High
Tic-Tac-Toe	Low	Low
Slots	High	Low

We gamemakers are at a special time in game history. Fifty years ago, games were made with no credit to the designers or perhaps had no designers at all, with changes being wrought by players over time. But our nascent game design community tends to comprise game experts; it's in our best interest to examine our own instincts openly with regard to how much luck should be in a game.

WHAT IS LUCK?

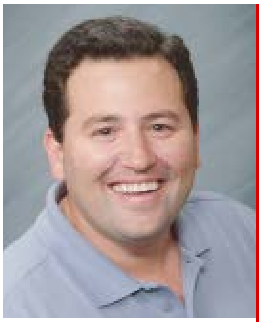
I define luck in games as uncertainty in outcome. If better players always win against weaker opponents, then there is no luck in the game. However, if the better player sometimes loses, then luck must be present, and the more a better player loses, the more luck is in the game.

Uncertainty in outcome is most strongly associated with randomizers, such as dice, spinners, shuffled cards, and in the case of video games, randomly generated numbers. But these overt luck generators are not the beginning and end of luck. If the game's outcome isn't certain—whether the game is baseball or rock, paper, scissors—there is luck involved. The variability in baseball may come from muscle fatigue, or weather, or endless numbers of more subtle influences that we have no more chance of determining than the path of a roulette ball. The variability in rock, paper, scissors? Any randomizer in that game lies in the players' brains.

This definition of luck, based solely on uncertainty of outcome, has an interesting consequence in that an otherwise deterministic game can have luck. Let's take for example a game I call pi-eye, in which each player has 30 seconds to guess a particular digit of pi, say the 37 billionth digit. There is no overt luck in pi-eye because it's possible to calculate the answer. Yet, most players would rather simply take a one in ten chance of guessing the correct digit. Players could improve their odds by studying pi or theories about its digit distribution, or even reduce the luck to 0 by discovering a formula to determine the digits of pi—but even though those possibilities exist, most of us would rather opt for luck.

CONTINUED ON PG 14

ADVERTISEMENT



Canadian-born Mark Rein is Vice President of Epic Games based in Raleigh, North Carolina. Their Unreal series of games is reported to have sold over 7 million copies world-wide. Epic's Unreal Engine 3 has won Game Developer Magazine's Frontline Award for Best Game Engine for the past two years. Since 1992 Mark has worked on Epic's licensing & publishing deals, business development, public relations, academic relations, marketing and business operations Currently in development at Epic: Gears of War for Microsoft and Unreal Tournament 2007 for Midway.

Upcoming Epic Attended Events:

Serious Games Summit
Washington, DC
October 30-31, 2006

Game Developers Conference
San Francisco, CAL
March 5-9, 2007

Please email:
mrein@epicgames.com
for appointments.

POWERED BY
UNREAL
TECHNOLOGY

Unreal® Technology News

by Mark Rein, Epic Games, Inc.

VOGSTER LICENSES UNREAL ENGINE 3

Vogster Entertainment, whose development centers are based in Kiev and Moscow, recently announced that it has licensed Unreal Engine 3 technology in connection with the development of Vogster's next-generation massively multiplayer online CrimeCraft™ game.

Arseniy Nazarenko, General Producer of Vogster's Kiev production office said: *"Vogster has set upon an ambitious game-development course. We are preparing to launch CrimeCraft™, our own unique and exciting MMO game, in 2008. CrimeCraft™ is going to shake the global video game industry with its innovative features and characters actions. The Unreal Engine 3 is the most powerful game engine available, and will play an instrumental role in the realization of our development plans."*

INTRODUCING THE INTEGRATED PARTNER PROGRAM

Epic Games has established the Integrated Partners Program (IPP) for the purposes of having a formal business relationship with selected companies making cross-platform technologies which integrate with, and are complementary to, Unreal Engine 3. Epic provides continuous UE3 source code access and full technical support to IPP members. Companies who join the IPP agree to provide a high level of technical support for UE3 licensees through Epic's established support channels and keep their implementations up-to-date with the latest UE3 version. The IPP will make it easier for UE3 licensees to incorporate 3rd party middleware solutions from IPP vendors.

10 companies have signed agreements to be in the IPP. Here are comments from the first four companies to announce:

IDV, Makers of SpeedTreeRT: *"IDV's technology relationship with Epic began several years ago and has put SpeedTree in the hands of a large community of outstanding game developers,"* said IDV President Michael Sechrest. *"Formalizing the partnership is an obvious next step and lets our customers know how committed we are to working with Unreal Engine 3."* See www.speedtree.com

Quazal, makers of Products and services for multiplayer games: *"We're very excited to have integrated our Rendez-Vous and Spark! technology into Unreal Engine 3,"* said Mike Drummelsmith, Developer Relations Manager of Quazal. *"Having Spark! on Unreal Engine 3 will allow for a very rapid deployment of a robust online game lobby, while the full Rendez-Vous API will be available for developers looking at more customized solutions. This should really help developers deploy their online games quickly, with advanced features."* See www.quazal.com

Engenuity, makers of AI.Implant: *"This is a significant milestone in the execution of our technology vision for game AI and a validation of our intensive 18 month iterative development process for the Unreal Engine 3,"* said Paul Kruszewski, chief technology officer at Engenuity Technologies. *"We have always set out to provide our users with only premier integration options. We delivered on this with our Maya® integration for our Film customers, and now we are delivering a leading-edge integration for our games customers with the world's most popular AAA system – Unreal Engine 3."* See www.ai-implant.com

EPIC GAMES CHINA

Fonix, makers of voice recognition software: *"The IPP is a great opportunity for Fonix," says Tim K. Hong, Vice President, Fonix Games. "It allows us to market our voice recognition software and integration code directly to videogame developers who have licensed Epic's Unreal Engine. Epic is currently one of the most respected middleware providers in the game industry, and joining the IPP allows Fonix to introduce our voice recognition technologies to the best developers in the business working on the latest cutting-edge games."* See www.fonix.com

EPIC ANNOUNCES EPIC CHINA

Epic is pleased to announce that we have opened a Chinese subsidiary, Epic Games China, being run by a team responsible for content development at high-profile studios including Ubisoft Montreal and Ubisoft Shanghai. The management team worked on many commercially successful, critically acclaimed titles in the Splinter Cell, Rainbow Six, and Ghost Recon franchises.

Epic plan to invest significant resources to ensure quality is up to its high standards. Epic will be outsourcing to Epic Games China for high-quality content for its upcoming games, including Unreal Tournament 2007. Epic Games China will also provide low-cost, high-quality outsourcing for our Unreal Engine 3 licensees.

EPIC GAMES

For UE3 licensing inquiries email:
licensing@epicgames.com

For Epic job information visit:
www.epicgames.com/epic_jobs.html

www.epicgames.com

GETTING LUCKY

CONTINUED FROM PG 12

LUCK VS. SKILL

What is a good amount of luck, relative to skill, for a game? This question sounds benign, but it contains a common fallacy about games. How much luck there is in a game has little to do with how much skill there is. A game can have a lot of luck and a lot of skill.

An example of such a game is poker. If you sat down with the world champion, you could win a hand, regardless of your skill. You might even be able to win a session. But once you start stringing sessions together you have no hope of winning (unless you too are a poker stud). In fact, repeated play will eradicate the luck from almost any game. If players play a game enough and there is any skill difference between them, the most skillful player will win the majority of the games.

If poker doesn't convince you that a game can rely on both luck and skill, I can introduce you to a game I've created called randochess. In randochess we each roll a die, and the high roll wins with ties broken by a game of chess. Randochess clearly has more luck than chess, yet in some sense it has just as much skill as chess. After all, every book of strategy ever written about chess applies equally to randochess.

It is just as challenging to be a good randochess player as it is a good chess player, but you won't win as often leveraging your skill in randochess as you will your skill in chess. This distinction is important because it illuminates the fact that games can't be trivialized merely on the basis of luck—games with a lot of luck can be as rich as any other game, and as hard to master. In fact, one could argue that games high in luck are harder to master since a player can more easily win with bad moves or lose with good moves—which will certainly slow down the learning process.

BENEFITS OF LUCK

There are three benefits to using luck in game design. First, high-luck games broaden the range of competition. Second, luck removes players' ego crutches. And third, luck increases the variety of the gameplay.

Range of competition. The more luck there is in a game, the more easily skilled and unskilled players can play together. In a game without luck, the more skilled player will win every competition giving the skilled player no challenge and the less skilled player no chance of victory. A game with low luck can be a fine game of course, but it demands that players of similar skill always compete against each other only. The less luck in the game, the tighter that range of skill the players will need to have for a satisfying experience.

Online games, of course, have less need for broadened range of competition due to computer matching (see "Ranking and Matchmaking," October 2006). However, there is something to be said for being able to choose opponents and teammates based on criteria other than their skill.

Ego crutch. Why do skillful players frequently criticize luck in games? It's probably because the luck in the game can marginalize their skill. When skilled players have played the better, more skillful game and still lose, they say, begrudgingly, that only fate is to blame. And when they win in a game that has a lot of luck, the opponents won't credit their brilliant play, only their good fortune. Luck can in this way become a player's enemy, denying them their rightful bragging rights and glory in either case.

This apparently negative aspect of luck is hiding a very useful concept for game designers. Many people take pleasure in blaming their defeats on bad luck, but have no problem taking credit for their victories, regardless of the circumstances. Certainly, these players will often complain about the luck when they lose, but really the element of chance is beneficial to them: it is protecting their egos, just as surely as it can injure the ego of a skillful player.

Variety of gameplay. Luck in games often broadens the type of strategies that people can use, adding variety to the game. With the uncertainty luck brings, the most conservative players will have to take crazy chances if they want to succeed from time to time, and the players who always take the long-shot will find they should sometimes ease back on the throttle and play it safe.

Suppose in an economic military game, players think that building a lot of tanks is the best strategy. Whether it is the best strategy is not too important—what counts is how the player sees it. Players would spend most of their time building tanks. The only ones who would typically stray would be the beginners who didn't know any better or the elite who were secure enough in their stature to experiment. Suppose we introduce a random element into the military units of the game, such as units being priced randomly, occasionally unavailable, or varied in power, based on the random availability of supporting technologies. Players might now regard tanks as being generally the best unit, but no one will believe as a rule they are always the best, which might lead to more players exploring more and different strategies.

CONTINUED ON PG 16

short+luck=good combo

FOR A GAME WITH A HIGH AMOUNT OF luck to be really satisfying for a broad audience of players, it should be a game that is also fairly short.

The definition of "short" varies from group to group: for people with a lot of free time and energy, having a high-luck game that is longer won't be so bad. However, a long game with a lot of luck does threaten to frustrate the more skillful players, who don't want to invest a lot of time and energy on a spurious outcome.

At the same time, a long game with a lot of luck holds little interest for less skillful players because they are not favored to get a taste of victory in a single play and, being a long game, starting a new game afresh might take a while. The shorter the game, the more likely a less skillful player will have at least some wins.

The variety of play available in a high-luck game really shines through in shorter games. The longer the game, the fewer games will be played during a particular game session, and the fewer games you play in a session, the less chance for that variety to show itself.

"A Computer Game for Real Life Crises"
-The Washington Post

Another way luck can extract variety from a game design is when it is used to rebalance the payoff for player skills. If a game involves two skills, A and B, and A is very important to winning the game, while B is not so important—by making it so that A has more luck involved you can raise B's relative importance. This could lead to a game in which the players have more strategic space to explore. Remember the old computer game ARCHON? ARCHON had a chess-like game, but the battles were resolved with an arcade-like game. My experience with it was that a player's skill at the chess portion of the game was irrelevant relative to the arcade portion. Presumably, with players close to one another in arcade skill, the chess portion of the game would become more important. If the arcade portion of the game had more luck, then the chance of that part of the game being interesting with disparate skills is higher, and the chess portion of the game will become more important.

LUCK AND SINGLE-PLAYER GAMES

Some of the points I've made so far don't apply—or apply differently—to single-player games. For one, there's no benefit to increasing the breadth of player skills that can compete together if the game is a one-player experience.

The role of luck as an ego crutch in a single-player game can still apply, though it is less important. Players are less likely to need the crutch when playing against a computer since it is generally less threatening to lose to a computer than to another flesh-and-blood human, who is much more likely to talk trash. But it still helps to have bad luck be your focus of blame for defeat.

ALTHOUGH ADDING LUCK TO AN EXISTING GENRE CAN ALIENATE FANS, IT IS OFTEN WORTH THINKING ABOUT IN TERMS OF GAME DESIGN. IT MAY BE POSSIBLE TO ADD AN ELEMENT OF LUCK THAT EVEN THE ELITE PLAYERS WILL FIND COOL.

Additionally, if the game is intended to be played only once, as many video games are, then there's little benefit to increasing the variety of the game through luck. Yet, if the game is a one-player game intended for repeat play, luck can be invaluable for changing the game each time. CIVILIZATION's immense replayability stems in part from the large range of strategic situations that can affect the player and that arise naturally from the vast quantity of random elements and how they interact.

Historically, one of the interesting things about games is how they become better over time, disposable games really only being children's games. People can play dominoes or bridge or chess their entire lives and the games just get better and better. Striving to make infinitely replayable games is one way to leverage the power of games.

LUCK AND MASSIVELY MULTIPLAYER GAMES

Massively multiplayer games can be competitive in the sense of player versus player, cooperative, player versus environment, and hybrid modes. For player versus player, luck functions as it does in a head-to-head game. Luck will help broaden the range of players that can compete against one another, act as an ego crutch for losses, and, if the system is well designed, generate variety of play.

In the player versus environment mode, luck will act much like it does in a solo game. In this mode, luck is unnecessary for broadening the range of players, useful as a crutch, but not as critical, and can be useful in generating variety of play.

Whether you have teammates in either mode will not really change the role of luck, except perhaps in making its use as an ego crutch a bit more important, since you can avoid losing face to your teammates by blaming bad luck.

The replayability of these games might be improved by the variety that extra luck will provide in the game mechanics. In many MMORPGs there is already a really interesting level of luck in combat. Unfortunately most of these games have reward systems that strongly discourage players from getting involved in the more interesting encounters, pushing them instead toward combat in which they are enough of a favorite to always win fairly quickly.

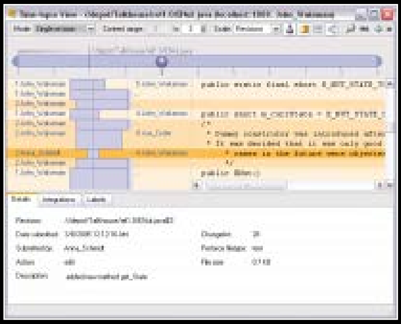
Replayability may be improved greatly if the reward system encourages players to take on challenges whose outcome is not so predictable. Since it's a staple of the genre to have the players play what are essentially the same battles ad nauseum, any potential for increasing the replayability should be of interest to the designers. In most MMORPGs it's the randomness of monster drops that provides an element of chance, and which keeps people coming back—which is a shame, since it's not directly a part of the gameplay.

APPLYING LUCK TO EXISTING GENRES

It's hard to introduce luck to an established game or genre of games. For example, shooters tend to attract players who have fast reflexes and accurate aiming; introducing luck will likely drive away established players, since they want their speed and accuracy tested. What's worse, a "shooter with luck" game will not necessarily find a new audience either, since the reputation of shooters as a genre is already established.

Although adding luck to an existing genre can alienate fans, it is often worth thinking about in terms of game design. It may be possible to add an element of luck that even the elite players will find cool. Also, if a game of this type is able to reach new audiences it could make up for the loss of some established gamers. Game designers would want to take this idea into consideration if they're working on a game that's likely to have

PHOTO COURTESY OF NAAMA YM



Real-time strategy games, such as StarCraft, can introduce luck by including it in their economic systems.

broad audience appeal, such as a game with a popular movie tie-in or one that will debut on a new platform (iPod games anyone?).

Let's look at a few established game genres and consider new ways to introduce luck. These techniques have certainly been used in one game or another, but the luck introduced is often minor or dominated by other factors, rather than really allowing for less skillful players to win from time to time.

Shooters. The expert shooter player has excellent reflexes and aiming. A lot of other gameplay elements may be present in varying degrees, such as tactics, strategy, and teamwork, but these tend to be dominated by reflexes and aiming. The introduction of more luck could be used to bring the value of these skills closer together, in addition to reducing the dominance of the better player.

