game developer

THE LEADING GAME INDUSTRY MAGAZINE

INSIDE: PATTERN RECOGNITION IN AUGMENTED REALITY

THE LEADING GAME INDUSTRY MAGAZINE VOL17 NO11 DECEMBER 2010

UBM

Scaleform®

THE #1 VIDEO GAME UI SOLUTION

NOW SUPPORTING

FLASH 10.1

AS3

Also featuring: Multi-threaded rendering, AMP Profiler tool, 3Di, Stereoscopic 3D, 2-5x Performance improvement

EVALUATE TODAY: WWW.SCALEFORM.COM

gamedeveloper

CONTENTS.1110

VOLUME 17 NUMBER 11

POSTMORTEM

26 **UBISOFT MONTREAL'S SPLINTER CELL CONVICTION**
Moving SPLINTER CELL's trademark stealth-based game play in an action oriented direction without undermining the series was a tricky problem for Ubisoft Montreal. The original game design for CONVICTION was radically different from the franchise's traditional mechanics and necessitated a mid-development reboot. Fortunately, clear direction and a well-tuned production pipeline kept the team from being knocked off balance by the shift.
By Patrick Redding, Alex Parizeau, and Maxime Beland

FEATURES

9 **FRAGGED**
Big explosions with lots of fragmentation and debris are a video game mainstay. But making them look convincingly unique is a tedious process for artists. Here, the authors describe a procedural technique for automatically generating fragmented meshes that can save both time and sanity.
By Robert Perry and Peter Wilkins

15 **FULL REACTIVE EYES ENTERTAINMENT**
Are the now-ubiquitous Quick Time Events in games a lazy way to keep players mashing buttons? Or, are they broadening the range of expression for game designers? Using examples from the past and present, this article looks to the future of QTEs.
By Tim Rogers

22 **ELECTRIC EYE**
Kinect and Move have arrived, and with them comes a growing interest in augmented reality. In this article, César Botana guides you step by step through the basics of the OpenCV library, an essential tool for pattern recognition in augmented reality games. Also covered is placement of 3D objects in game worlds based on these patterns' orientations.
By César Botana

DEPARTMENTS

2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]
Move Back to Kinect

4 **HEADS UP DISPLAY** [NEWS]
IGDA Leadership Forum, the Automatypewriter, and Bunten papers archived.

6 **2010 FRONT LINE AWARD FINALISTS** [NEWS]
The Award Finalists for Art, Audio, Game Engine, Middleware, Networking, and Production/Programming tools.

33 **TOOL BOX** *By Tom Carroll* [REVIEW]
Autodesk 3ds Max 2011

36 **PIXEL PUSHER** *By Steve Theodore* [ART]
The Balkans

38 **THE INNER PRODUCT** *By Giacomino Veltri* [PROGRAMMING]
Game Configuration at Crystal Lake

41 **AURAL FIXATION** *By Jesse Harlin* [SOUND]
The Weight of Silence

42 **DESIGN OF THE TIMES** *By Soren Johnson* [DESIGN]
Stop Making Sense

44 **GOOD JOB!** [CAREER]
Bradley Johnson Q&A, who went where, and new studios.

45 **EYE ON GDC** [GDC]
Independent Games Festival news and IGF China finalists.

46 **EDUCATED PLAY** *By Jeffrey Fleming* [EDUCATION]
David Arenou's IMMERSIVE RAIL SHOOTER.

48 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]
Our Last, Best Hope

MOVE BACK TO KINECT

ARE THE NEW MOTION SYSTEMS INFLUENCING THEIR DEMOGRAPHIC BY PROXIMITY?

NOW THAT BOTH SONY'S MOVE AND MICROSOFT'S Kinect have hit the market, we can take full stock of the new kids on the motion control block. Where Move goes for a "Wii plus camera plus greater precision" setup, Kinect hopes to make your body the controller.

It's no secret to anyone that these technologies are spurred on by the success of Nintendo's Wii. If the drive to create the technology wasn't necessarily Nintendo-inspired, the desire to release it with such pomp and circumstance certainly was.

Both companies have tried to get a drink of Nintendo's milkshake in slightly different ways. Each platform lends itself to a different experience, but both companies have unfortunately gone straight for the Nintendo-alikes.

KING OF THE ME TOOS

>> Many developers have played with either or both systems by now, but for those who haven't, I'll explain a bit. The Move uses a camera to identify your movements (in a 2D sense), and also to put you "in the game" at times, like the EyeToy before it. It uses wands to interact with objects and avatars on the screen, using both motion control and buttons simultaneously.

With the Kinect, there are no buttons, just body (especially hand) recognition through the 3D camera. Menus are navigated and confirmed through swipes, and holding your hand over an on-screen button for a certain amount of time. In-game actions are all performed with your body, and an avatar usually does its best to mimic your movements.

These setups each lend themselves more naturally to certain kinds of activities, and less well to others. Let's look at two genres—pets, and sports.

In the pets genre, Move has EYEPET, and Kinect has KINECTIMALS. In both, you're supposed to be interacting directly with a little pet character. For the Move, it's a bit odd, because not only are you holding this wand in order to interact with the pet most of the time, your legs are physically in the picture, brought in-game by the camera. This creates an illusion that you could reach down and grab the pet, but of course you're always physically behind it in the screen, no matter how close you get to the camera. The interface doesn't feel natural, and you don't feel as connected to your pet, even though it's visually "in" your space.