Inaccuracy is one of the obvious ways to increase the amount of luck in shooters. Games such as Counter-Strike do this, but the high rate of fire and the ability of skilled players to minimize this effect make it so the aggregate game may actually have less luck rather than more. It's easy to imagine a game in which nearly every shot has a high enough variance that moving without cover into enemy fire zones, while generally a bad idea, does not ensure instant death.

Another natural way to increase luck is highly variable damage. Naturally, if you combine this with a high rate of fire, the luck you have introduced is incrementally removed. Right now, the standard way to inject highly variable damage is via head shots, which does introduce some randomness, since random sprays of bullets will occasionally yield a critical hit—but again, this is a randomness that diminishes rapidly with skill.

Randomly distributed power-ups could be another way to go. Most games have specific spawn spots for weapons and armor. If these were inconsistent, or the power-ups themselves were very swingy, it would introduce luck to the game.

Real-time strategy games. The expert RTS player has excellent massively parallel management skills and speedy clicking. Also important, of course, are both strategy and tactics; but the best strategy and tactics won't help if you can't implement them fast enough while juggling all the elements these games typically throw at you. More luck could be used to raise the relative importance of these game components, in addition to making the expert player easier to beat.

Reducing the chance of being hit by a unit or increasing the variance on its damage might be a way to increase the amount of luck in an RTS. Units could have special abilities that are completely out of the player's control and which are used inconsistently. The units could have morale that is to some extent randomly implemented so that your units might start to panic in a battle you might otherwise have won. Similarly, units might make all sorts of AI checks which are guided by the outcome, ignoring players' commands, gaining bloodlust and the inability to stop attacking, or maybe focusing entirely on their sworn enemies.

Economics is a natural place to introduce luck as well, in particular since this facet of many RTS games is often highly important and yet mostly rote in play value. Mines could give more variable payouts, and technologies could cost varying amounts or be randomly available. The expert player may even enjoy the freedom of exploring parts of the tech tree that are generally less effective rather than feeling obligated to use the same proven approach every time.

In thinking about research, players could be kept from learning what their research might yield, or when it was going to yield it. Perhaps the research could be guided a bit, without the players knowing whether the exploration of metals, for example, would yield good troop armor, good tank armor, a good conductor, or perhaps a vital ingredient for teleporters. Designers might get ideas from looking at games such as Alpha Centauri.

Racing games. Racing games reward players who have knowledge of their vehicles' capabilities, knowledge of the racetrack, and reflexes. If your opponent has you beat in these areas, you will lose every time. Increasing the luck in the game will allow the less skilled player to take higher risk strategies and thus occasionally challenge the opponent.

One way to introduce luck into this genre is to create danger zones, or areas or situations that sometimes get you into trouble, but not always. An example of a danger zone is a maximum safe speed, beyond which there is a chance of mishap or random cornering checks. Good players would know not to drive faster than the maximum for fear of a mishap—unless they're desperate. Alternatively, the further ahead you are, the more conservatively you should drive.

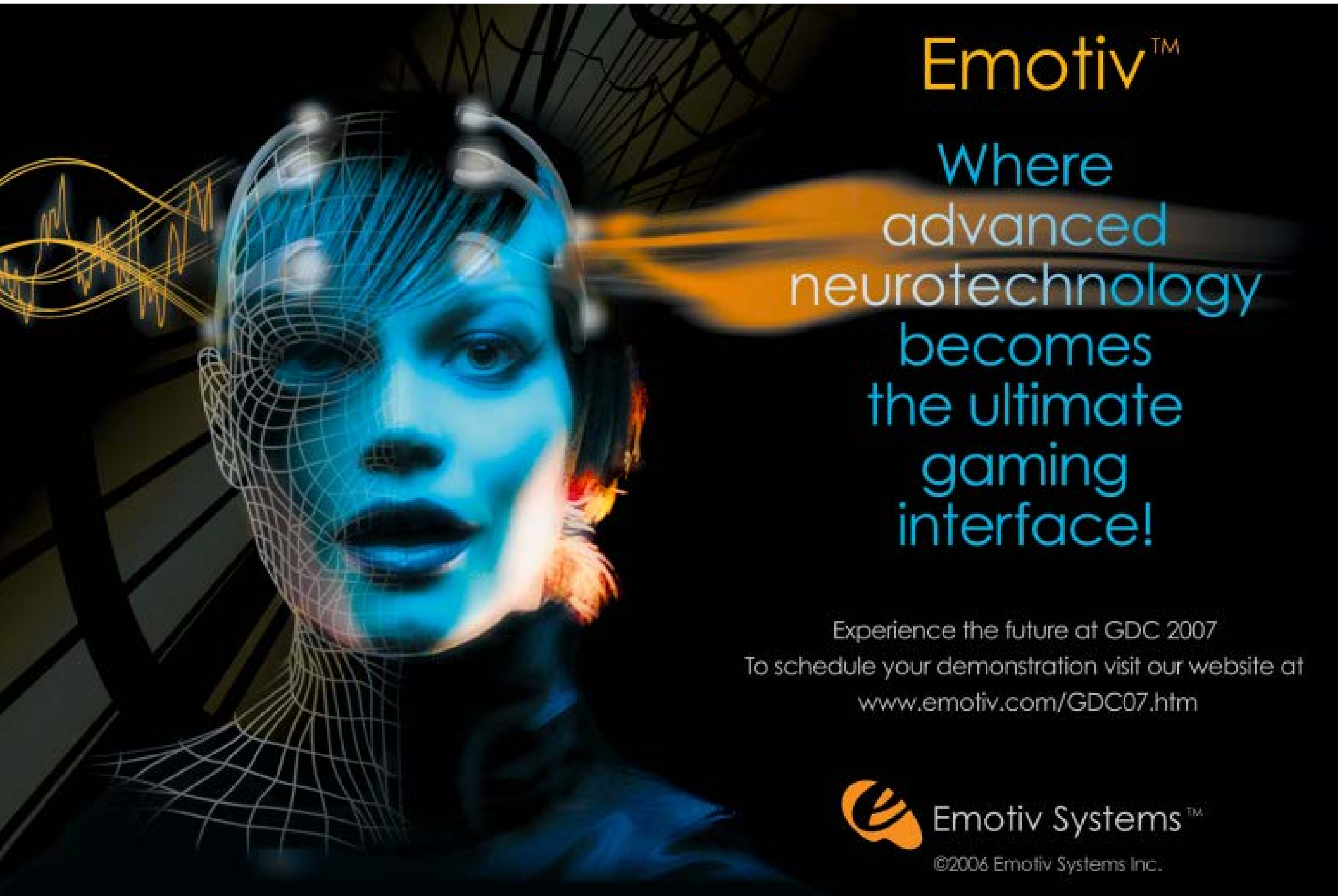
Including random, wacky power-ups is another way to add luck to racing games and some titles, like Mario Kart, already use this feature very effectively. Missile launchers, booster jets, smoke screens—bring them all!

Shortcuts that are dangerous but are also navigable by all players could be implemented to increase luck. Many games have shortcuts, but they typically only favor expert players; less advanced player can't navigate the shortcuts or don't know they exist. To increase the luck, you need something more like a chasm that saves you some time but destroys the car 20 percent of the time regardless of expertise.

FUTURE GAMES

If you're working on a project that for some reason is difficult to categorize, or may appeal to a different audience than an existing game, you might consider erring on the side of including too much luck. You'll gain the benefit of broadening your player base's competitive range while increasing your game's variety. Over time, games have a tendency to go down in luck rather than up, so you can correct your gameplay more easily in that direction in subsequent expansions and versions.

Video games are journeying into game design territory that paper games could never go. But the large body of information that exists outside electronic games can guide all game developers and help in that exploration. I am hopeful that one day I will have a collection of computer games that will handle any group of players in collective play in a manner that rivals my paper game collection.


Emotiv™
Where advanced neurotechnology becomes the ultimate gaming interface!
Experience the future at GDC 2007
To schedule your demonstration visit our website at
www.emotiv.com/GDC07.htm
Emotiv Systems™
©2006 Emotiv Systems Inc.

GDC

erik winter and matti larsson

GO WITH THE FLOW

WATER EFFECTS AND FLUID DYNAMICS IN GAMES

PUTTING CUTTING-EDGE INTERACTIVE PHYSICS INTO A GAME can help make it stand out and increase immersion. Faster processors, hardware acceleration, and new algorithms can greatly enhance not only the game's look, but also its feel. When it comes to creating these cutting-edge effects for water, it's now possible to live up to the player's expectations of how the game should look, feel, and play when the character dives into an ocean.

For our new game, CLUSTERBALL 2, we wanted to achieve the most realistic water possible. The simulated water area had to give the impression of being without boundaries, just like the real ocean. We also wanted the water to follow the larger natural movement of the sea, with waves caused by the wind. Another goal was to make the water interactive, with splashing, waves, and wakes. Finally, we wanted to capture and simulate the characteristic visual properties of water.

RENDERING WATER VISUALLY

Visually, water is very much defined by the way it interacts with light. Light rays hitting the surface are both reflected and refracted, making the surface work as a semi-transparent mirror. The rays are perturbed due to the normal of the surface and the different indices of refraction of water and air.

The Fresnel effect shows that the mirroring property of the surface is greater at a distance while transparency is greater when looking straight down at the water. Capturing these three effects in simulated water gives it a much more realistic visual appearance. Yet, the effects must be bound to the water mesh in some fashion.

One popular way to implement the effects is to use projective textures, a method that will be revised here. This solution works very well as long as the object's surface shape stays close to a plane. Two different cameras are used to render the reflection and refraction images to two textures. These textures are then blended into each other and projected back onto the water surface in a shader. Figure 1 (pg. 22) shows a simple scene in which some objects will be reflected and some refracted on the water's surface.

One camera is used to render all the objects below the surface, the ones that will be refracted, to a texture T_{refr}. This camera has the same position and direction as the main rendering camera. A second camera is used to render all that can be seen in the reflection of the water surface—only objects that are above the surface are rendered. The camera position and direction is mirrored around the surface plane to capture the same scene as the reflection. The render target is set to the texture T_{refl}.

Finally, the main camera is used to render the water. The use of projective coordinates allows the textures to be mapped back

ERIK WINTER *and* **MATTI LARSSON** *are, respectively, lead programmer and CTO of Resolution Interactive, an online games and technology company with 10 years in the business. The company's previous titles include SKI-DOO X-TEAM RACING (PC) and CLUSTERBALL (PC and mobile phone). They are currently working on CLUSTERBALL 2 (PC, Xbox 360), an online action game combining the elements of flight and racing as a sport with lots of balls. Email them at* **ewinter@gdmag.com** *and* **mlarsson@gdmag.com**.

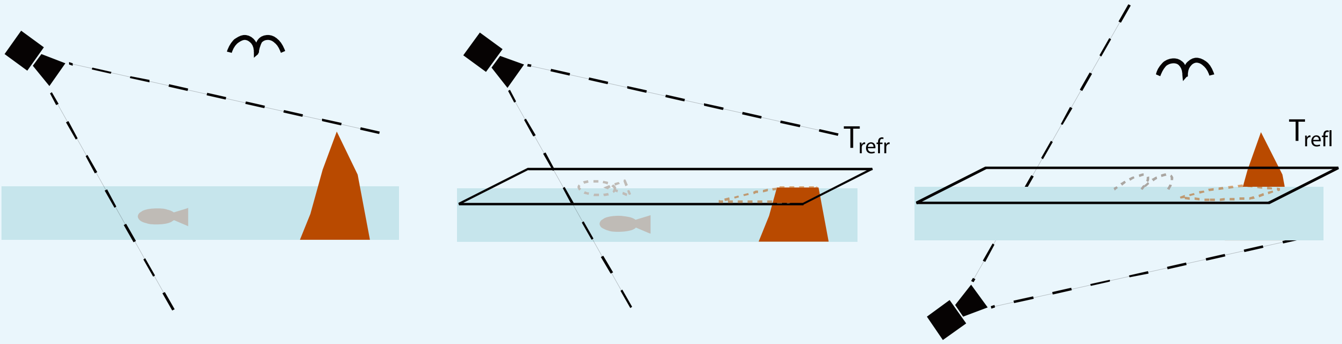


FIGURE 1 Two cameras can be used for reflection and refraction textures.

to the surface plane. The water pixel shader blends between the refracted and reflected images, following the Fresnel effect.

This technique works well when the surface is perfectly flat. But what happens when the surface changes? In Figure 2, the image on the left shows a flat surface plane with three basic geometrical objects reflected in the water. A ray of light ray is emitted from the camera until it hits the water at point P. The ray reflects around the surface normal N and, in this case, finally hits the circular object.

We imagine the texture rendered from the camera to be placed in the surface plane. The texture coordinates are applied with projective coordinates so that they match the point P where the reflection occurred.

The right side of Figure 2 shows the surface being perturbed with the normal directed a little toward the camera. The reflected ray then changes according to the new normal and hits an object a little closer to the camera, represented by the rectangle. We do this by using the same texture as the flat surface and implementing one additional trick. The rendered texture is placed as in the left image, but with a vertical offset D. The texture coordinate used for the rendering is then updated to correspond to a new point T instead of P. T is created by following the surface normal from P until it hits the displaced texture plane.

For reasons discussed later, the wind-driven waves have to be created with pretty low frequencies. To increase the realism, higher frequency waves are added in a normal map. These normals are blended with the real vertex normals in the pixel shader, before reflection and refraction calculation.

COLORING THE WAVES

Another property of water to capture is spray and foam. To account for these fragmentized effects, we implement disconnected volumes as particles. To visualize it in the mesh, we first needed a way to measure its activity. A very active surface is signified by strong waves with short wavelength. Therefore we chose to quantify the vertex activity as an interpolation depending on both the vertex height and the deviation of its normal from the surface plane normal.

As the vertex activity increases, a second, highly distorted normal map is blended onto the old one. The colors from the refracted and reflected images are replaced with green and white, respectively. Together this gives the impression of a more fragmentized surface with foam creation.

Listing 1A (pg. 26) shows an excerpt of the water vertex shader. The wave height at each vertex is calculated. As the water surface plane is the same as the xz-plane in our implementation, the wave height will be the same as the vertex y coordinate in world space. The vertex height is used to calculate a value in the interval [0,1]. This value is passed to the pixel shader to define how much the reflected/refracted images should be replaced by the colors of a fragmentized surface. The function interpol2P() is used for a linear interpolation, but with three control points instead of two. Its results are in the interval [0,1].

Listing 1B (pg. 26) shows the pixel shader. Three different normals are used: the real vertex normals, normals from the high frequency normal map, and normals from the distorted normal map. The deviation of the vertex normal from the surface plane normal is measured with the corresponding dot product. The result is stored in DistNormWeight and used to interpolate between the two normal maps. The constant VX_NORMAL_WEIGHT then blends between the normal maps and the vertex normals.

The final normal displaces the UV coordinates as described above before we sample the refraction and reflection images. We interpolate the sampled colors with the fragmentized surface colors, using the ratio calculated in the vertex shader.

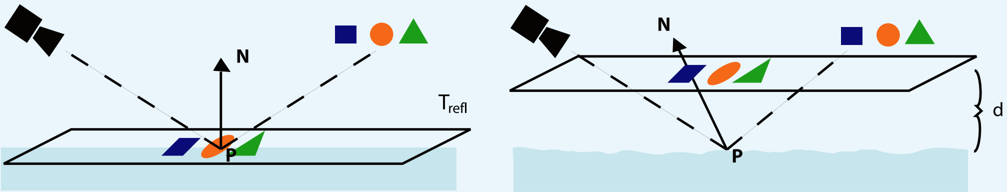
LET THERE BE HEIGHTS

The physical simulation of water has been studied extensively. Although the concepts have been well defined, their numerical solutions still require a lot of computing power.

Simulation methods are highly specialized to the nature of the liquid being simulated. For example, the simulation of milk being poured into a glass for the animations of *Shrek* might have been just as complex as the animation of the sea movement for some of the scenes of *Titanic*, using totally different algorithms. Therefore, it's necessary to choose an appropriate model for the special case.