With KINECTIMALS, there's a clear distance between you and the screen—you're in your living room, and the animal lives inside the TV. But it actually works much better, because all your interactions with the animal are done via virtual hand avatars that mimic your actual hand movements. So it's much easier to feel like you're really interacting with this little beastie—and the minigames (for the most part) also take this interface into account. In this camp, the Kinect wins.

Then there's sports. Here, the Move makes great sense. For many sports you have a bat, or a racket, or a bow, so holding a physical object—the wand—makes you feel connected to the world. With Kinect, steering a car or holding a bat feels bizarre, because you don't have an actual "prop." Whereas with KINECTIMALS, the feedback was very positive, here it's much more difficult to feel connected to your actions, because in reality these actions would center around a physical object. Pantomiming them doesn't cut it.

It's clear that each solution has its strengths, and areas in which it excels. But what I'm seeing right now is some strong me-too-ism. The first titles offered on both consoles are very much in the vein of Nintendo's biggest titles for the Wii, or the third party successes, regardless of whether it fits the system. The dance, exercise, minigame, and pets genres are all well represented, and I guess that's a start. But to really succeed here, developers are going to have to figure out how to maximize the unique qualities of these systems.

THE PROXIMITY PROBLEM

>> One similarity between Move and Kinect, which differentiate both from the Wii, is that they use cameras. It's interesting technology, but also represents a curious limiting factor.

In order to get Move or Kinect to recognize you correctly, you've got to be about seven-to-eight feet away from your television. You also need to have a clear, unobstructed view, meaning you've got to move the coffee table or ottoman.

I live in an urban environment, as many game developers do. I also am not the richest human being on the planet. With that combination, getting seven feet from my television is a bit of a challenge. I don't have a whole lot of space to move things around, and I really don't want to move my couch every time I play a game.

As the online space has taught us, any barrier to entry significantly limits your audience. So instead of potential players being those who own a Xbox 360 or PS3 and can afford a Kinect or Move (or buy the whole package if they have neither), layered into who actually wants the device, now you have the added element of "can it function in my home?" That's a big deal. At each layer, you lose a portion of your audience.

Take the entire nation of Japan, for instance. Very few people live in homes that are large enough to accommodate a seven-foot buffer without reconfiguring their entire living space. Urban dwellers across the U.S. and Europe face similar issues.

In the U.S., the persons who will be most able to use the Move and Kinect live in the suburbs, and own or rent entire homes. This, by and large, means families. Both companies have specifically singled out families as the group they want to market their new devices toward, but due to the space requirements, they may accidentally be ensuring that those are the only people who can play with these technologies.

—Brandon Sheffield

gamedeveloper
www.gdmag.com

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

SUBSCRIPTION SERVICES

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

FOR DIGITAL SUBSCRIPTION INFORMATION
www.gdmag.com/digital

EDITORIAL

PUBLISHER
Simon Carless l *scarless@gdmag.com*

EDITOR-IN-CHIEF
Brandon Sheffield l *bsheffield@gdmag.com*

PRODUCTION EDITOR
Jeffrey Fleming l *jfleming@gdmag.com*

ART DIRECTOR
Joseph Mitch l *jmitch@gdmag.com*

PRODUCTION INTERN
Tom Curtis

CONTRIBUTING EDITORS
Jesse Harlin
Steve Theodore
Giacomino Veltri
Soren Johnson
Damion Schubert

ADVISORY BOARD
Hal Barwood **Designer-at-Large**
Mick West **Independent**
Brad Bulkley **Neversoft**
Clinton Keith **Independent**
Brenda Brathwaite **Lolapps**
Bijan Forutanpour **Sony Online Entertainment**
Mark DeLoura **Google**
Carey Chico **Independent**

ADVERTISING SALES

GLOBAL SALES DIRECTOR
Aaron Murawski e: *amurawski@think-services.com*
t: 415.947.6227

MEDIA ACCOUNT MANAGER
John Malik Watson e: *jmwatson@think-services.com*
t: 415.947.6224

GLOBAL ACCOUNT MANAGER, RECRUITMENT
Gina Gross e: *ggross@think-services.com*
t: 415.947.6241

GLOBAL ACCOUNT MANAGER, EDUCATION
Rafael Vallin e: *rvallin@think-services.com*
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER
Pete C. Scibilia e: *peter.scibilia@ubm.com*
t: 516-562-5134