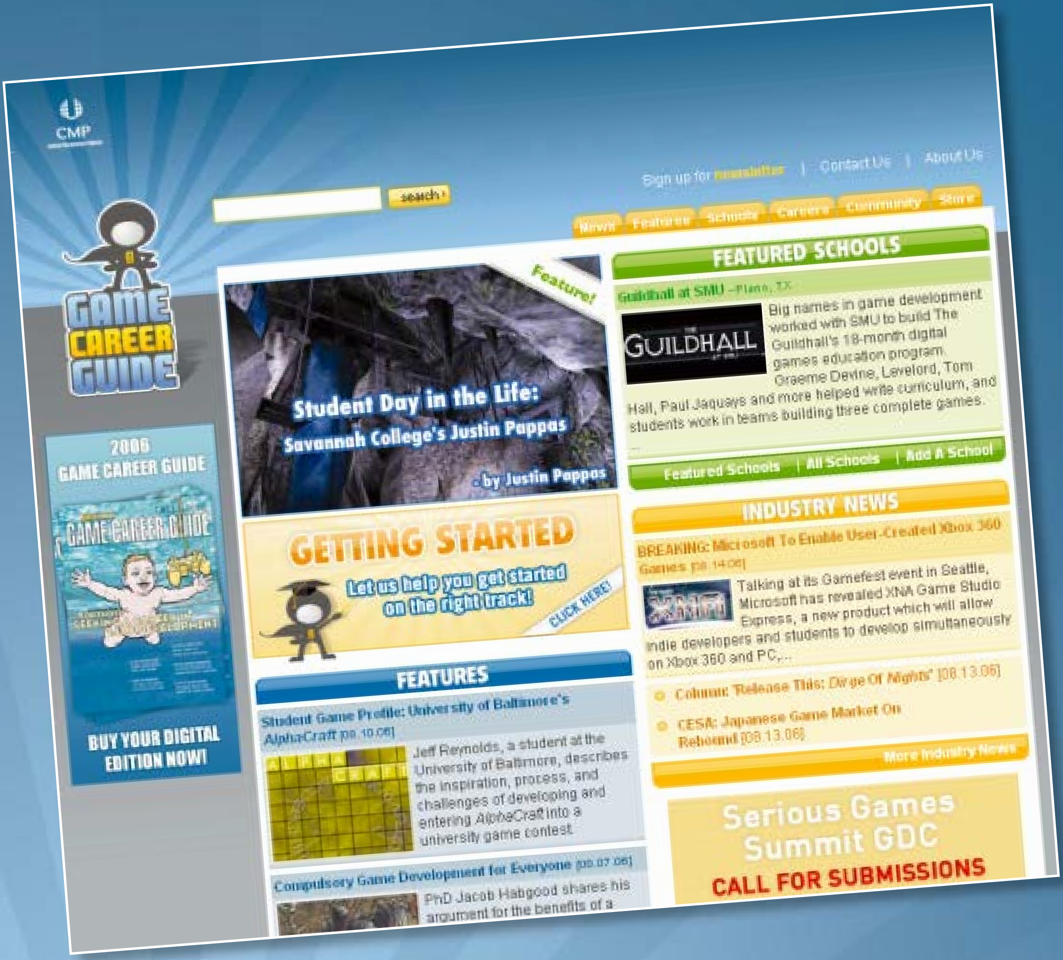
The water surface in CLUSTERBALL 2 should be able to handle the propagation of wakes and the interaction of objects falling down in the water. Because the surface is really large, we need a fast method, such as using a heightfield. Instead of

FIGURE 2 To handle reflection around perturbed normals, you can offset the sampled texture.



ANNOUNCING...

GAME CAREERGUIDE.COM



Student Day in the Life: Savannah College's Justin Pappas

by Justin Pappas

FEATURED SCHOOLS

Guildhall at SMU – Plano, TX

Big names in game development worked with SMU to build The Guildhall's 18-month digital games education program. Graeme Devine, Levelord, Tom Hall, Paul Jaquays and more helped write curriculum, and students work in teams building three complete games.

Featured Schools | All Schools | Add A School

GETTING STARTED

Let us help you get started on the right track!

INDUSTRY NEWS

BREAKING: Microsoft To Enable User-Created Xbox 360 Games [08.14.06]

Talking at its Gamefest event in Seattle, Microsoft has revealed XNA Game Studio Express, a new product which will allow indie developers and students to develop simultaneously on Xbox 360 and PC...

- Column: 'Release This: *Dirge Of Nights*' [08.13.06]
- CESA: Japanese Game Market On Rebound [08.13.06]

FEATURES

Student Game Profile: University of Baltimore's *AlphaCraft* [08.10.06]

Jeff Reynolds, a student at the University of Baltimore, describes the inspiration, process, and challenges of developing and entering *AlphaCraft* into a university game contest.

Compulsory Game Development for Everyone [08.07.06]

PhD Jacob Habgood shares his argument for the benefits of a

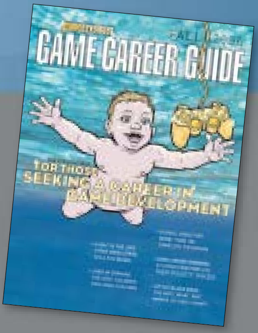
Serious Games Summit GDC

CALL FOR SUBMISSIONS

GET YOUR GAME ON

EVERYTHING YOU NEED TO KNOW TO GET INTO THE GAME INDUSTRY!

- News and features for students and educators
- Getting Started section — an invaluable how-to guide
- Faculty and student research
- Comprehensive listing of schools
- Message Boards



You can purchase your digital edition of the 2006 gamedeveloper Game Career Guide online at

www.gamecareerguide.com

CMP
United Business Media

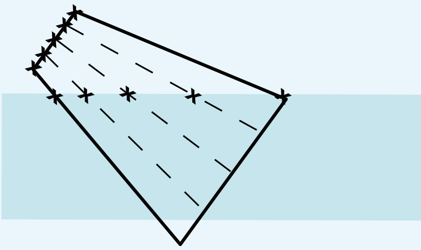


FIGURE 3 A screen space uniform grid can be projected onto the water's surface.

using a full 3D representation for the fluid, a 2D grid is used where each grid coordinate has a corresponding height. Therefore, we can exclude one dimension from the calculations.

One disadvantage to using heightfields is that a body with overhang, such as a breaking wave, can't be modelled, as only one height value is permitted at each 2D position. As long as overhangs aren't needed, the heightfield gives efficient approximations.

A variety of different heightfield methods exist. In CLUSTERBALL 2, we used a membrane-based model. Objects interacting with the surface results in the heightfield grid points being offset, causing waves to propagate from that position. Although the membrane is a fast model, it still can't be used for the whole water surface. The algorithm complexity grows with the two dimensions; thus, we must find a way to confine their size.

If we add an upper limit to the number of simultaneous interactions, it's possible to assign a membrane object to handle each interaction. These objects are instances of the physical model, restricted to a limited area. When all interacting objects have left the area, the corresponding membrane object could be removed.

The membrane method works well when describing local waves, such as a ship travelling in the water, but it's not useful when simulating bigger effects, such as wind-driven waves.

With the requirement that the simulated water should be really large comes the need for a physical representation that can give results over a large area fast. We achieve this by making sure that the model is tileable, that it can be repeated in space and time. We used Perlin noise for the wind-driven waves because of its ability to give fast and realistic-looking results.

COMBINING PHYSICAL MODELS

We still need a way to combine the results of the two physical models: the membranes and the Perlin noise. One way is to include the results from one model into the calculations of the other. Intuitively, the wind-based data should be used when calculating waves made by object interactions, such as a boat wake, which might be seen as a small perturbation on the greater wind-driven movement of the sea.

Is there a way to completely avoid this model interaction? A straightforward solution is to simply let the models be superimposed over each other. However, care must be taken when performing this drastic simplification. For a realistic visual impression, it's important that the models' precedence to each other follow our expectations.

We created the wind-driven waves with low frequencies and low amplitude. That way, they are clearly separated from the high and steep ship wakes, making the superimposing a valid approximation.

But there's an instance when the models can't be kept separated: when calculating the water's influence on interacting objects. The interaction is defined in the object interaction model, but many of the calculations require the current water height. For example, buoyancy is dependent on how much of the object is covered by water. This means that whenever water height is needed in the calculations, it has to be updated with data from the wind-driven model.

This sharing of information shouldn't affect performance, as there are a limited number of interacting objects.

MESH TESSELLATION

In CLUSTERBALL 2, we wanted to have a really large ocean surface—so large that it would seem infinite. The mesh representing the ocean surface had to give the impression that such a large area is covered. To be able to represent all local effects, the mesh resolution still has to be close to the camera, so we need some kind of level-of-detail technique.

That the whole mesh should be animated to follow the wind-driven waves results in another problem. Every mesh vertex has to be updated each frame, so it's extremely important to maximize the vertex efficiency (the number of mesh vertices that finally show up on the screen). How might the camera's viewport be used to achieve this goal?

We focused on the projected grid algorithm (see Resources). Its algorithm is well constructed to minimize the time it takes to construct the mesh, while still attending to some of the problems inherent to a viewport-based approach.

The major drawback of using these approaches is that the mesh tessellation has to be restructured for each frame update. Performance can be improved if the restructuring is brief compared to the extra time caused by calculating height data at unnecessary vertex positions.

PROJECTED GRID

One of the goals of the projected grid concept was to create a mesh that, when transformed to screen space, would be as close to a uniform grid as possible. Such a mesh automatically has a continuous LOD handling; the mesh has a higher resolution near the camera, as can be seen in Figure 3.

To maintain high vertex efficiency, the mesh boundaries need to be as close to the screen boundaries as possible. When finding these boundaries, we must be careful to include all parts of the surface that could possibly be seen in the camera.

As the mesh height can vary along its surface normal, it's not

projected grid algorithm

1. Find the boundaries of the surface on screen, using the rendering camera's frustum. This means, find all points where the camera frustum intersects S_{lower} or S_{upper}. Add these points to a buffer. If any of the frustum's corners are within $V_{displaceable}$, add them as well. If no points were found, then none of the water surface is within the camera frustum and the rendering can be aborted.
2. Aim the projecting camera so backfiring is avoided.
3. Project the buffer points onto S_{base}.
4. Transform the projected buffer points to the projecting camera's space.
5. Find the maximum span in world space of the buffer points along the projecting camera's x and y axis. Construct a matrix M_{range} that scatters screen coordinates to this span.
6. M_{range} is used to transform the mesh corner points to the world space mesh span. If homogenous coordinates are used all other vertices can interpolate their world space positions from these four points. Thus a per-vertex matrix multiplication is avoided.
7. Finally, height data from the physical models is used to displace the vertices vertically.

—Courtesy of Claes Johansson

enough for the boundaries to include the surface plane—they must also include the whole volume spanned by all possible vertical positions of the mesh.

To enclose this volume the water surface plane is defined as S_{base} and its normal N_{base}. If the waves are constricted to a maximum waveheight H_{max}, then two planes can be used as limiters of the variations:

$S_{lower}=S_{base}-H_{max}*N_{base}$

and

$S_{upper}=S_{base}+H_{max}*N_{base}$.

All possible vertical positions of the mesh should then be within $V_{displaceable}$, the volume spanned by these two planes. The constructed mesh must include all parts of the screen where $V_{displaceable}$ can be seen. Furthermore, we must be sure to account for how much of the screen should be covered by the water surface. When looking down at the water, the whole screen should be covered. When looking toward the horizon, only a portion of the screen should be covered. This puts two demands on the mesh. First, it should be possible to represent all conceivable vertex heights, and second, the horizon should be used to limit the mesh on the screen.

How should we calculate the right span of the mesh? Figure 4 gives some hints. Every part of the surface that can be seen in the camera must be represented in the mesh, which includes the intersection of the camera frustum with all possible vertical heights of the surface (the volume $V_{displaceable}$).

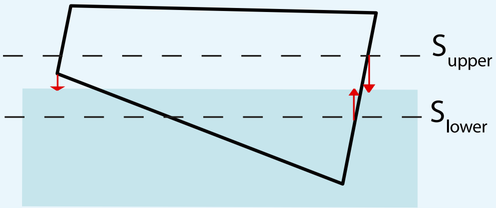


FIGURE 4 Here, the camera frustum intersects with $V_{displaceable}$.

Every point that's necessary for defining the mesh is stored in a buffer. The maximum world space area, including these buffer points, is used as the mesh span. Which points should then be regarded as necessary?

On the right side of Figure 4, a line is drawn between the far plane's corner points. The intersection point between the line and the upper plane signifies the rightmost position a portion of the water can have within the camera frustum. All intersections between the boundary planes and the frustum lines should therefore be stored in the buffer.

The left part of Figure 4 shows another scenario. The boundary plane intersections are not enough because one of the frustum corner points falls between the planes and will itself represent the leftmost position. As a result, all frustum corner points within $V_{displaceable}$ have to be stored in the buffer as well.

Constructing a screen space uniform grid will settle the x and y coordinates of each vertex on the screen. Still, it says nothing about the z coordinate, the depth in the image, which remains undefined. This means that each point on the screen will correspond to a line in world space. The object of the

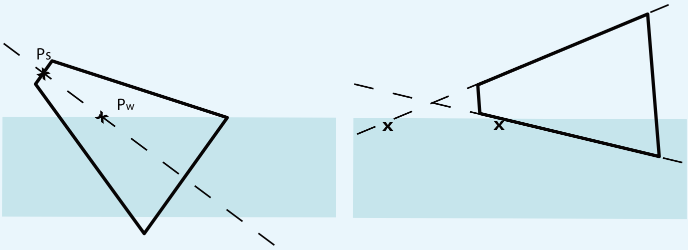


FIGURE 5 The left image shows a screen coordinate and its corresponding mesh vertex while the right image shows a camera backfiring.

mesh represents the water surface, and the final world coordinates of the mesh vertices will be where these lines intersect the water plane.

In Figure 5, point P_S is chosen, and a line is drawn along all its possible z-values. The final corresponding mesh vertex will be at P_W, where the line crosses the surface. But a problem arises in the right image when the camera is no longer directed down toward the surface.

Two points on the screen are chosen: one on the upper edge and one on the lower. Their corresponding lines should then follow the camera frustum. The lower line crosses the water surface in front of the camera as before, but for the upper line, no such intersection exists. The intersection is instead behind the camera, as a consequence of the underlying mathematics. The camera is said to have backfired.

The resulting mesh should be spanned by all intersections of lines from the screen points and the surface. The area spanned goes toward infinity in both directions, clearly not the behavior we wished for.

The proposed solution in the projected grid algorithm is to use two cameras: a regular rendering camera and a projecting camera. This camera is used when constructing the mesh. It inherits its position and direction from the rendering camera, but with the ability to adjust them to avoid backfiring. The camera is always aimed at the surface of the water and keeps the projecting camera outside of $V_{displaceable}$. (See the sidebar "Projected Grid Algorithm.")

PARTICLES AND ACCELERATION

We chose a mesh-based model as the main simulator for the water system. The mesh representation restricts the water from breaking into smaller pieces, such as when a large wave creates spray, which we implemented by adding a particle system.

Particles are created when the vertical velocity of a vertex gets too high. The particles should be created with a direction following the vertex normal and a velocity depending on the vertex velocity.

From which of the mesh representations should the particles spawn? Using the projected grid's vertices would be the correct choice as this is the representation presented to the user. However these vertices will not be statically located, so finding the velocity of a fixed world position is not possible.

The Perlin data could be tested but this would be consuming, as particles have to be emitted for each noise tile. However, since the energy of the wind-driven waves already has been chosen as low, the waves will be small and won't spray. It's enough to check the velocities in the object interaction grids and create particles from them.