REPRINTS

WRIGHT'S MEDIA
Ryan Pratt e: *rpratt@wrightsreprints.com*
t: 877.652.5295

AUDIENCE DEVELOPMENT

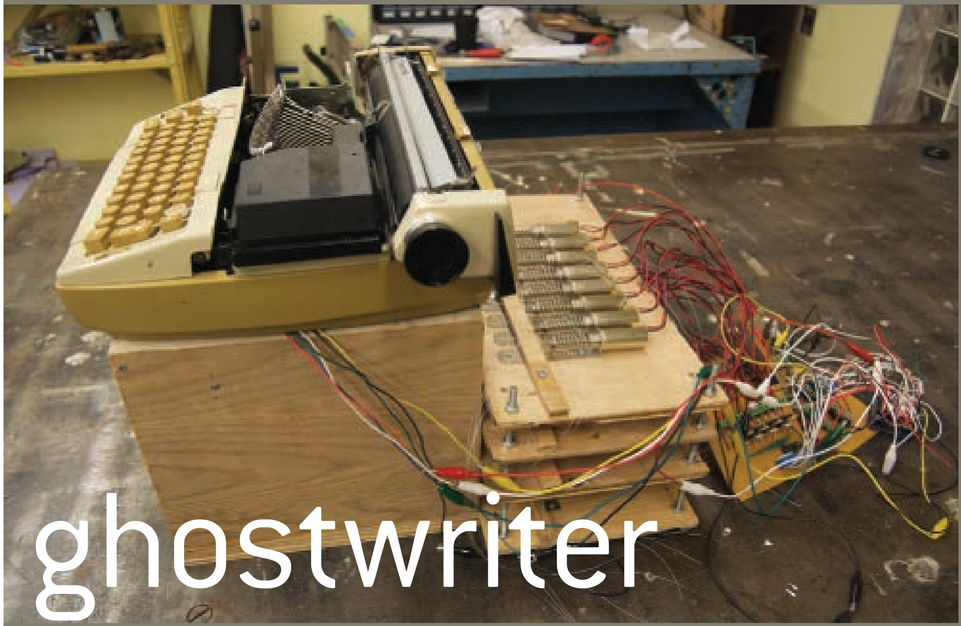
TYSON ASSOCIATES Elaine Tyson
e: *Elaine@Tysonassociates.com*
LIST RENTAL Merit Direct LLC
t: 914.368.1000

MARKETING

MARKETING COORDINATOR Nahal Agahi
e: *nahal.agahi@ubm.com*

United Business Media
WWW.UBM.COM

THE 2011
gametreetv
DEVELOPER COMPETITION
CHANGE THE GAME
FINALISTS DEMO AT GDC 2011
CELEBRITY JUDGING PANEL
OVER $50,000 IN PRIZES
TRANSGAMING
Sign up at http://gametreetv.com/competition



ghostwriter

/// The next time you find yourself lost in a maze of twisty passages, follow the sound of clacking typewriter keys. Interactive text adventures are moving off the computer screen and into the real world of ink and paper thanks to the hacking skills of Toronto resident Jonathan Guberman. Currently in prototype phase, Guberman's Automatypewriter uses a modified electric typewriter to output text from ZORK in real time, as well as pass typed commands from the user back to the program.

Working as a kind of electro-mechanical puppet, each of the electric typewriter's keys is connected via a network of fishing line to banks of solenoid switches. When the computer running ZORK sends out a signal for the typewriter to type a line of text, a connected Arduino board uses IC-based shift registers to assign the key presses to the appropriate solenoid switches. When a given solenoid fires, it pulls against the fishing line connected to a key, resulting in a letter being typed. A bit of timing correction on the computer side ensures that the automatic typing follows a natural, human-like rhythm.

Because the Automatypewriter is designed for playing ZORK in real time, resistors positioned under each key of the typewriter can detect commands typed in by the user. Signals are passed through shift registers back to the Arduino board, which sends the key press information to the computer. Guberman wrote a Python script to handle serial communication to and from the Arduino board and ZORK runs in a slightly modified version of the Rezrov (http://edmonson.paunix.org/rezrov) Infocom game interpreter.

An important design goal on the project has been Guberman's desire to implement the Automatypewriter's controls without significantly altering the host typewriter's mechanics—a constraint that will allow him to use a variety of antique typewriter models. "Now that I have it working successfully, I've invested in an Oliver No. 9 typewriter that dates circa 1915. The big issue here is going to be the carriage return, because on most manual typewriters you had to do this by hand. So, I've been working on ways to attach a motor drive to automate that process. That's the biggest change on the horizon," Guberman told us.

"I'm swapping out the fishing line for metal jewelers' wire, and I'm changing the structural setup to make it easier to adjust without taking the entire thing apart and putting it back together, a process that currently takes me about four hours. All of the mechanics are going to be hidden to the user. It'll just look like a typewriter sitting on a desk, that happens to be able to type by itself," he said.

While the Automatypewriter is currently set up to run ZORK, other possibilities exist and Guberman is collaborating with Jim Munroe (EVERYBODY DIES) to create unique interactive fiction for the machine. "I'm planning to write it with Inform 7, probably as a z machine file as is common for text adventures. We're going to test various approaches extensively on the platform itself rather than assume we know what will work," Munroe said.

To see the Automatypewriter in action visit Jonathan Guberman's blog at http://upnotnorth.net/projects/typewriter. Also check out Site 3 coLaboratory (http://site3.ca) and Hacklab.TO (http://hacklab.to) to see what interesting new projects Toronto's vibrant maker community is brewing up.

—Jeffrey Fleming

m.u.l.e. papers donated

/// The International Center for the History of Video Games (ICHEG) at the Strong National Museum of Play in New York has obtained a collection of notes and papers documenting the career of gaming industry pioneer Dani Bunten Berry (Dan Bunten).

Bunten, a Hall of Fame inductee at the Academy of Interactive Arts & Sciences, created the landmark multiplayer strategy title M.U.L.E. for computer platforms in 1983. Bunten was also involved in the development of titles such as ROBOT RASCALS, HEART OF AFRICA, and CARTELS & CUTTHROATS.

Bunten's children donated the papers to ICHEG and they have formed a company called Ozark Softscape (www.ozarksoftscape.com) that is dedicated to Bunten's legacy.

"It is a privilege to have our dad's work at a place that truly understands what 'play' is all about and our dad would be humbled and honored," said Bunten's daughter, Melanie Bunten Stark.