CONTINUED ON PG 28

LISTING 1A water vertex shader (VS 2.0)

```
float interpol3P(float x, float minHeight, float middleHeight, float
          maxHeight, float weightMiddle  ){
   if(x > maxHeight )
      return 1.0;
   else if(x > middleHeight )
      return  (x-middleHeight)*(1-weightMiddle)/(maxHeight-middleHeight)
          + weightMiddle;
   else if(x > minHeight )
      return (x-minHeight)/(middleHeight-minHeight) *weightMiddle;
   else
      return 0.0;
}

VS_OUTPUT vs_main( VS_INPUT In )
{
   VS_OUTPUT Out;
   ...
   // Transform vertex position from object space to world space
   float4 PositionScaleWS = mul( matWorld, In.Position );
   // Transform from homogenous coordinates
   float3 PositionWS = PositionScaleWS.xyz * PositionScaleWS.w;

   // Find a measure on how distorted the water surface should be,
          depending on vertex height
   // Out.col.x will be used to send the value to pixel shader
   Out.col.x = interpol3P(abs(PositionWS.y), WH_CTRL_PNT_1, WH_CTRL_PNT_2,WH_MAX,
WEIGHT_CTRL_PNT_2);
   ...
}
```

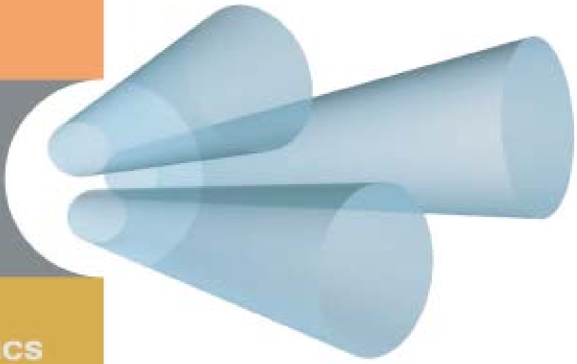
LISTING 1B water pixel shader (PS 2.0)

```
   ...
   // High frequency normals from normal map
   float3 HFNormal = tex2D( NormalMap, In.NormalMapUV );
   // Convert color values to vector
   HFNormal = ( HFNormal - 0.5 ) * 2;
   // Highly distorted normals for fragmentized surface
   float3 DistNormal = tex2D( NormalMapDist, In.NormalMapUV );
   // Convert color values to vector
   DistNormal = ( DistNormal - 0.5 ) * 2;
   // interpolation between normal maps due to strength of real normal
   //The water surface normal is aligned with the z axis
   //The dot product of the vertex normal and surface normal can then be simplified
to inNormal.z
   float DistNormWeight = smoothstep(NRM_INTPOL_MIN, NRM_INTPOL_MAX, In.Normal.z);
   float3 Normal =  lerp(In.Normal, lerp(DistNormal, HFNormal, DistNormWeight),
VX_NORMAL_WEIGHT);
   ...
   // Projective coordinates are used to sample the refracted/reflected images
   float4 RefractionColor = tex2Dproj( RefractionMap, In.RefractionUV );
   float4 ReflectionColor = tex2Dproj( ReflectionMap, In.ReflectionUV );
   //
   RefractionColor = lerp(RefractionColor, DistRefrCol, In.col.x);
   ReflectionColor = lerp(ReflectionColor, DistReflCol, In.col.x);
   ...
   // The final pixel color is blended from reflection and refraction colors
   float4 TotalColor = lerp( ReflectionColor, RefractionColor, Fresnel );
   ...
```

DAVID H. EBERLY

SERIES IN INTERACTIVE 3D TECHNOLOGY

3D GAME ENGINE DESIGN



A PRACTICAL APPROACH TO REAL-TIME COMPUTER GRAPHICS
SECOND EDITION

BY DAVID H. EBERLY

November 2006. ISBN 978-0-12-229063-3

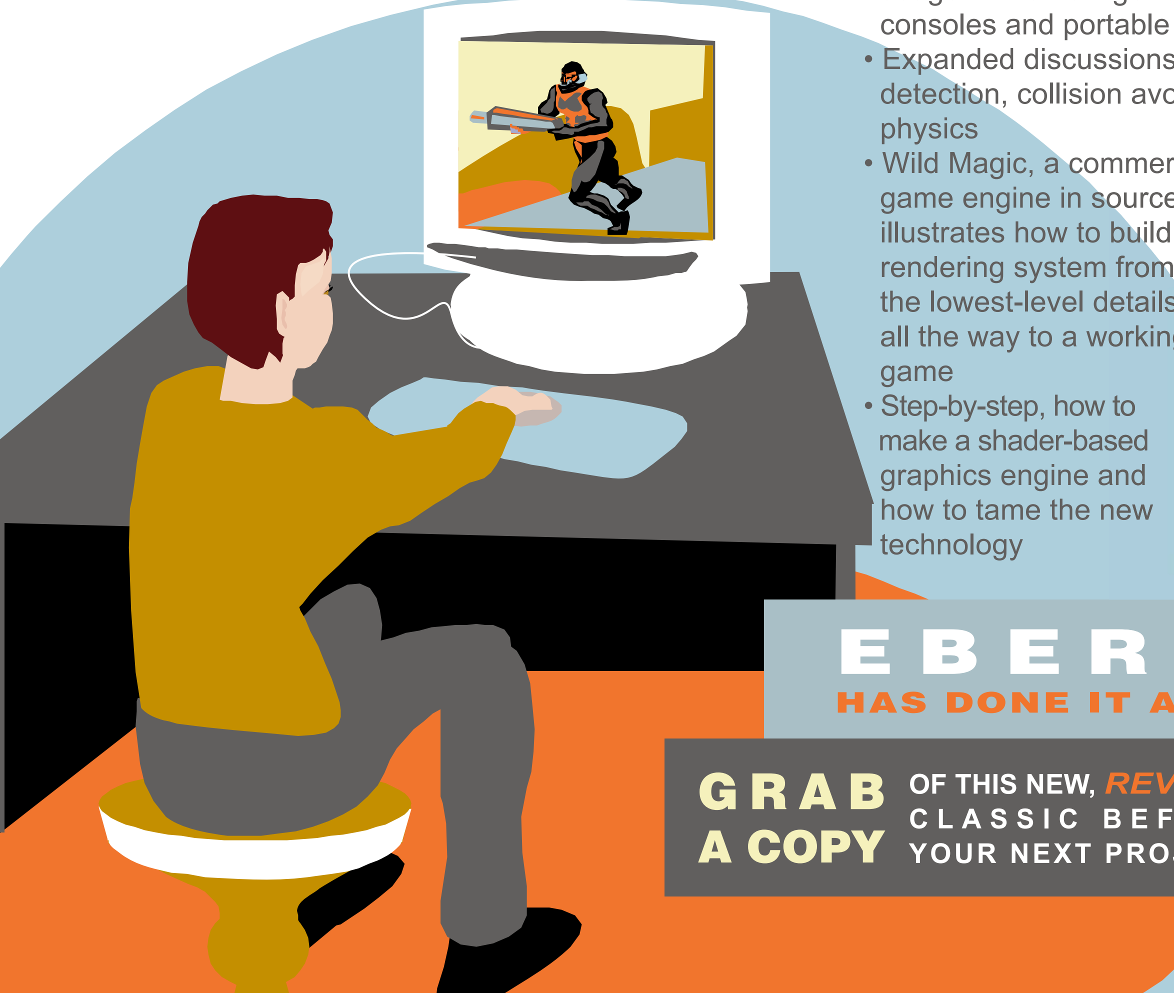
LEARN ABOUT HOW WILD MAGIC WILL HELP YOU GET STARTED ON YOUR OWN ENGINE TODAY!

SIMPLY ORDER ONLINE AT **WWW.MKP.COM/GRAPHICS** OR BY PHONE AT **800.545.2522**

Price: $69.95

This new edition includes:

- More than twice the coverage of the essential techniques of scene graph management
- New methods for managing memory usage in the new generation of game consoles and portable game players
- Expanded discussions of collision detection, collision avoidance, and physics
- Wild Magic, a commercial-quality game engine in source code that illustrates how to build a real-time rendering system from the lowest-level details, all the way to a working game
- Step-by-step, how to make a shader-based graphics engine and how to tame the new technology



EBERLY HAS DONE IT AGAIN!

GRAB A COPY OF THIS NEW, *REVISED* CLASSIC BEFORE YOUR NEXT PROJECT.

CONTINUED FROM PG 26

The particles are rendered with a simple texture, alpha blended into each other. As they represent small pieces of fragmentizing spray there's no need to attach any higher demanding effects, such as reflections. The size and alpha value of the rendered particles varies with their lifetime, becoming smaller and more transparent to mimic dissolving.

One important decision to make is how the particles' physical properties should be updated. We want particles to move due to forces like gravity, and bounce when they collide with rigid bodies. But should the particles be able to affect each other, too? If these particle-particle interactions could be ignored, many fast implementations are possible.

The results of such a particle model will not be realistic when the particles move slowly or close to each other. Effects like surface tension need the particles to share information with each other.

Other models (in which inter-particle forces are used to update the particle positions) tend to be too slow when the particle system becomes large. A way to circumvent this problem is to use hardware-accelerated physics. For CLUSTERBALL 2, we looked into using Ageia's physics processing unit (PPU).

Many suggestions have been made as to how physical algorithms can be implemented and accelerated in the GPU. Both basic algorithms such as collision detection and, more interesting, water simulation methods have been discussed as suitable for this. Ageia's solution is to use dedicated physics hardware, adapted to accelerate physics calculations. The PPU works in many ways like the GPU, with several processing elements performing calculations in parallel. One important difference in physics algorithms is the need to access other objects' data; the PPU has been constructed with an emphasis on high inner bandwidth.

The inter-particle interaction model used in Ageia's PPU is smoothed particle hydrodynamics (SPH). The technique was first developed for use in astrophysics when studying interstellar gas flow. It has become popular for simulating other highly deformable materials such as liquids or sand.

RESOLVING SAMPLE ARTEFACTS

With two different representations of water, the graphical model has to sample its data from the physical. Care must be taken to avoid the classical artefacts combined with sampling. A thin, strong spike in the physical representation can fall in between the mesh vertices if the mesh is too sparse. If the vertices manage to represent the spike at some instant and miss it in the next, the mesh will start to flicker.

To avoid this problem, the mesh resolution should be kept as high as possible, which has to be weighted against the extra time cost for more vertex calculations. The wavelength of the physical representation should be kept small, which can be arranged for the wind-driven waves. But as the wavelength increases, the realism does, too. The waves caused by object interaction will always be small and strong, thus putting even more pressure on the problem.

Another way to soften the appearance of the problem is to smooth the mesh. Every vertex height is filtered by its neighbors, so sudden differences will decrease. The smoothing will decrease the flickering, but in the meantime, it may erase those sudden, steep waves that we want to see.

The projected grid provides a mesh on which the resolution decreases with a growing distance to the camera. This means that the sampling problem will be even more noticeable at far distant parts of the mesh.

The step length between two adjacent grid points could thus be used as a measure of the probability that sampling artefacts will occur. To hide the artefact, the vertex heights are multiplied by a smoothing factor, which reaches zero when the grid step exceeds a predefined maximum value.

EXTENDING THE MODEL

As proposed, the wind-driven waves will have pretty low frequencies. For now, a normal map has been used to add higher frequency waves. Because the normal map is a non-animated texture, the waves sometimes add a static impression to the water surface, especially when the camera is far away from it. A better visual result could be achieved if the normal map was animated too, by the use of high frequency noise. For an efficient implementation, you could generate the noise in shaders.

The methods and code samples described in this article have been implemented using vertex and pixel shader 2.0. With shader model 3.0, we now have the possibility to sample textures as early as in the vertex shader. This means that a height map could be sampled to offset the vertex heights.

The mesh in this article has to be high-resolution, and the underlying representation has to be sampled in each vertex, which leads to a high time cost. You could avoid this problem if the wind-driven waves were sampled and superimposed in the vertex shader. The noise height map should then also be constructed in the shader.

However, when calculating the interaction between objects and the water surface, the total water height must still be accessed. The height map should be available in the main application as well. But these queries are not at all as frequent as the per vertex samplings, so performance should still improve.

The water simulation method presented in this article works well as long as the number of concurrent interactions can be minimal. The time it takes to update the mesh-based simulation grows proportional to the number of active membrane instances.

The per-vertex cost of the final implementation proved to be relatively high. The chosen tessellation scheme was a good choice as it tries to keep a high ratio of mesh vertices within the camera frustum. ⁙

RESOURCES

Water shader sample code from ResolutionInteractive.com:
www.resolutioninteractive.com/watereffects

Rendering using projective textures:
Lombard, Yann. "Realistic Natural Effect Rendering: Water I," GameDev.net, September 7, 2004.
www.gamedev.net/reference/articles/article2138.asp

Projected grid algorithm:
Johanson, Claes. "Real-time water rendering—Introducing the projected grid concept," Master's thesis, Department of Computer Science, Lund University, Sweden, March 2004.
http://graphics.cs.lth.se/theses/projects/projgrid/

Ageia PhysX development:
www.ageia.com/physx

Noise displacement in vertex shader:
Kryachko,Yuri. "Using Vertex Texture Displacement for Realistic Water Rendering," in *Game Programming Gems 2*, Hingham, Mass.: Charles River Media, 2005.
http://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch18.pdf

Did You Know?

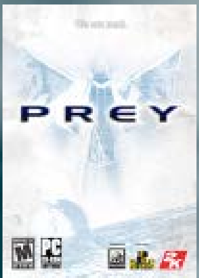
- New prospective Hires often use MobyGames to review key Team member BEFORE accepting a new position.
- Each and every contribution is Reviewed by one of more MobyGames Approver to ensure Accuracy and Correctness.
- 100% of MobyGames is User contributed.
- Each and every contribution is review by one or more MobyGames approver to ensure accuracy and correctness.

Make sure your gameography is correct.
Update your profile TODAY.

Moby Games™

The Largest Games Database on the Planet

GAME DATA



DEVELOPER
Human Head Studios and (produced by) 3D Realms; 360 version by Venom Games

PUBLISHER
2K Games

NUMBER OF DEVELOPERS
30

DEVELOPMENT TIME (FROM NEW CONCEPT TO SHIP)
5 years 2 months (May 2001–July 2006)

LINES OF CODE
About 750,000

RELEASE DATE
US: July 11, 2006
Worldwide: July 14, 2006

PLATFORMS
PC, Xbox 360

POSTMORTEM

CREEPING DEATH

DESIGNING THE DEATHWALK SYSTEM IN 3D REALMS' AND HUMAN HEAD STUDIOS' PREY

DEATH IS AN INEVITABLE PART OF MOST VIDEO GAMES. UNLIKE IN REAL LIFE, though, death in video games is hardly permanent. The player simply has to insert another coin, restore a saved game, or restart from a checkpoint to continue playing.

While designing PREY, we at Human Head wondered if there was another way to deal with death. Could we handle death in a less jarring manner than stopping the action and flashing a large, red GAME OVER? Is there a way to design the concept of death into the game that actually makes it part of the overall experience instead of something that inhibits the player from proceeding?

The game system we created, known as DeathWalk, was intended to reduce the negative impact of dying and make it a deeper part of the overall play experience.

THE HISTORY OF PREY AND THE DEATHWALK SYSTEM

Before delving into how we handled dying in PREY, here's a bit of history about the game itself.

PREY was originally conceived by 3D Realms around 1996 as an internal project. Prey went through various incarnations until it was put on a back burner in 1999. A year later, after Human Head Studios released RUNE, 3D Realms contacted us about partnering with them on a project, similar to the relationship 3D Realms and Remedy had during the development of MAX PAYNE.

After discussing various ideas, we eventually decided to resurrect PREY in May of 2001.

While designing PREY, we tried to keep the game as immersive as possible. In fact, that idea became one of the core design fundamentals of the game, leading to concepts such as keeping the game in a first-person perspective at all times. We used no cinematics in the final product, and the player's control is never taken

*CHRIS RHINEHART is a 12-year veteran of the game industry, having previously worked on titles such as HERETIC and HEXEN. A co-founder of Human Head Studios, he was a programmer on RUNE, and most recently project lead on PREY. Email him at **crhinehart@gdmag.com**.*

The PREY development team at Human Head.

away without a logical reason. We also kept the heads-up display simple and iconic so as not to distract the player's immersion. DeathWalk was of course a primary element in helping to maintain a high level of player engagement.