The Bunten donation follows ICHEG's recent acquisition of a collection of personal notes belonging to SIMCITY creator Will Wright. ICHEG plans to display both collections at Strong's upcoming eGameRevolution interactive exhibit (www.icheg.org).

—Danny Cowan

notes from the igda leadership forum

The annual IGDA Leadership forum was held this past November in Burlingame, California. Here are some of the highlights from talks given at the two-day event.

(ON SMART MANAGEMENT)

/// Laura Fryer (WB Games Seattle): Many leaders will throw their team under a bus in order to save themselves. They hide information because they're afraid of telling their team the truth. By way of example, Fryer discussed when she was working on the Xbox 360 while in her former position at Microsoft Game Studios. Epic Games founder Tim Sweeney came up and said, "You have to add more memory to the box." It was a labyrinthine structure at Microsoft, so it was hard to know who actually makes the choices. At Microsoft, they call it "walking the dog"—you walk around, try to figure out who can help, and find out who knows about the problem you're having.

People weren't happy to talk about it, and it was "definitely a case of the messenger repeatedly being killed," Fryer said. "I had one executive say this was a big CLM (career limiting move) and I should just stow it. I said I didn't think I was going to have a career if the Xbox 360 failed, and I thought it was going to fail if we didn't get more RAM!"

(ON MANAGING R&D)

/// Lucien Parsons (4mm Games): 3D Realms announced DUKE NUKEM FOREVER in 1997. "They'd blown away the competition with DUKE NUKEM, and they wanted to do it again!" Parsons said. But they kept trying to leapfrog the competition, any time something new came out. "By focusing on always trying to get the best possible experience, the best possible graphics," they ignored the immediate for the long term, he said.

Companies that invest more in R&D have better margins than those who spend less, he says, which makes sense. But there's no actual correlation between that spend and the overall success, based on actual data. "We've seen games that come out and are based on some new tech feature, and it's really cool ... for about 10 minutes," said Parsons. "Because if they let the technology drive the game, and not the gameplay drive the game," then nobody cares.

(ON FINDING INSPIRATION)

/// Don Daglow (Stormfront Studios): "If I pick one thing people have said to me—by very famous game designers, highly respected people—it's, 'I have to stop beating myself up about ideas that don't work,' or 'I have to stop talking myself out of good ideas'," said Daglow.

He has heard many people say "I'm having internal dialogue with myself and I'm talking myself out of stuff."

"It's so easy to have a self-defeating dialogue. Deeply respected game designers whose names are famous are having these same conversations."

"Freeing the mad scientist in you is when you get the good moment that is not on the chart. That's when you get GUITAR HERO, WORLD OF WARCRAFT, games that change the rules."

"If I were to leave you with no other thing than this—when all of this logic brings you to an obstacle, that is not the moment to be stopped by logic, that is the moment to be seized by inspiration," said Daglow.

(ON THE AGILE WAY)

/// Kim Sellentin (Sega Studios Australia) and Ike Ellis (Zynga East): "Agile is actually a mindset and an approach to leadership," said Sellentin. The project leaders learn how to listen to the team's needs, becoming "servant leaders", and that's important to the project.

Scrum also opens up members of the team to autonomy, and, Sellentin said, "there is so much leadership on your teams and you just don't know it yet."

"When you're starting off with Scrum you're going to run into a lot of misconceptions about what you're doing. One of these is that scrum brings unmanaged chaos. This is really off-base," said Ellis.

"You'll know what people are doing at a good level of detail and, more importantly, the people doing the work will know what they're supposed to be doing." However, it does create more management overhead—the producer has to be totally aware of all tasks and progress on all tasks, for which Ellis relies on an Excel spreadsheet.

"I've tried a bunch of things for a backlog, but I always come back to Excel." With the help of version control and some macros, it is the best tool—he totally dislikes all custom Scrum tools he's tried for one reason or another.

While Scrum is full of process, the "secret," he said, is that "it's all about talking out loud. All of the trappings, what they have in common, are that they foster conversation between people who know what they're talking about."

(ON SOCIAL GAMING)

/// John Vechey (PopCap): While the era of "cheap, free, easy traffic" is gone, Vechey says the item-buy business model was "first really accepted in a broad way in Western markets" thanks to Facebook. And the Social Graph, "the ease of interacting with your friends" is tremendously relevant.

"If you take advantage of the Social Graph, then every single game will be made better," said Vechey. Take, for example, MINECRAFT, "It may be one of the most important games of the decade. MINECRAFT would be better if it had the Social Graph inside of it. I don't fault them for not doing this, they're indies. WORLD OF WARCRAFT would be better with the Social Graph from Facebook."

That's because interacting with your friends through games makes them more fun. Vechey recounted how he didn't realize his friend was playing LEAGUE OF LEGENDS for months because of the lack of Social Graph interaction, and when he did, he felt cheated out of fun experiences playing together he could have had.

"PEGGLE would be better with the Social Graph, even if you play through the single player experience, to see your friends' best shots, their best scores," said Vechey.

(IF YOU WANT TO DESIGN GAMES, YOU SHOULDN'T BE AN EXECUTIVE)

/// Danny Bilson (THQ): Ultimately, money-wise you are a bit beholden to your stockholders, he admitted. "But there's only one way that I know of to make the stock go up," he said. "It's to make a great game. What is your customer responding to? This gets lost in all kinds of nonsense."

Games have to be led by creative. "I don't really believe in collaborative art. But people say 'well we've got 200 people!' There has to be one vision though, and it has to be communicated to all those people. All those people have the ability to create within their disciplines."

"There was this gag in the past, where marketing would make a forecast," he said. "The forecast would dictate the budget. And the budget dictates the features and what you can do in the game. So they can change the forecast to manipulate what they want. Why are the non-artists in charge of the art? Makes no sense to me."

"I don't think games should be directed from corporate in any way," he concluded. "If you want to design the game, you should get in the studio. You shouldn't be in the corporate headquarters, and you shouldn't be an executive."

—Brandon Sheffield and Christian Nutt

CORRECTION: In last month's "*Game Developer* 50" we incorrectly attributed the art direction for Volition's RED FACTION GUERRILLA. Jasen Whiteside is in fact the art director for the game and we deeply regret the error.

FRONT LINE A

We're proud to announce the finalists for the 13th annual *Game Developer* Front Line awards—the essential awards for tools and software that make the world's leading games possible—in the following categories: Art, Audio, Engines, Middleware, Networking, and Programming/Production. In determining the winners of the 2010 awards we went through a multistep process. Open nominations were held in October for software that had been released or had been updated between September 2009 and August 2010. From that list we consulted with our advisory board to narrow the results down to five entries in each category. We then handed the nominees to over to you, the readers of *Game Developer*, via an online survey in November, so that you could have a voice in picking the recipients of the Front Line Awards. As this issue goes to press, we're compiling the results, and in our January 2011 issue we'll reveal the winners. We'll also be inducting one special game development tool into the Hall of Fame that has been of enduring importance to the development community. Because the editors of *Game Developer* decide the Hall of Fame winner, it is not eligible in the regular categories.

Art

Photoshop CS5
ADOBE SYSTEMS
www.adobe.com

/// With its broad feature set and overall flexibility, Adobe Photoshop CS5 improves upon the established Photoshop suite by adding features that include the content-aware fill, HDR imaging, and puppet warp to supply artists with even more image manipulation tools.

Autodesk 3ds Max 2011
AUTODESK
http://usa.autodesk.com

/// The popular 3ds MAX modeling, animation, and rendering software provides a variety of tools for artists, with the latest version adding a node-based material editor, HDR-capable compositing tools, and other features to help accelerate modeling and texturing tasks.

Substance Designer 1
ALLEGORITHMIC
www.allegorithmic.com

/// Allegorithmic's Substance Designer is a procedural texture creation tool that uses a node-based authoring environment to quickly generate complex texture files. The resulting textures are resolution independent and can be dynamically altered at any point in the art pipeline including at runtime with the addition of Substance Air.

Autodesk Softimage 2011
AUTODESK
http://usa.autodesk.com

/// Softimage 2011 helps 3D artists create texture maps, and features a texture sequencer that allows users to manipulate bitmaps, vector graphics, and procedural elements to create more complex textures.

ZBrush 4.0
PIXOLOGIC
www.pixologic.com

/// ZBrush is a digital sculpting and painting program that features an interface that is more akin to sculpting than other 3D modeling software, allowing artists to create assets without getting bogged down by menus and obtuse options. The latest release aims to provide more features that allow artists to bring their models from concept to production within ZBrush itself.

Audio

FMOD Designer 4.32
FIRELIGHT TECHNOLOGIES
www.fmod.org

/// FMOD Designer offers a suite of options to help create high quality game audio. The software is fully integrated into several popular game engines, including Unreal Engine 3, Unity, and Scaleform, and allows bulk editing of multiple sound files at once.

Miles Sound System 8
RAD GAME TOOLS
www.radgametools.com

/// The Miles Sound System 8 middleware provides a versatile package that supports many types of audio files and runs on nearly all available platforms. RAD game tools prides itself on the software's stability and speed across all types of hardware.

ProRemote 2.0.1
FAR OUT LABS
www.folabs.com

/// Available for iOS devices, ProRemote provides 32 channels of remote control for digital audio workstations with a range of options that allow users to easily manipulate complex audio via touch screen.

ProTools 8.0.4
AVID TECHNOLOGY
www.avid.com

/// The ProTools music and audio production software has become practically an industry standard for recording and editing audio. It features a complete suite of production tools, including automatic delay compensation, and allows users to open projects created in other video or audio editing software, offering high flexibility.

Wwise 2010.1.2
AUDIOKINETIC
www.audiokinetic.com

/// Audiokinetic's Wwise audio pipeline allows sound designers and audio programmers to prototype projects quickly, and is integrated into many current game engines. Its latest version includes side-chaining for automatic volume adjustment and convolution reverb to simulate the way sound reverberates in an enclosed space, along with a host of new effects and options.

Game Engine

CryEngine 3
CRYTEK
http://mycryengine.com

Crytek's CryEngine 3 will power the studio's upcoming CRYSIS 2, and is available to license not only for game development, but also for interactive simulation and education software. The multiplatform engine allows for real time editing of game environments with its Sandbox technology.

Gamebryo Lightspeed 3.1.1
EMERGENT GAME TECHNOLOGIES
www.emergent.net

/// The Gamebryo Lightspeed engine emphasizes rapid prototyping, iteration, and real time updates, enabling developers to quickly put together playable content so teams can build upon their own work throughout the development cycle.

Unity 3
UNITY TECHNOLOGIES
http://unity3d.com

/// Unity Technologies' Unity 3 game engine has been engineered from the ground up to provide an integrated development framework

WARDS 2010 FINALISTS

that features graphical editing and live previews during runtime. In addition to a robust scripting, programming, debugging, and optimizing environment, the engine provides sophisticated rendering and lighting capabilities.