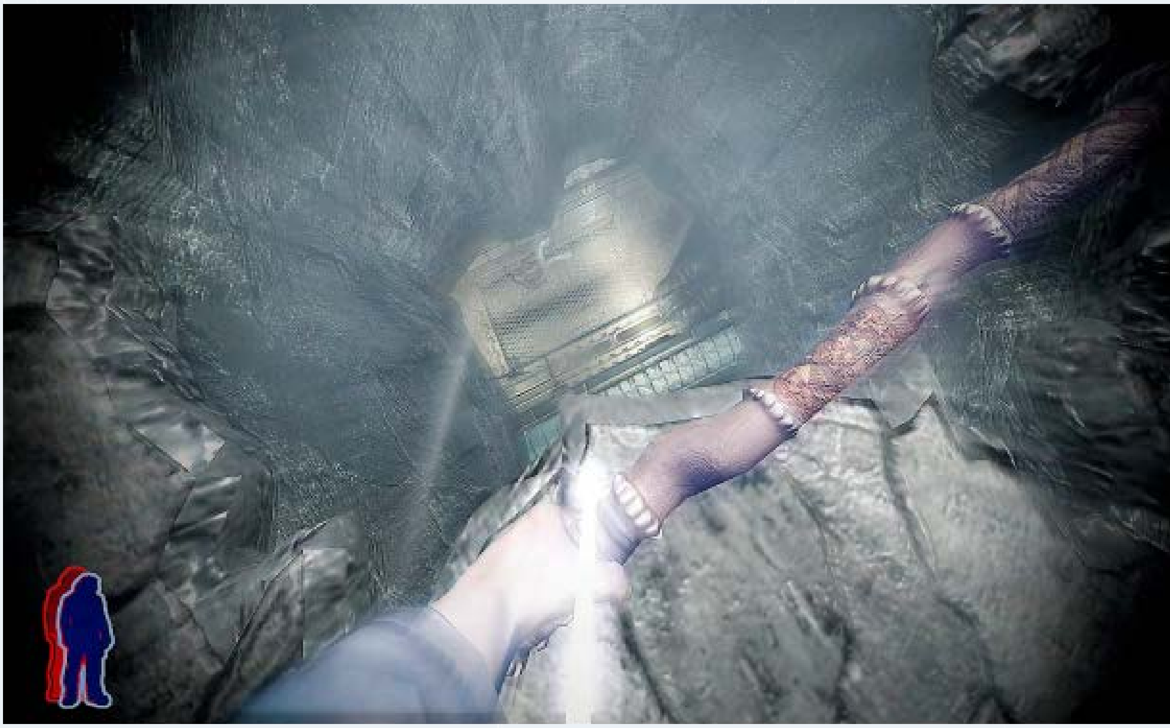
DeathWalk is a system in PREY that prevents the action from stopping just because the player-character dies. Upon dying, instead of resuming the game at some predetermined position, the character is transported to a dark underworld where he or she has a limited amount of time to fight creatures (called DeathWraiths), which restore the character's health and spirit power. After the set amount of time expires, the ground opens up and slowly pulls the character back into the physical world, where he can continue playing exactly where he died. This mini-game takes between 15 and 20 seconds on average, depending on how well the player avoids getting pulled back into the physical realm.

THE EVOLUTION OF DEATHWALK

DeathWalk wasn't in our original design for PREY. The idea didn't come about until early 2003. At that point, we were discussing various ways to make the game more immersive. The intriguing concept of being able to somehow resurrect a dead character to give the player another chance was mentioned, and the initial concept for DeathWalk was born.

The first step was to prototype the system to see if it even worked. We worked through at least five major iterations of DeathWalk before we discovered what we were looking for.

First take. The original implementation of DeathWalk didn't have an underworld. When characters died, their bodies dropped to the ground, but they continued to exist in a ghostly form where they could walk around but not interact with anything else in the world. After a short time the DeathWraiths would swoop into existence, to attack the character and steal spirit power. If the DeathWraiths reduced the spirit power to zero, the character would die permanently—death in the conventional

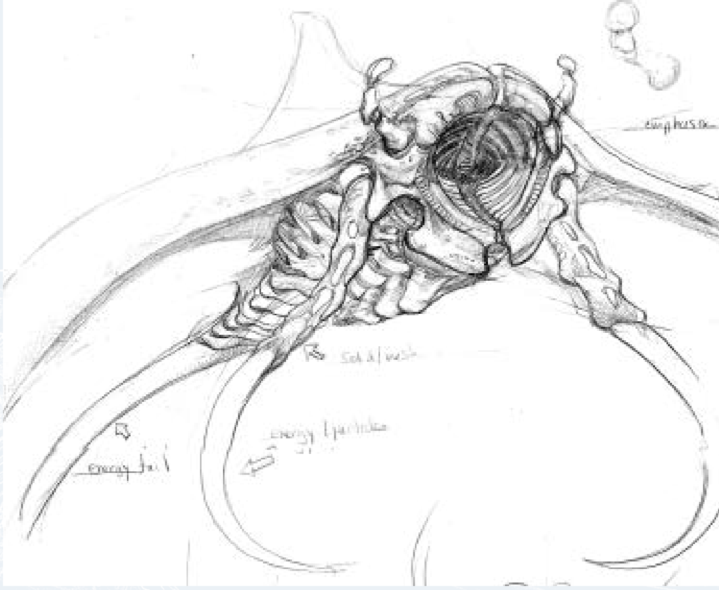


video game sense—and the player would have to reload from a saved game.

The player's mission was to fight the DeathWraiths (using a specific weapon available when dead) and gain spirit power for each kill. Once the character's spirit power was full the player could be resurrected on the spot where he first died.

Ultimately, we rejected this first version due to the unpredictability of where the character could die. Fighting DeathWraiths was fun in wide-open spaces, but was frustrating in tight corridors or other unmanageable spaces.

Take two. The second version solved the problem of the unpredictability of death location by introducing the underworld. After dying, the "dead" character was pulled into an underworld arena to battle the DeathWraiths. The same criteria for winning and losing carried over: The character had to fill up her spirit power to win, whereas running out of spirit power meant losing.



A sketch of a DeathWraith.

In this second version, we also experimented with the character having to navigate to the center of the arena, where her spirit power would slowly recharge. When the character was not in the center, her spirit power would slowly diminish. Meanwhile, the DeathWraiths continued to swoop in, stealing spirit power and pushing the character out of the center.

We rejected this version due to the confusion about the slow ticking of spirit power (as testers would suddenly lose without much warning). This was also rejected due to the passive nature of the gameplay. Standing in place and waiting for a meter to fill up just wasn't very exciting.

Take three. We designed the third version of DeathWalk around two stages: In the first stage the character's dead body was also in the underworld, floating in the air, and slowly rising up into a glowing cone of light from above. Once the body reached the top of the cone, the player would lose.

The objective of this stage was for players to rescue their bodies by shooting DeathWraiths, which would energize the body and drop it closer to the ground. Once the body reached the ground, the second stage of this DeathWalk triggered. The world shook and a hole leading to the physical world opened up in the ground. A whirlwind formed which would gradually pull the body and the player character into the hole, back to the land of the living.

Next, the whirlwind continued to pull at everything in the underworld with increasing force. During this stage, shooting a DeathWraith would give the character a little bit of health. The objective was to shoot as many wraiths as possible before getting sucked back into the physical world. Players who shot no DeathWraiths, would resurrect with one point of health, although this was later changed to 25 percent and then 50 percent health because only giving a single point of health caused many testers to die repeatedly in certain circumstances.

This version was the first time we really felt the DeathWalk mechanic would actually work well in the game. It was rejected, though, because of the negative feedback loop introduced by the rising body. The more wraiths the player missed, the more time passed, causing the body to continue to rise higher, meaning that many more wraiths had to be hit.

Take four. In the fourth version, we made some major changes to the mechanic of the body rising and to how the DeathWraiths functioned. This time, the body started high and slowly moved down toward the ground. Once the body reached the ground, the hole opened up as usual. In previous versions, one of the goals was to save the body by fighting the DeathWraiths. However, in this version, because the body automatically lowered to the ground, the whole sequence was essentially on a timer, making the goal to kill as many DeathWraiths as possible in the short time allowed.

We also changed the DeathWraiths by introducing a second type that gave spirit power when shot—so this time the player simply had a short amount of time to hit as many wraiths as possible to fill up the character's health and spirit reserves before getting pulled back into the physical world.

FINAL VERSION

A few tweaks had to be made to the gameplay mechanic, specifically when dealing with death pits. We couldn't have the character resurrect at the bottom of the pit. To fix this issue, we created death zones and resurrection locations, a place that the character would return to if they died in a given death zone. This seemed to be a pretty simple solution that required a bit more design work to place and test to ensure the character would resurrect in a decent location.

Another tweak we made was to give the character a few seconds of invulnerability when they return from DeathWalk. Characters often were killed in a firefight, so a few seconds of

invincibility gave them a chance to flee for cover or gain the upper hand in the battle.

Finally, we constructed three different DeathWalk arenas; but the differences are primarily cosmetic, as the gameplay is essentially the same in all the arenas.

WHAT WENT RIGHT

1 **INCREASED IMMERSION.** Since the character continued to play even after dying, PREY became a continuous experience without the jarring "game over" interruptions that mentally knock the player out of the flow of the game.

Additionally, in most games, when a character dies, the player often decides it's an opportune moment to take a break from playing. However, in Prey, death is handled as more of a state change than a do-over, and game players tend to continue playing because they mentally stay inside the game world.

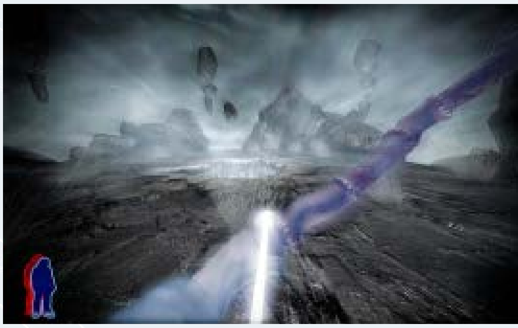
2 **REDUCED THE FRUSTRATION OF DYING.** Dying can be an exercise in frustration in games, especially if the character dies over and over in the same location trying to get past a particular puzzle or firefight. Repeatedly dying in the same place can result in the player quitting the game out of frustration—sometimes never to return. What's even more frustrating is when players die only to remember that they last saved the game more than an hour ago.

The DeathWalk mechanic helps reduce the frustration by not forcing players to start the puzzle or fight from the beginning; they can resurrect and continue where they left off. This is especially important to non-hardcore players who are more interested in enjoying the experience and finishing the game than getting beat down around every corner.

Additionally, players can relax, not worrying about when they last saved the game, as DeathWalk alleviates the need for constant save crawling.

CONTINUED ON PG 36

DeathWalk in action.



CONTINUED FROM PG 35

PREY's protagonist, Tommy.

3 SOMETHING NEW IN THE GENRE. Quite simply, this type of mechanic hasn't really been implemented like this in an action first-person shooter before. Most FPS games deal with death in the same way, so it was a refreshing change to play a game with a different system for death. We had a number of testers report that once they became accustomed to DeathWalk in PREY, they started to miss having this feature when they played other FPS games.

4 TIME AND FREEDOM TO EXPERIMENT. Because we were working with 3D Realms and they have an open approach to game design and a "When It's Done" philosophy on release dates, we had the freedom to experiment with alternative game design ideas, as well as the time to prototype and refine the ideas. Without that freedom, we wouldn't have been in the mindset to develop new ideas in the game. Without the time, we never would have been able to go through as many experiments, revisions, and tweaks as we did.

5 A DEDICATED AND OPINIONATED TEAM. Throughout development, we encouraged team members to bring up their criticisms about any part of the game. This generated a lot of discussion about various aspects of the game, ranging from music to art to story.

DeathWalk was no exception, as people brought up suggestions quite often about the look and the gameplay of the mini-game. This feedback was always taken seriously, and it quite often generated new ideas and directions for the prototyping of DeathWalk. Without the feedback of the team, we wouldn't have explored as many different variations of the DeathWalk mechanic before deciding on the final version in the game—nor would the look of the area be as polished as it turned out to be.

WHAT WENT WRONG

1 DEATHWALK MADE THE GAME TOO EASY FOR SOME PLAYERS. Because dead characters come back right where they died, the game became too easy for some players. Experienced gamers could just push through the game, die and come back quickly to finish the fight.

This is especially true during boss battles, when a player can fight as long as possible, die, and come back repeatedly until the boss is defeated; the boss' health does not reset when the character dies.

2 REDUCED PLAYER CONCERN ABOUT DEATH. Some testers reported that death no longer felt like a penalty and they no longer feared dying in PREY. This was one of our concerns throughout the game's development and something we watched closely in the reactions of the focus testers and of the team.

It's interesting to note that opinions on this were split. Some testers reported that death felt just as much a penalty as in other games, while others reported that DeathWalk reduced the impact of dying. In certain circumstances, we noticed, it was advantageous to die. For example, if players are low on health, they can kill themselves, and use DeathWalk to regain even more health.

3 MADE FOR A SHORTER GAME. One of the major comments we've heard particularly from experienced and hardcore first-person shooter players is that the game is rather short. This happened for various reasons, but one of them is DeathWalk.

Had we gone the route of conventional FPS games, the player (upon getting killed) would have had to reload a save game or restart from a previous checkpoint, either of which would have required the player to replay a section of the game, possibly a large section. But with the DeathWalk mechanic, players continue from right where the character died, without retracing their steps. Because we didn't require players to repeatedly play sections over again until they solved them without dying, the overall game time was reduced.

4 NOT ENOUGH VARIATION IN DEATHWALK. The gameplay mechanic of DeathWalk is pretty simple and was intended to be like a mini-game. While this mechanic worked well, it can become a bit tedious later if a character dies repeatedly throughout the course of the game. It would have been interesting to expand upon the mechanic and provide more variation in the objectives while still maintaining the original goal of keeping DeathWalk quick and fun.

5 NO LIMITS ON DEATHWALK. DeathWalk wasn't limited in PREY, so the character is able to die and resurrect as many times as needed. The lack of limitations on DeathWalk contributed to many of the other things that went wrong in development or that we would change if given another go around. Had we limited the number of times or how often the character could play in DeathWalk, it would have reduced or eliminated some of these other elements that went wrong.

One suggestion that developed in hindsight is to have set places where DeathWalk cannot occur, such as during boss fights.

KICKING THE LAST BUCKET

Controversy surrounded DeathWalk after PREY was released. Opinions on the feature were polarized, as some people liked the aspect of never having to die permanently, whereas others felt it reduced the impact of dying and made the game too easy.

Like other game features, DeathWalk could be improved upon, but ultimately I'm proud of how it turned out and the fact that we had the opportunity and ability to implement something as unique as this system. I think the true test of the effectiveness of DeathWalk is that it's a feature that I find myself sorely missing when I play other games (especially if I've forgotten to save in a while). Even if our system isn't copied directly, I hope that alternative methods of handling death become more commonplace in future games.

PAT CHRISTEN

SERIOUS EXPECTATIONS

Conflating science and game character creation

MORE THAN A DECADE AGO, HOPELAB board chair Pam Omidyar had an idea to combine science and entertainment technology to create a video game that would give young people with cancer a sense of power and control over their disease. The idea became a reality this year when HopeLab, the nonprofit Pam founded in 2001, launched RE-MISSION, a third-person shooter with 20 levels of gameplay featuring a gutsy, cancer-fighting heroine named Roxxi.

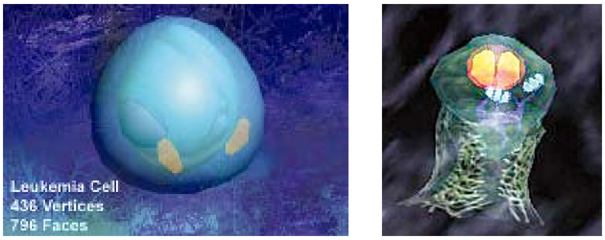
For most of us, RE-MISSION marked the first time we had worked with game developers, so we weren't familiar with the typical challenges they face, like designing levels, optimizing playability, and creating engaging characters.

The goal of RE-MISSION was clear from the start—to improve health-related outcomes for cancer patients. Now it was up to us make this goal fit into the game development process.

WHEN SCIENTISTS AND DEVELOPERS COLLIDE

Between the HopeLab team and our outside collaborators, we hold a great deal of expertise in the sciences. Still, we had great respect for the fact that game developers, not scientists, are the real experts when it comes to creating high-quality interactive games that young people actually want to play.

Predictably, tensions surfaced at certain points in the development process as we tried to incorporate key biologic principles into the game's design, not always realizing that these principles didn't necessarily make for interesting gameplay. A clear example was the challenge of developing one of the most important "enemies" in RE-MISSION: the cancer cells.



Between the concept stage (left) and final art work of the leukemia cells (right) in RE-MISSION, scientists, game designers, and players all provided input.

Under a microscope, cancer cells aren't particularly fearsome or intimidating. But our goal was to clearly illustrate to cancer patients the biology of cancer cells and how they behave in the human body. In RE-MISSION, as in real life, if you don't kill all the cancer cells you can find, they multiply and become a deadly threat. Accurately representing this threat in gameplay was incredibly important.

Concept sketches of the enemy cancer cells were reviewed by our team of collaborators—both the scientists and the game designers. The drawings that the scientists liked best were not at all what the developers considered to be viable prototypes for video game villains. Finding a balance between these two perspectives was critical.

SEEING EYE TO EYE

We decided to seek the advice of young people with cancer, our primary audience for RE-MISSION.

Throughout the development process, HopeLab consulted with young cancer patients to understand the challenges they faced physically and psychologically as they endured cancer treatments in order to accurately reflect their experiences and address their needs in RE-MISSION.

Asking these young people whether they thought the game that was taking shape was not only true to their life experience but actually cool and fun to play was equally important. So when conflict arose over how precisely we should represent cancer, our primary enemy in RE-MISSION, it made sense to go back to these young people for input.