Unreal Engine 3
EPIC GAMES
www.unreal.com
/// Epic's Unreal Engine 3 has played a prominent role in multiplatform development over the past several years, powering a range of titles across a variety of genres. The engine builds upon previous iterations of the Unreal Engine, and supports features that include High Dynamic Range lighting, per-pixel shading, and dynamic shadows.

Vision Engine 8
TRINIGY
www.trinigy.net
/// Trinigy's Vision engine emphasizes developer freedom, supporting game development across a range of genres and platforms. The engine boasts exporters for the latest versions of 3ds Max and Maya, a streamlined scene editor, and integration with a host of middleware technologies.

Middleware

Box2D 2.1.0
BOX2D
www.box2d.org
/// Box2D is an open source physics engine for 2D games, and has powered titles such as CRAYON PHYSICS DELUXE and FANTASTIC CONTRAPTION. While the engine is primarily used for 2D games, it supports shapes including convex polygons and edge shapes, and applies gravity, friction, and resistance to objects.

Havok Physics 2010.1.0
HAVOK
www.havok.com
/// Havok Physics has been used in a multitude of titles across a wide range of platforms, and has proven to be flexible. According to Havok, it's designed "based exclusively on customer requirements." Havok Physics can be seen in a number of high-profile titles including UNCHARTED 2, DEMON'S SOULS, HALO: REACH, and JUST CAUSE 2.

Scaleform GFx 3.2
SCALEFORM
www.scaleform.com
/// Scaleform GFx is a Flash-based, vector graphics-rendering engine that is used for game elements such as UI, HUDs, animated textures, and minigames. Games featuring the Scaleform GFx middleware include BATMAN: ARKHAM ASYLUM, DRAGON AGE: ORIGINS, and STARCRAFT II.

Simplygon 2.9
DONYA LABS
www.donyalabs.com
/// The Simplygon pipeline toolkit aims to automatically optimize 3D content by automatically creating LODs, meshes, and low-poly models to help streamline the resources used in complex game scenes.

XaitControl 3.1
XAITMENT
www.xaitment.com
/// The XaitControl tool is designed to help design complex and streamlined AI behavior, and lets developers build behavior hierarchies that allow for efficient reuse of specific behaviors to conserve resources.

Networking

Facebook SDK
FACEBOOK
http://developers.facebook.com
/// Facebook's open source SDK allows developers to easily integrate the popular social networking service into a variety of apps and games, on platforms including mobile devices and Facebook itself. The Facebook SDK encourages social network integration and allows developers relatively easy access to player metrics.

GameSpy Technology
GAMESPY INDUSTRIES
www.poweredbygamepy.com
/// GameSpy Technology provides several important services for online-enabled games, including multiplayer connectivity via dedicated servers, peer-to-peer matchmaking, stat tracking, and cloud storage for user-created content.

OpenFeint 2.6
AURORA FEINT
www.openfeint.com
/// OpenFeint provides a persistent social network across a range of Apps on iOS and Android devices. It offers achievements, leaderboards, as well as features that allow end-users to track their friends across a range of supported applications.

Plus+
NGMOCO
www.plusplus.com
/// Ngmoco's Plus+ is a third party social network for Apps on iOS devices, allowing players to compete for high scores, earn achievements, and broadcast scores on Twitter or Facebook. Users can also challenge friends and send push notifications to challenge other users.

RightScale Cloud Management Platform
RIGHTSCALE
www.rightscale.com
/// RightScale provides cloud management services that allow developers to scale the servers and resources for their online game based on user demand. Developers can get their game running on the cloud quickly, and can then scale their game to optimize running costs.

Programming and Production

FlashDevelop 3.2.2 RTM
FLASHDEVELOP PROJECT
www.flashdevelop.org
/// The FlashDevelop open source code editor is a free Microsoft .NET application that supports ActionScript 2, ActionScript 3 & MXML, and HaXe, and includes simple integration with Flash and command line compilers.

Graphics Performance Analyzers 3.0
INTEL
http://software.intel.com
/// Intel's Graphics Performance Analyzers are a suite of software tools that provide performance data across different types of hardware in order to help developers tailor their games for the ever-evolving desktop PC and mobile spaces.

Hansoft 6.1
HANSOFT
www.hansoft.se
/// Hansoft provides infrastructure for more efficient development, and supports realtime reporting, bug tracking, workload coordination, and portfolio and document management. The software aims to manage the complex elements of game development within a single system to create better scheduling and productivity.

Perforce 2010.1
PERFORCE SOFTWARE
www.perforce.com
/// Perforce is a scalable source code management system that helps organize digital assets by storing them on a server and tracking user activity to help teams easily access their art assets, code, bug reports, and more across Windows, OS X, and Linux.

XNA Game Studio 4
MICROSOFT
http://create.msdn.com
/// Microsoft's XNA Game Studio 4 allows indie developers to easily publish their games on the Xbox 360 and PC using a simple, yet flexible toolset.

ADVERTISEMENT

UNREAL TECHNOLOGY NEWS

BY Mark Rein
Epic Games, Inc.

NINJA THEORY USES UNREAL ENGINE 3 TO DOUBLE DOWN ON CREATIVITY

Coming off the success of their critically acclaimed PlayStation 3 action game, *Heavenly Sword*, the developers at Ninja Theory had a slightly different plan in mind for their next adventure. "With *Heavenly Sword*, we had developed the engine and toolset entirely ourselves," says Mike Ball, co-founder and chief technical officer. "That required a large amount of resources, and it would take just as much effort to update the technology, improve the toolset and add the new features we needed for our next game."