Ultimately, this customer-focused approach is what enabled everyone involved with RE-MISSION to achieve our goals. It probably comes as no surprise that our young experts pushed for something a bit more fantastical than factual. But the result is an enemy that's not only based on accurate biological principles, but also is a real menace when it comes to gameplay. It's gratifying to have Roxxi blast the cancer cell villains, and, just like in real life, it's not always easy to get every last one.

MARK OF SUCCESS

The results we've achieved for a project as ambitious as RE-MISSION have been terrifically successful by almost any measure. In March, HopeLab announced preliminary results from a large-scale, randomized, controlled trial of RE-MISSION that indicate the game improves key health-related outcomes for cancer patients who play it.

Needless to say, we're thrilled with these results and greatly appreciate the expertise and collaborative efforts of the game developers who participated with us. RE-MISSION works, and the rational engineering and rigorous research behind it demonstrate that video games can do good in the world. ✖

PAT CHRISTEN *is president of HopeLab, which created the serious game RE-MISSION. The game is accessible free of charge to young people with cancer through the RE-MISSION web site: www.re-mission.net. To learn more about HopeLab and the making of RE-MISSION visit* ***www.hopelab.org****.*

March 5-6, 2007 in San Francisco

GDC MOBILE

GameDevelopers Conference Mobile 07

Connect with leading industry professionals at the definitive mobile game event of the year. At GDC Mobile 2007, you will garner strategic business contacts, access the latest market information and learn how to challenge current industry constraints by applying alternative business models. The power is yours to seize the mobile market. GDC MOBILE - *TAKE CONTROL*.

REGISTER TODAY!

www.gdcmobile.com

CMP
United Business Media

MICK WEST

THE INNER PRODUCT

STYLUS CONTROL

A look at the input schemes of alternative game devices

UNTIL RECENTLY, THE MAJORITY OF games have been controlled with either a handheld sticks-and-buttons controller or a combination of keyboard and mouse. Two factors are changing this. First, the casual game market's emphasis on simple and accessible gameplay has resulted in a large number of games that are mouse-only, and that only use single clicks of one mouse button. Second, the release of the Nintendo DS has hugely increased the potential audience for games that are controlled by a touch screen and stylus.

Both factors converge in Nintendo's Touch Generations branded games, which are essentially casual games for the DS that are played with a stylus. An additional factor is the increase in the installed base of tablet PCs and the related emerging market of ultra-mobile PCs (like the Microsoft Origami spec) that use touch screens with a stylus or a finger as their primary input device.

This article discusses a few of the programming and control design issues involved with implementing stylus control (and the related single-button mouse control) in a game.

DEFINE YOUR ROLE

What should the programmer's role be in implementing stylus player control? Are you implementing the player control or implementing tools that allow someone else to implement it? Programmers have always been a key part of implementing player control, and it's one of the few remaining areas where the programmer is directly involved in the most critical aspect of gameplay—the interface between the player and the game.

Yet, like most aspects of game development, even player control is shifting to a more data-driven approach, where a game designer is able to define the player control with some script language or table of data. Problems arise with this approach when the capabilities supplied by the programmer do not adequately match the needs of the designer, and it can be especially problematic when the programmer is tasked with implementing a specific set of input functionality and handing it over to the designer before moving on to other tasks.

The implementation of player control is an organic, exploratory task, especially when dealing with a controller (such as a stylus) that's new to the team. Unforeseen inadequacies will inevitably be found in any control scheme technical design, and subtle control bugs will crop up throughout the course of the project. Hence, it's highly recommended that a significant portion of the programmer's time is allocated to making refinements and fixes.

Dedicating programmer time to the task at hand is especially important if the coder is working on the actual player control, not just the underlying code. In that situation, the programmer needs to be free to make very rapid changes to the player control when the need arises.

The role of the programmer is unique in this area since the effective implementation of intuitive player control requires an understanding of what's going on at a per-frame level. Most designers are not typically experienced with such low-level functions, leaving them to rely heavily on the programmer to explain what's going on when "this just doesn't feel right."

Again, programmers are not simply implementing a control specification; they are an integral part of organically developing a seamless user experience.

MOUSE VS. STYLUS

At first glance, a stylus may seem to be just a mouse that draws on the screen, and indeed with a tablet PC, you can use the stylus pretty much as you would use a mouse. But if you're asked to develop a game that works well with both a mouse and stylus (or convert from one system to another), you need to think about what differences exist.

Other than other obvious physical distinctions, the fundamental logical difference is that a stylus has no need for a permanent cursor. A mouse is always moving a cursor object around the screen, but the stylus is its own cursor.

The second major difference, which is related to the first, is that you don't always know where the stylus is. With a mouse, if you move it from one position to another, say to click on one icon, then another, the code can detect the movement of the mouse between these two icons and use that information as a hint to the player control.

On platforms such as the Nintendo DS, the stylus becomes invisible when it's lifted off the screen; it essentially vanishes from one point to appear on another. On tablet PCs, the stylus can be detected moving in the air an inch or so above the surface, but it can still move out of range, and then re-appear somewhere else.

DEBUG BEFORE CODING

The single most important tool in implementing player control is the ability to visualize exactly what's going on. The very first thing you should implement is the display of the device input data in an easily understandable form. This need not be complex. For example, all the figures in this article use alternating red and black diamond shapes to represent every recorded stylus position, with a line drawn between them. This visualization will give you a good initial idea of the type of input you will be handling and can

MICK WEST *was a co-founder of Neversoft Entertainment. He's been in the game industry for 17 years and currently works as a technical consultant. Email him at* ***mwest@gdmag.com****.*

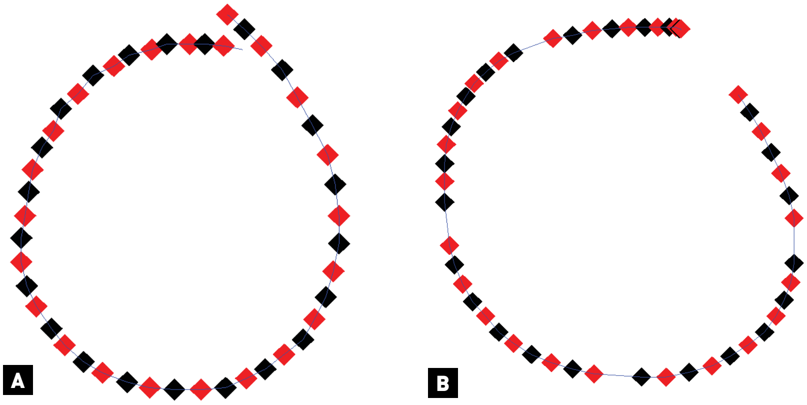


FIGURE 1 **These circular lines represent the slightly inaccurate path of a ball-based mouse (A) and the smoother, yet incomplete path of a laser optical mouse (B).**

highlight unexpected issues with either the hardware or the driver layer you're using to read the stylus or the mouse.

Figure 1 shows approximately the same stroke performed by two different mice; each read the same way by simply handling the WM_MOUSEMOVE messages. In Figure 1A, notice the points are fairly evenly dispersed, and the curve is reasonably smooth—but there are a few small kinks here and there. In Figure 1B there are two differences: the line itself is smoother, with fewer kinks, and more importantly, there are four samples "missing" from the data.

The smoothness of the line can be attributed to the second mouse (Figure 1B) being an expensive wireless laser optical mouse, whereas the first mouse (Figure 1A) is a cheap ball-based mouse that came with the computer. The gaps in the line could be anything, maybe a driver bug, or a problem in some higher layer, but the important thing here is that the simple visualization reveals these problems before any coding is done.

DEVELOP A LANGUAGE

For efficient communication between programmer and designer, you need to agree on a common language. The fundamental, low-level, building blocks of player controls are the device "events" you are probably already familiar with, specifically, the movement events and the contact or button events. But at a higher level, stylus control consists of a series of strokes.

A stroke is the path defined by the collection of points that the stylus moves through between a down event and an up event. A stroke can be as short as a single tap on the screen (equivalent to a mouse click), or as long as a stroke that covers the entire screen and indicates something like the path a weapon should take or a set of objects to be selected.

Other high-level control events are game specific. A "throw stroke" might indicate throwing something in a particular direction. Words such as "tap," "drag," "gesture," and "path" have different meanings depending on the game type, and it's important to establish exactly what these terms mean when discussing player control.

DIFFERENT STROKES

In my article "Pushing Buttons" (*Game Developer*, May 2005), I discussed the problem of "sloppy thumb," where players hold the controller in different ways causing different patterns of input, which the programmer needs to account for. Similar factors apply to stylus control and simple mouse control.

A stylus can be held at different angles, which affects how much it might slip when making contact with the screen. The force applied when tapping can also affect the shape of the resultant stroke. A light-handed person may give a nice smooth line, whereas a more heavy-handed person, or someone with poor motor control, may start the stroke inadvertently in the wrong direction as the style makes contact.

Figure 2 shows three different attempts at drawing the same simple left-to-right stroke. In Figure 2A, the player creates a nice clean stroke, holding the stylus firmly yet precisely, moving it smoothly and cleanly. In Figure 2B, the player has hit the screen hard with the stylus, but is holding it loosely, causing it to slip upward slightly at the start of the stroke. In Figure 2C, the start of the stroke is again indeterminate, as the player has tapped the stylus down hard and paused

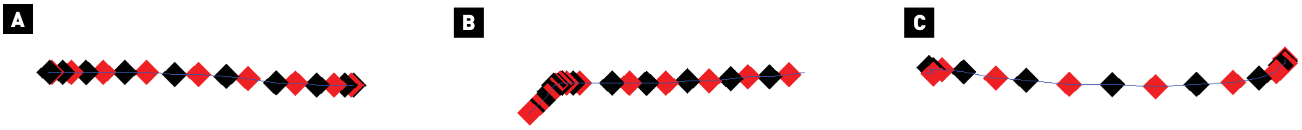


FIGURE 2 **The same simple left-to-right stroke can be interpreted differently depending on how the user handles the stylus: using a firm and steady hand (A); hitting the screen hard initially and holding the stylus loosely so it slips (B); or striking hard initially and pausing before completing the stroke (C).**

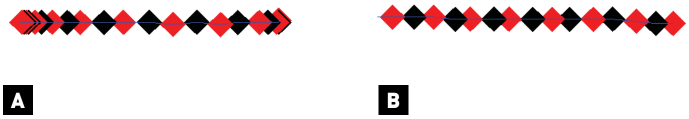


FIGURE 3 **Stylus acceleration varies with the user's intent. A stroke may begin and end slowly (A) or may have a continuous and steady speed (B).**

for a fraction of a second before starting the stroke. At the end of the stroke, the hand movement slows down, and the angle of the stroke drifts upward. This ending is more typical of left-handed players who hold their styluses with a firm overhand grip, as they would a pen.

What is the programmer to make of these strokes? It depends on what's going on in the game, but a common control element is "throwing" something, or shooting a missile in a particular direction. We need to translate the stroke into a direction vector. Two obvious approaches are to either use the vector from the first point in the stroke to the last, or to use the vector formed as the average from all the individual components of the stroke.

But as we can see from the strokes, the results of these calculations would give us a direction vector that is not in agreement with the intent of our sloppy players. Our precise player (Figure 2A) would be fine, but in both Figures 2B and 2C, the resultant vector would tend upward.

A possible solution would be to simply chop off the start and end of the stroke by a certain amount, ignoring, for example, the first and last 10 percent or maybe 0.05 seconds of a stroke. But a more sophisticated solution would be to try to identify the "straight" portion of the stroke, which we can easily recognize, but is a little more complex to program.

Whether you would actually want to implement this solution depends on the type of game you're making and its intended audience. Some games such as golf or bowling might depend on the nuance of a stroke for fine control of ball spin, and so the degree of slack you want to give the player would be less. But in ball tossing games such as MAGNETICA or LUXOR, all you want is a direction vector.

ACCELERATION INFO

The raw vectors that form a stroke tell you where on the surface the player moved the stylus and how fast. But by looking at the acceleration information in the stroke data, the programmer can gather information that indicates what the user was doing before and after making the actual stroke.

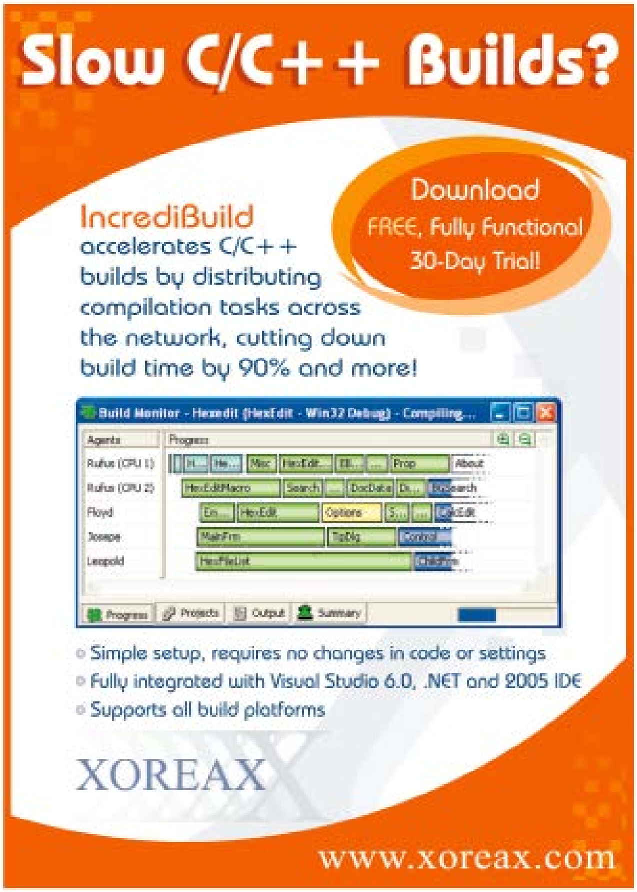
The two strokes in Figure 3 both cover about the same distance in the same direction. However, Figure 3A shows significant acceleration at the start of the stroke and deceleration at the end, which indicates that the player deliberately made the stroke from one point to another and that the stylus was not really moving before or after the stroke. In Figure 3B, the stroke was made at the same speed throughout, indicating the player was moving the stylus both before and after the stroke at the same speed. This is like the player moving the stylus through the air, dipping it down to briefly touch the surface and continuing.

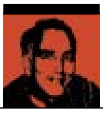
These two movements are very different, yet the interpretation of the strokes may or may not be different, depending on the type of game.

DEVICE-DRIVEN DESIGN

Games controlled by a stylus or mouse are increasingly common. Their technical knowledge of the device (and how input is received by the device) makes programmers integral to the design process and the organic implementation of the player controls.

Visualization is vital. Players have different input styles and mental expectations of stroke control, and by accommodating as many styles as possible without compromising coherent controls, you will expand the potential market and the conversion rate for the game.





NOAH FALSTEIN

GAME SHUI

CHANGING BRAINS

'I aimed at the public's heart, and by accident I hit it in the stomach.'

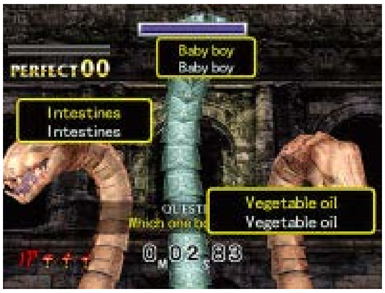
—Upton Sinclair, about his 1906 book *The Jungle*

ONE OF THE POIGNANT STORIES I remember from my high school U.S. history class was that of the writer and socialist activist Upton Sinclair whose book *The Jungle* was intended to awaken the world to the awful injustices and working conditions imposed on workers in the meat-packing industry. And he did awaken the world—but to the awful contents of the sausages they ate. Later that year, the Pure Food and Drug Act was passed, but workers continued to suffer.

What does this have to do with designing games? Let's consider another quote. Stop me if you've heard this one: "How many therapists does it take to change a light bulb? Just one, but the light bulb has to want to change."

That joke gets at the heart of a fact of human nature: getting someone to change is really tough, unless they really want to change. Most people would rather just have fun, perhaps by playing a game. And yet, as regular readers of this column know, games are ultimately about learning, and learning is the process of changing one's knowledge, and often consequently one's actions.

Perhaps the proper joke for our industry is: "How many games does it take to teach a player? Just one, but the player has to want to learn."



Sega's Typing of the Dead is an example of a game that teaches players without removing them from the action.

SURVIVING ON BROCCOLI

For years, most educators have looked at games as the enemy, something their students choose to do instead of serious study. In the 1980s a few tried to harness the power of games, reasoning that if students were motivated to play, there might be a way to get them to learn in the process.