Once Ninja Theory decided to go multi-platform with the new game, *Enslaved: Odyssey to the West*, they made the move to Unreal Engine 3. This time around, Ninja Theory had 80 people on hand for the production period, and would be able to use all that talent to focus on creativity rather than engine technology. "Unreal Engine 3 allowed us to start iterating on the gameplay much more quickly," Ball says. "We needed to maximize our production time by making sure the team's workflow was effective, and UE3 was the right choice to do that."

Using UE3, the Ninja Theory team was able to improve on many of the lessons they learned with their previous game. "We wanted an engine that was truly focused on getting the most out of designers and artists," explains Ball. "Despite the awesome results we achieved with *Heavenly Sword*, that was really the weak point of our own engine. It just didn't scale."

Ninja Theory also wanted to improve on the total length of gameplay with *Enslaved*, which is where Unreal Kismet came into play. Ninja Theory's designers used Kismet's visual scripting tools to build much larger play environments for players to explore. Kismet also enabled the team to create a wider variety of gameplay across the greater game length, which Ball said is ultimately the key to keeping the player attached to the story.

Ninja Theory has built a reputation for creating beautiful environments – an ability that would be put to the test with *Enslaved*. *Enslaved* is based on a 400-year-old Chinese novel called *Journey to the West*, which is itself based on much older folklore. Enslaved brings this ancient story to a future New York City. To do justice to this rich source material while bringing it to life in a hyper-modern environment, the Ninja Theory team knew it needed to push its graphics capabilities as far as they could go.

Ball says that UE3 gave the artists much more control, and with its advanced production tools the team was able to pack more detail into the environment. "This was really important for the portrayal of the New York we wanted to build," says Ball. "Through elements of the environment, we wanted the player to experience the story of people in the past, caught in destruction and trying desperately to escape. Giving the art team control over shaders and post-processing chains allowed us to set up some beautiful scenes that contrasted well with the scenes of destruction."

Ninja Theory also built on top of UE3's advanced combat capabilities. "We did a lot of experimenting with the camera to make every hit feel like it counts and draw you into the drama of combat," said Tameem Antoniades, Ninja Theory co-founder and chief design officer. "We used close-up shots during takedowns and finishing moves to really show the emotional impact combat has on Monkey, the main character. It's more of a cinematic twist on combat mechanics."

Along the way, the Ninja Theory team was able to connect with other Unreal licensees through the Unreal Developer Network. In fact, Ball says that the extensive community support has been a strong point of working with UE3. Other developers were always happy to help with tough questions, and Ball recalls lots of occasions where a technical query was answered by a studio on the other side of the world before anyone else even touched it.

The result? One of the freshest, most beautiful games of 2010, boasting longer, more intense gameplay than any of Ninja Theory's previous games. The game was launched in October to wide acclaim.

Thanks to Ninja Theory for talking with freelance reporter John Gaudiosi for this story.

Canadian-born Mark Rein is vice president and co-founder of Epic Games based in Cary, North Carolina. Epic's Unreal Engine 3 has won Game Developer magazine's Best Engine Front Line Award four times along with entry into the Hall of Fame. UE3 has won three consecutive Develop Industry Excellence Awards. Epic is the creator of mega-hit "Unreal" series of games and the blockbuster "Gears of War" franchise. Follow @MarkRein on Twitter.

UPCOMING EPIC ATTENDED EVENTS

Montreal International Game Summit
Montreal, Canada - Nov. 8-9, 2010

Game Connection
Lyon, France - Nov. 16-18, 2010

Dubai World Game Expo
Dubai, UAE - Nov. 29-Dec. 1, 2010

DICE Summit
Las Vegas - Feb. 9-11, 2011

GDC
San Francisco, Calif.
Feb. 28-Mar. 4, 2011

Please email: mrein@epicgames.com for appointments.

WWW.EPICGAMES.COM

POWERED BY UNREAL TECHNOLOGY

EPIC GAMES

fragged

procedural generation of fragmented meshes

ROBERT PERRY AND PETER WILKINS

////////////////////// PROCEDURALLY GENERATING FRAGMENTED MESHES FOR IN-GAME DESTRUCTION CAN REDUCE COSTS AND IMPROVE GAME APPEAL. THE GENERATED MESHES NEED NOT BE BLOCKY OR BORING. NEW CONCEPTS PROVIDE INCREASED FLEXIBILITY AND CONTROL OVER CURRENT TECHNIQUES WHILE DECREASING PRODUCTION TIME.

EXPLOSIONS ARE FUN

Everyone likes explosions in games. Good explosions are big, noisy, and throw rubble everywhere. Explosions can be a powerful way to communicate excitement, risk, drama, and scale to the player. Games are filled with explosions both big and small that emphasize story moments and draw the player's attention. For example, rocks tumble from cliffs, barrels explode, pipes burst, bridges collapse, and bombs detonate. Unfortunately, game explosions can draw a player's attention for the wrong reasons. Barrels and crates, in particular, tend to break in exactly the same pattern. Overly repetitive explosions can be a distraction that damages player immersion and game appeal.