Many of the early edutainment titles were, in the terms of developer Brenda Laurel, "chocolate-covered broccoli," merely the same old boring drills with the trappings of gameplay as a superficial and ultimately unsatisfying "topping." It's really only in recent years (with what has become known as the serious games movement) that we're starting to see the true integration of good game design techniques with good pedagogy.

SURVIVING ON SKILLS

Still, there's the tendency to view gameplay as a kind of fairy dust that can be sprinkled over boring academic content. But if you look at the most popular games, invariably they are focused firmly on themes of survival, and the skills one learns while playing are intimately connected to survival and social skills (for more on this topic, see "Natural Funativity," www.gamasutra.com/features/20041110/falstein_01.shtml).

Frequently, I've found that when an educator turns to games as a way to make boring topics more fun, those topics are far removed from everyday survival.

For straightforward topics, just putting the material into an arcade context can sometimes help. Typing of the Dead is a typing trainer, and although there's no logical reason why being a faster typist should blow up zombies, it can really work as a motivator.

However, if you have a more complex subject that is inherently far from a student's everyday survival needs—for instance, solving differential equations, or understanding the causes of the Great Depression—representing the material in a game becomes much tougher. Conversely, when you want to teach something that is closely related to survival, you can do very well—hence the success of America's Army and Full Spectrum Warrior.

MAKING CONTENT PLAYABLE

In the end, it may be that games are the wrong way to go. Games are powerful teaching machines, but not the ultimate solution to every problem.

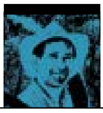
Alternative approaches are out there, though. One is to simplify. If a complex math concept can be broken down into very simple steps, which can in turn be linked to an abstract game, that game might not be able to teach the full subject of solving differential equations, but it could get you part of the way there and shorten (or reinforce) the classroom instruction.

Another solution is to find a way to link at least part of your topic to survival. The political causes of the Great Depression may be abstract, but the way it affected individuals and families was quite real, and could transition easily into gameplay representation.

One of the few big successes from the early edutainment days was Oregon Trail, which taught about the American West through a theme that was heavily survival oriented. I recently used that approach myself in the design for a Lauer Learning game called Freedom Fighter 56, which teaches about the Hungarian revolution 50 years ago by putting the player into the action.

Ultimately, if we designers are supposed to motivate players to want to change, we need to reach them with topics and themes that they care about. Otherwise, we are likely to aim for their brains, and hit them in their hand-eye coordination. And everyone knows if you want to stop a zombie, you *must* hit its brain.

NOAH FALSTEIN *has been a professional game developer since 1980. 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**.



STEVE THEODORE

>> PIXEL PUSHER

THROUGH A SCANNER DARKLY

LAST MONTH, PIXEL PUSHER WAS chirruping happily about what a great time it is to be a video game artist. This month, we return to the moody artist stereotype to dredge up some doom and gloom.

Anyone who has been to a developers' beer night or sat through a state-of-the-industry conference panel knows the litany of scary, potentially career-changing technologies and trends perpetually on the horizon. There's outsourcing, next-gen team bloat, the Autodesk-Alias merger, any number of ongoing workers' rights lawsuit, and the decline of PC gaming—not to mention bird flu, the loss of Pluto as a planet, and the return of hair metal bands.

If you're not depressed already, here's another challenging technology that hasn't made it onto most artists' early warning radars just yet—scanners.

3D scanners have been around for a couple of decades in various forms, so they're not exactly "new," but in the last couple of years falling prices and improving quality have made them increasingly attractive. Don't be surprised if you see a lot of breathless articles in the press over the next year about the new generation of scanners, how they were used for this movie or that game, and how they produce beautiful assets on command.

This party line sounds a lot like the one we heard in the early days of motion capture, when the media predicted that animators would be laid off in droves while an army of drama students in spandex grabbed all the jobs. The reality is, of course, not quite so simple or baleful—but even if scanning won't put us all on the breadline, it's still worth our time to look at how the technology behind it is evolving, what it's good for, and what it may mean for the industry.

HUMBLE BEGINNINGS

Older scanners were impressive at first glance, but the results didn't always stand up to close examination. Those complimentary "let us scan your art director" models that came back from every Siggraph junket often featured poor handling of undercuts, noisy data, and textures with built-in lighting and projection smears. Mechanical models sprouted digital acne worthy of Joey Ramone, with dimpled planes, rough edges, and crude sculpting. Worst of all, the raw scan data was monstrously heavy.

Between bloated data, laborious cleanup, and mediocre results, scanning didn't put many modelers on the dole. The technology did find useful roles in a handful of niche jobs, such as cranking out recognizable likenesses of sports figures or movie stars.

Cinema teams less bound by the limits of real-time performance were also able to experiment with scanned characters, and if nothing else, scanners were unbeatable for dealing with wrinkles and folds in clothing. Yet on the whole, scanning was definitely an exotic way to generate game content, the sort of technology that seems to exist primarily for trade show demos.

THAT WAS THEN

Even the early scanners were impressive technical achievements. Attempting to translate a real world object into a 3D model is a daunting task (pat yourselves on the back, poly pushers). For us, the key is to find significant features, like the corners of a box, the roundness of a ball, or the features of a face, which become the basis of the model layout. It's intuitive to us, but it's way beyond the capacities of contemporary computers.

Scanners, on the other hand, take a brute force approach. They collect millions of perfectly accurate bits of data, but they have no idea which of those bits are important and which are filler, using sheer volume of data to capture the nuances. In a hand-built model, every contour is there because the artist knows it should be. In a scanned model, by contrast, the key visual features don't really exist, their presence merely suggested by fortuitously placed vertices (see Figure 1, pg. 44).

In this way hand-built models are like vector drawings, whereas scans are more like bitmaps. Hand-built models elegantly describe objects with strong forms, such as sculptures or machines. Scans, like bitmaps, do a great job on noisier and more organic shapes (for example, human faces), but they are also memory hogs. A scan requires far more data to get visual results that approximate the quality of a good model. In the last generation of games, turning a scanned model into a real-time-ready model was almost as hard as building one from scratch, so scanners rarely emerged from their specialist roles.

THIS IS NOW

Nothing lasts for long in the game business. Three important factors have conspired to make scanning a mainstream option today.

The first big change is on our end: while the basic technology behind scans hasn't changed, our output technology has. Having cheap and plentiful normal maps means we can finally derive some tangible benefit from all those zillions of

STEVE THEODORE has been in the game industry for 12 years. He's worked at FASA Interactive, Valve, RAD Game Tools, and until recently was the technical art manager at Zipper Interactive. He's now a founding partner at Giant Bite. Email him at stheodore@gdmag.com.

scanned polys. Per-pixel lighting preserves the important features of the high-resolution scan without busting our geometry budgets. The latest version of Direct3D and a handful of games have even toyed with real-time displacement maps, so we may be able to adopt the Hollywood technique of storing scan data as displacement maps on top of Nurbs curves or subdivision geometry.

Advances in scanning software have also helped push scanners back into the limelight. Early scanning applications were mostly concerned with the messy business of converting huge clouds of point data into usable data, or matching up multiple scans into coherent models. Nowadays, scan software (such as GSI Studio, CySlice, or InSpeck) also gives artists a lot of freedom to draw in animation-friendly topology, produce useable UV maps, and quickly generate normal maps off the original data (see Figure 2).

All these benefits add up to a real reduction in turnaround time. A high quality scanned asset can be almost ready for prime time in a day or two.

Technical evolution has also brought scanners back to center stage. The latest generation of scanners are accurate to a truly scary degree. There are scanners on the market today that boast a spatial resolution of about 0.003 inches (or 0.075mm), literally the thickness of a human hair.

The increased resolution is less important for spatial accuracy than for properly capturing surface texture. A modern scanner can capture the stitching in a pair of jeans, the leather texture of a football, or the pores on a face. This microscopic resolution, combined with even larger numbers of sample points, now lets scanners tackle planar surfaces and sculpted contours with much greater success. Instead of the pockmarks and pimples, the newest scanners produce very passable renditions of sculpted surfaces. Modern scanners can even capture the grip texture on a pistol or the lettering embossed on a button.

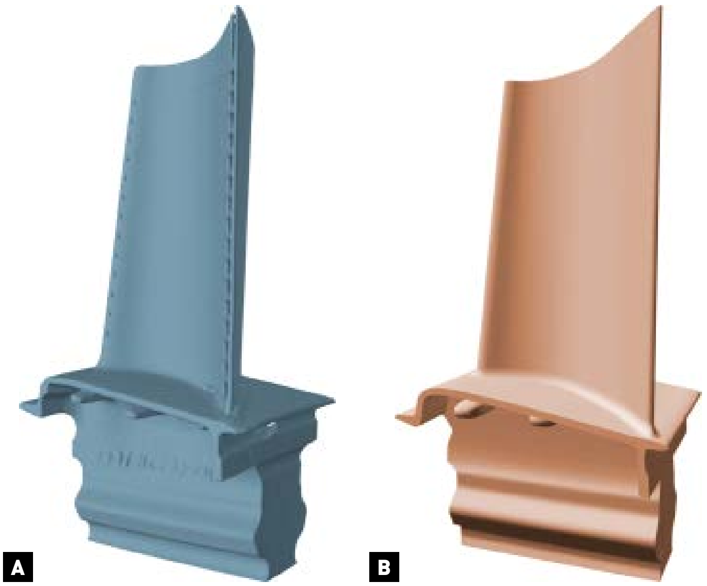


FIGURE 1 **This model of a turbine blade (A) has almost 900,000 vertices, but it still can't represent the smooth curvature of the blade perfectly, illustrating the inefficiency of raw 3D scans. The hand-built model (B) has fewer than 2,500 vertices, but it shades more cleanly.**

FIGURE 1A COURTESY OF THE GEORGIA TECH LARGE MODELS ARCHIVE.

FIRE!

With all these new and almost magical capabilities, it's easy to get spooked and start thinking the real output of these new machines is going to be pink slips. At this very moment, some vice president of something-or-other is looking at a brochure for a $300,000 scanner, wondering how many vacation-taking, benefit-using, class-action-lawsuit-bringing modelers it might replace.

Keep dreaming. There are several reasons massive layoffs aren't right around the corner. Remember that we've been through this cycle before, and the skeptics have always been right. Survivors of the first wave of commercial mo-cap production can certainly bear witness to how people talked about the end of hand animation; some of us even remember when people argued seriously that the availability of 300dpi scanners would end the art of computer painting almost before it began.

We're not quite ready to bow before our robot overlords just yet. There are four key areas where scanners need a lot of help from traditional techniques.

Shaders. The biggest reason scanners aren't an end-to-end solution is material handling. There's no commercially available way to capture the surface properties of an object. Scanners do capture texture maps, but typically they do so with plain old digital photography, so getting those scans into proper textures with neutral lighting and no shading or false highlights takes a lot of attention. Scanned textures are a great starting point for hand texturing, since they provide perfect registration with the geometry, but the process is nothing like "scan it and can it."

On top of all that, scanners don't capture all the other surface information that modern shader systems require. Specularity or incandescence, for example, or translucent or transparent surfaces are obstacles to scanning. Shiny or transparent areas often need to be painted over to achieve good quality scans.

Professional doomsayers may point out the body of academic work on bidirectional radiance diffusion function (BRDF) probes, which can capture a very

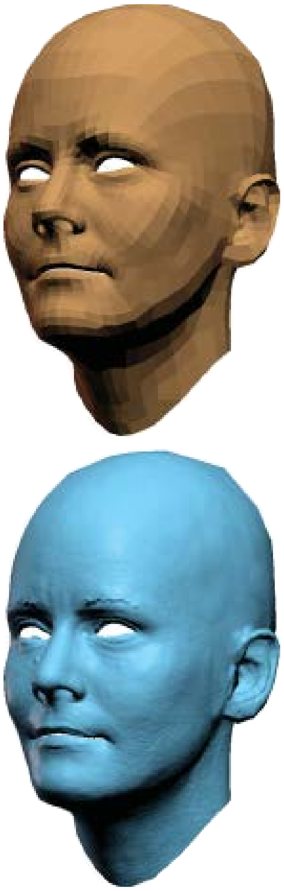


FIGURE 2 **With normal maps, we can finally shoehorn all that scanned data into real-time models.**

full representation of how a surface interacts with light. The technique was used to great effect in the last *Matrix* movie, but there's still no off-the-shelf solution for those of us without a PhD.

Flexibility. Much like motion capture or asset outsourcing, 3D scanning cuts the load of asset creators, but it also creates a lot of work for producers and art managers. Unless you're planning to shell out $300,000 for an in-house scanner, you'll probably be sending work to a service bureau, possibly in another city or state.

A scan-heavy workflow also means managing relationships with prop houses, costume agencies, model fabricators, and possibly modeling agencies or actors' unions. Riding herd on a bunch of in-house modelers may come to seem pleasantly simple by comparison.

Style. Of all the potential problems, style is the hardest to quantify but potentially the most important.

The exactitude of a good scan is awe-inspiring. Unfortunately, the parts of your game that can't be scanned could look pretty pokey by comparison. Sports games, racing games, and perhaps contemporary military games can make very extensive use of scans without busting their illusions, but other genres are more problematic.

Some creature-makers are scanning in Hollywood-style maquettes. This produces very satisfying results ("Leon" from Matthias Worch's 2005 Game Developers Conference talk is a great example), and hefty clay statuettes are great sales tools when dealing with skeptical suits.

Real sculpture, however, can be even more of an expensive craft than 3D modeling, so this route is for the artistically ambitious rather than the budget conscious.

Uncanniness. Last but far from least is the yawning pitfall of next-generation games, the infamous Uncanny Valley where characters become so superficially real that every technical or artistic flaw becomes weirdly repulsive (see "Uncanny Valley," December 2004).

The precision of modern scanners can actually make it harder to capture people well. Since no actor can sit perfectly still, upright scans suffer from drifts and misalignments. Scanning an actor on a bed or recliner solves the motion problem, but then the pull of gravity distorts the face into a disturbing zombified mask. Scans will be perfectly "accurate" and nearly "photorealistic," but without careful art direction they won't always be appealing.

THE PENULTIMATE TRUTH

Like motion capture and digital photography before it, 3D scanning is not going to bring the world crashing down around our feet. Falling prices and continuously improving quality mean that you will definitely be seeing more scans in the future. It's quite likely that some people who make models today will find themselves touching up scans in the future, just as some people who used to set keys now work from mo-cap data.

If the prospect of being second fiddle to a laser appalls you, start thinking now about what you can do that scanners can't. Do you have a knack for design or a strong personal style? Are you a pipeline wizard who can find new ways to squeeze quality out of a game engine? Can you mimic any style with aplomb? The niche you pick matters less than giving serious thought what you're good at and what you want.

That, at least, is something the machines can't do ... yet. ⁙

RESOURCES

For a list of all things scanning related, including a long list of scanner manufacturers' web sites, see:
www.simple3d.com

The Stanford 3D Scanning Repository:
http://graphics.stanford.edu/data/3Dscanrep

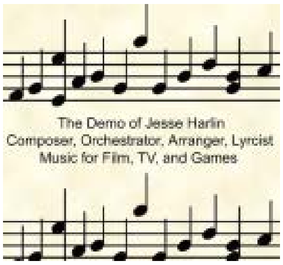
The Georgia Tech Large Models Archive:
www-static.cc.gatech.edu/projects/large_models

Scanning service bureau resources:
www.gentlegiantstudios.com
http://xyzrgb.com
www.eyetronics.com
www.cyberware.com/info/scanningCenters.html

SCANNED HEAD COURTESY OF CYBERWARE, INC.

BE A DEMO DYNAMO

THE VIDEO GAME INDUSTRY IS NOMADIC by nature and no field is more transient than audio. As a result, all audio professionals are forced throughout their careers to prove and reprove their worth solely on the strength of their resume and the first 30 seconds of their demo.


The Demo of Jesse Harlin
Composer, Orchestrator, Arranger, Lyrcist
Music for Film, TV, and Games

A bad demo disc cover, typos and all, will become lost among all the other poorly designed discs.

Freelance audio folks rarely get to see the demos of their competition and audio directors don't have time to give critiques on the demos that end up in their "no" pile. So how do you know if you're a "no" or not? More importantly, how can you ensure you're in the "yes" pile?

PRESENTING YOUR BEST

No demo exists in a vacuum. To the contrary, almost every demo will find itself sitting in the middle of a stack of demos. Before any tracks can be heard, the first impression your demo will make is a visual one. As such, presentation is everything. Imagine an audio director staring down a stack of CDs without any idea of what the audio on them sounds like. At that moment, the selection of one demo over another is almost completely arbitrary. The only advantage your demo has is its ability to stand out from the crowd.

FASTESTMAN
INTHEWORLD

A good demo disc cover, from composer Duncan Watt, uses bright artwork and a catchy design to make it stand out.

This means that CD- and DVD-Rs that have been written on with a Sharpie marker are automatically at a disadvantage. At their best, they look sloppy. Far too frequently, however, they contain illegible or incomplete text. No matter how attractive you may think your handwriting is, it can't compare to a printed CD label. But what do you print on the label?

First and foremost, always include your name and contact information on every part of your demo—whether it's the case, the disc, the resume, or a business card you tossed in for good measure. Discs and cases are easily separated and if there isn't any clue as to whose audio the disc contains, that nameless disc just lost itself a gig.

Second, if you're looking to stand out, avoid the standard cliché images common to the industry. For composers, this means never cover your demo with pictures of sheet music, treble clefs, noteheads, violins, etc. So many composers use these images as the easy way of saying, "This is a music disc." Hundreds of composers are all using the same imagery.

For sound designers, stay clear of pictures of waveforms, speakers, or screenshots from Protools as they're all overdone. Both disciplines are also guilty of another common cliché. If you include a photo of yourself, don't take it while sitting in front of your gear. Take a picture of yourself anywhere else. You're looking to stand out and be different. Trust me. Audio directors will still believe you know a mod wheel from a pan pot if they don't see you beside one.

If you don't have the graphic design or Photoshop skills to design something other than the clichés for yourself, hire a graphic designer to do it for you. It's that important. You're creating a brand of yourself that you're then marketing throughout the industry. You owe it to yourself to do everything you can to ensure that your disc stands out from the pile of unknowns.

FOCUS YOUR EFFORTS

Once your demo has been yanked from the pile, your audio chops are finally on display. There are a number of different schools of thought regarding how best to present your work, and unfortunately no clear answers with which everyone agrees. Should you make a single montage piece or a series of smaller, separate tracks? As someone who reviews demos, I find that I prefer to see separate tracks since I can easily skip back and forth through the demo to hear tracks again. Should your demo vary widely stylistically or should it be most representative of your strongest talents? That depends on the specifics of the job for which you're applying.

Without a doubt, the most important aspect that is often overlooked with demos is the ability to focus it specifically to the job for which you're applying. An all-purpose demo is great for GDC, but if you're applying for a specific job you should submit a demo that speaks directly to that position. A sound design job does not warrant a music demo. A company that only makes racing games does not want to hear a sweeping fantasy soundscape. Your demo should be flexible enough that you can select from a pool of available tracks that are appropriate at different times for different job opportunities.

BACK TO BASICS

Never forget the basics. Make sure each disc works on both Mac and PC platforms, as you never know what the listening environment on the other end will be. Listen through your demo to make sure all of the tracks play through to the end of the file. Make sure your demo is short—no longer than 10 minutes, preferably closer to five. Unless specifically requested, don't send data discs with anything other than audio and video clips. Word documents and Excel spreadsheets are not the stuff of demos.

There's a certain element of alchemy to crafting the right demo. Listen when people give you feedback, update your demo frequently, but mostly don't feel too stressed by demos. Our industry is full of dream jobs and you'll have plenty of opportunities throughout your career for fine-tuning. ⁙

JESSE HARLIN *has been composing music for games since 1999. He is currently the staff composer for LucasArts.You can email him at* *jharlin@gdmag.com*.

NeurOK LLC seeks VP of Strategic Marketing to develop & establish company's North American marketing strategy, negotiate agreement with U.S. partners for Neurok products, including overseeing marketing and sale of company's products in North American markets; develop business plan in cooperation with sales, business development, & engineering functions; oversee manufacturing process, collaborate with manufacturing partners on technical specifications, set production timelines & assure quality control & compliance with federal regulatory standards; work with company partners & customers to explain technological features of products; oversee the company's U.S. sales & marketing budget; assist outside counsel with preparation of U.S. patent, trademark & copyright applications; work regularly with Research & Development team (located in Russia). Min. req'ts: Master's degree in Bus. Admin., Finance or Mktg + 5 years exp. in information technology.
Candidates must be fluent in Russian.

Send resume, salary requirements & cover letter to:
Thomas Streigler, Job Code: VPSM,
Neurok LLC,
4370 La Jolla Village Drive, Ste 400,
San Diego, CA 92122.

EOE.

ADVERTISER INDEX

Game Developer (ISSN 1073-922X) is published monthly by CMP Media LLC, 600 Harrison St., 6th Fl., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as CMP Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 1601 W. 23rd St. Ste. 200, Lawrence, KS 66046-2703. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 841-2624. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence. All content, copyright *Game Developer* magazine/CMP Media, LLC, unless otherwise indicated. Don't steal any of it.

Cryptic Studios, Inc. is an independent
100% employee-owned video game developer.

Right now we're developing Marvel® Universe Online,
City of Heroes® expansions, and multiple top secret titles!
Join our team and work on some of the biggest,
most exciting, and most successful games online.

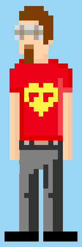
We're hiring unique and talented individuals
from inside and outside the video game industry.
If you are interested in a great environment go to:

www.CrypticStudios.com

CRYPTIC™

ARTISTS • DESIGNERS • PROGRAMMERS
LET'S RIDE!
ALL EXPERIENCE LEVELS NEEDED
SECRET LEVEL™
{ WWW.SECRETLEVEL.COM/JOBS }
SEGA®

Image by Eva Kolenko, BFA, Photography



create
YOUR WORLD

UNITED STATES POSTAL SERVICE

Statement of Ownership, Management, and Circulation

1. Publication Title: Game Developer. 2. Publication No.: 1073-922X. 3. Filing Date: September 29, 2006. 4. Issue Frequency: Monthly 5. Number of Issues Published Annually: 12. 6. Annual Subscription Price: $49.95. 7. Complete Mailing Address of Known Office of Publication (Not Printer): CMP Media LLC, 600 Harrison St., San Francisco, CA 94107-1370. Contact Person: Ken Savoy. Telephone: 516-562-7297. 8. Complete Mailing Address of Headquarters or General Business Office of Publisher (Not Printer): CMP Media LLC, 600 Harrison St., San Francisco, CA 94107-1370. 9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor: Publisher: Philip Chapnick, CMP Media LLC, 600 Harrison St., San Francisco, CA 94107-1370; Editor: Simon Carless, CMP Media LLC, 600 Harrison St., San Francisco, CA 94107-1370; Managing Editor: Jill Duffy, CMP Media LLC, 600 Harrison St., San Francisco, CA 94107-1370. 10. Owner: CMP Media LLC, 600 Community Drive, Manhasset, NY 11030-3875, an indirect, wholly owned subsidiary of United Business Media plc, Ludgate House, 245 Blackfriars Rd., London, SE1 9UY, U.K. 11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities: None. 12. Tax Status: Has Not Changed During Preceding 12 Months. 13. Publication Title: Game Developer. 14. Issue Date for Circulation Data Below: October 2006.

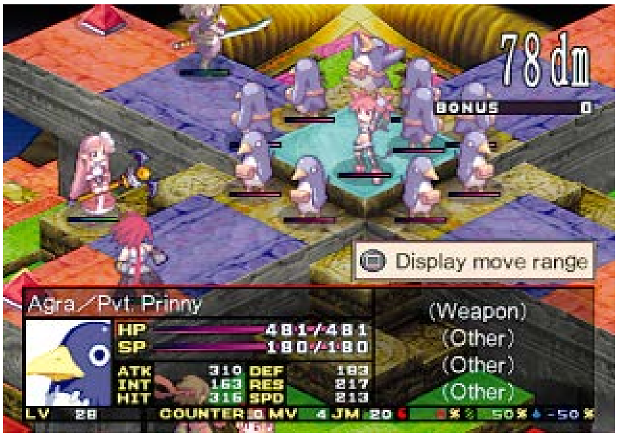
15. Extent and Nature of Circulation:	Average No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total No. Copies (Net Press Run)	42,770	42,121
b. Paid and/or Requested Circulation		
(1) Paid/Requested Outside-County Mail Subscriptions Stated on Form 3541. (Include advertiser's proof and exchange copies)	31,829	31,221
(2) Paid Inside-County Subscriptions Stated on Form 3541. (Include advertiser's proof and exchange copies)	0	0
(3) Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Non-USPS Paid Distribution	2,731	3,202
(4) Other Classes Mailed Through the USPS	0	0
c. Total Paid and/or Requested Circulation [Sum of 15b. (1),(2), (3), and (4)]:	34,560	34,423
d. Free Distribution by Mail (Samples, Complimentary, and Other Free):	0	0
(1) Outside County as Stated on Form 3541:	0	0
(2): In-County as Stated on Form 3541:	0	0
(3) Other Classes Mailed Through the USPS:	0	0
e. Free Distribution Outside the Mail (Carriers or Other Means)	2,217	1,227
f. Total Free Distribution (Sum of 15d and 15e)	2,217	1,227
g. Total Distribution (Sum of 15c and 15f)	36,777	35,650
h. Copies Not Distributed	5,993	6,471
i. Total (Sum of 15g and 15h)	42,770	42,121
Percent Paid and/or Requested Circulation (15c Divided by 15g Times100)	94.0%	96.6%

16. Publication of Statement of Ownership: This Statement of Ownership will be printed in the November 2006 issue of this publication. 17. Signature and Title of Editor, Publisher, Business Manager, or Owner (signed): Simon Carless, Date: September 29, 2006.

A THOUSAND WORDS

DISGAEA 2: CURSED MEMORIES

DISGAEA 2 is the PlayStation 2 sequel to the anime-style strategy RPG hit from Nippon Ichi Software, with character designs by Takehito Harada. The character depicted here is Laharl, one of the main characters, with a Prinny hanging from above.



78 dm

BONUS 0

Display move range

Agra/Pvt. Prinny

HP 481/481

SP 180/180

ATK 310 DEF 183

INT 163 RES 217

HIT 316 SPD 213

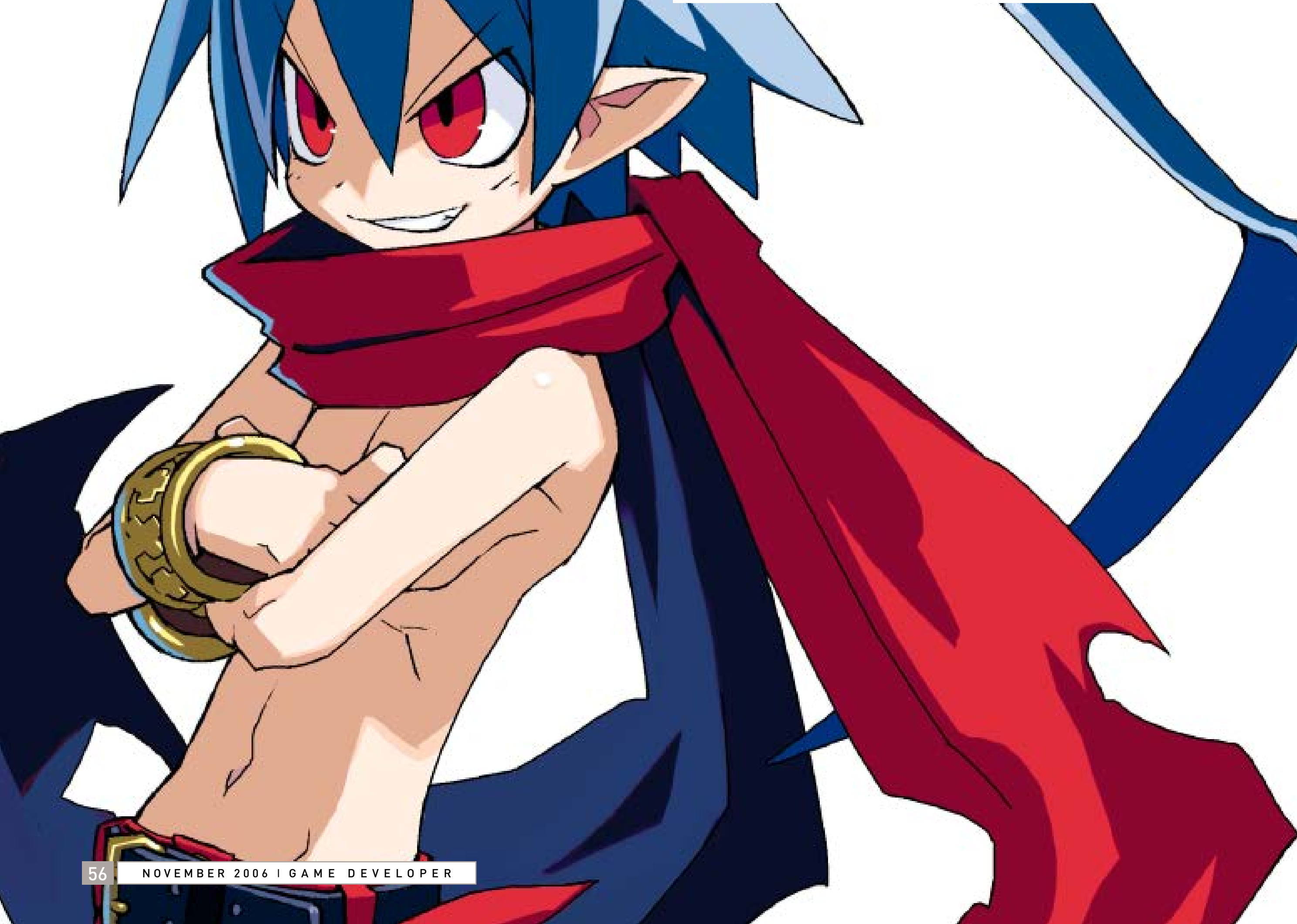
LV 28 COUNTER 0 MV 4 JM 20 % 50% -50%

(Weapon)

(Other)

(Other)

(Other)



Are Your Characters Developing a 'Tude?

- Author Behaviors Interactively
- Plug in custom procedural nodes
- Leverage the full power of Havok Physics and Animation

Havok Behavior Gives You Complete Character Control

havok®
www.havok.com
Stay Ahead of the Game!™

Miles Solid Gold Hits

finest quality
Middleware
100% Royalty Free
amazing lifelike STEREO
in fabulous
HIGH FIDELITY!
MILES
Sound System
New Version 7 now available - discrete multichannel output (5.1 and more), chainable DSP filters, seamless MP3 looping, integrated 2D and 3D audio, and much more!
New platforms! Now for Wii, PC, Mac OS, Sony PS2, Sony PS3, Xbox, Xbox 360, and Linux.
New
Wii support!
MILES SOUND SYSTEM
Greatest Hits in Game Development Technology
RAD GAME TOOLS
MILES

...cclaimed

... the most sophisticated sound system available. It includes integrated 2D and 3D digital audio, digital audio streaming, audio compression (MP3 with patent rights, Ogg and ADPCM), chain-able DSP audio filters, deep MIDI support with an integrated DLS software synthesizer, and tons of other sound related features (red book audio, timers, memory, file I/O, conversions, etc).

Top of the Charts

The Miles API has been used and refined for almost 15 years. It is robust, consistent, and easy to use. Over 3,200 games have now used Miles!

Digital Hit Parade

Miles has unmatched digital audio features. We support discrete multichannel output for the latest 5.1 and up speaker systems, along with EAX 2, 3, and 4. We have a dozen DSP filters that can be chained for fancier effects. Miles also includes internet voice chat codecs and an amazingly fast MP3 decoder (with patent rights), optimized for x86, SSE, 3DNow!, PPC and MIPS. Our software mixer uses massively optimized assembly kernels for x86, PPC and MIPS.

MIDI Music Madness

Miles includes a robust MIDI sequencer with a full-featured DLS synthesizer. It supports everything you need to add interactive music to your game.

Platform Platinum

Miles is available for Nintendo Wii, all Windows (including Win64), all Mac OS, Sony PS2, PLAYSTATION(R) 3, Xbox, Xbox 360, GNU/Linux and even DOS.

For More Information
425.893.4300
www.radgametools.com

RAD GAME TOOLS ©

THE BEST IN GAME DEVELOPMENT TECHNOLOGY

BINK VIDEO — Bink Video Technology

GRANNY 3D — Granny 3D Animation System

PIXOMATIC — Pixomatic Software Renderer

All trademarks mentioned herein are property of their respective owners.