CREATING CONTENT FOR EXPLOSIONS IS NOT FUN

Watching or causing explosions is fun. Creating lots of fragmented pieces for an explosion is not fun. Splitting meshes with binary operations or cutting tools is time consuming, labor intensive, and tediously repetitive. Many simple tools can be created to accelerate the process, but generally fragmenting meshes

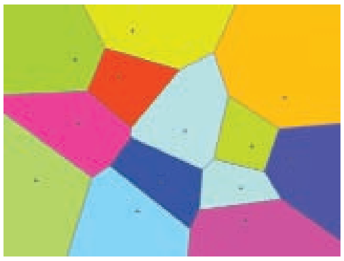


FIGURE 1 shows a two-dimensional Voronoi Graph.

remains an arduous task that no one wants assigned to them.

Cost and schedule constraints often limit the number of iterations possible for a given object as well as the number of unique objects that can be prepped for destruction. As the complexity of the original geometry increases, so does the cost of creating the destructible content that goes with it.

THE MAKINGS OF A GOOD EXPLOSION

Explosions are often implemented with a model swap where the mesh for the undamaged object is hidden, and meshes for the broken fragments are made visible in a single frame. Almost inevitably, there is a visible "pop" as the new pieces come into view. This is caused by the fact that the fragmented pieces don't perfectly match the original profile of the object being destroyed. In some cases, they don't really match at all. For example, several wooden shacks of different size and proportion may use identical fragments when they collapse. Some of the discrepancy can be hidden from the player by adding particle effects to cover things up. This can easily end up being a balancing act between covering the "pop" and obscuring the fragments altogether.

In order to avoid most of the "pop," the resulting fragments need to fit together tightly. Even if lots of smoke and dust particle effects are involved, the fragments need to at least maintain the original object's profile when assembled. Ideally, the fragments would fit together seamlessly. The exterior faces should have UV coordinates and materials that match the original. This makes for a fragmented mesh that's nearly indistinguishable from the original before being blown apart.

For an explosion to fit in context, the resulting fragments should reflect the material of the original object. Glass, stone, wood, metal, plastic, and other materials make up the objects that surround the player in a game world. If the fragments are only on the screen briefly, their shape may not be important, as they can be considered a large particle effect. However, if rubble stays persistent in the world after an explosion, then shapes that are true to the material of the original object become more important. Since many of these meshes will only be visible for a few seconds at a time, it's hard to justify the time and cost required for detailed modeling. The whole process begs for automation.

ENTER VORONOI

3D Voronoi graphs provide an easy way to automate much of the mesh fragmentation process. Particle systems can be used to place points in a model that are then used as seeds for the Voronoi graph. A graph is generated and used to subdivide the model. The graph is generated by identifying the region of space where a given point has an influence greater than any other. The result is a collection of convex cells with planar faces that looks somewhat like a honeycomb. A series of cuts can be made on the planar faces of each cell in the graph. This can be implemented easily using binary operations on the original mesh. The resulting fragments fit together tightly.

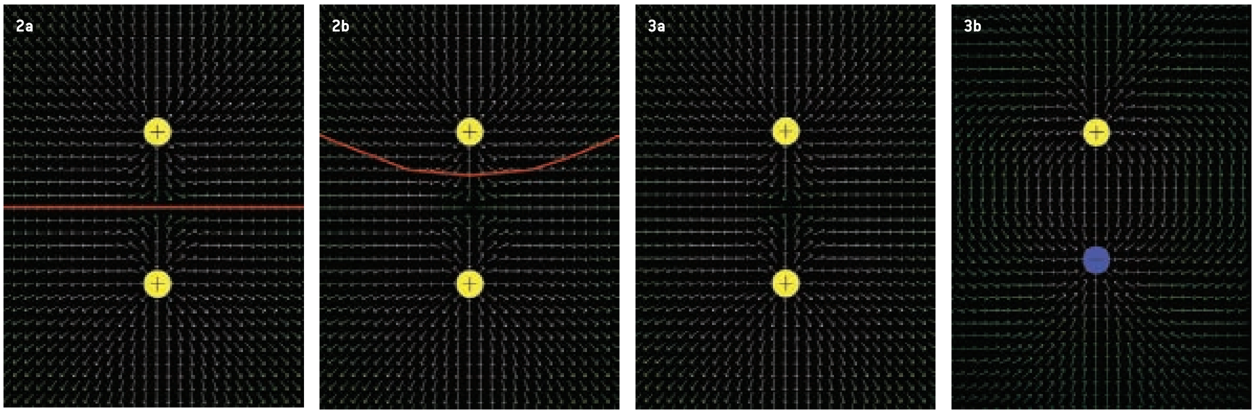
Another benefit of the Voronoi graph is that it allows for the creation of entirely interior fragments. This is a great feature for making rubble more realistic and interesting. Unfortunately, Voronoi graphs have some specific limitations. Voronoi cells are always convex and always have planar faces. The result is that Voronoi-generated fragments tend to be blocky and uniform, much like breaking salt. This works well if the original material is granite or safety glass, but does not work as well if the object is wood or plastic. Games are rarely full of objects made out of granite, so it would be convenient to be able to automate the generation of appropriate fragments for other materials. Figure 1 shows a two-dimensional Voronoi Graph.

MOVING BEYOND VORONOI

In order to create fragments with more complex shapes, a different mathematical approach is required. Fortunately, an alternative that is fundamentally very similar can be used. Electrical potential fields provide all of the functionality of a Voronoi graph, and allow for new possibilities as well. Electrical potential fields can create complex and concave geometry with a simple input set similar to what would be used to generate a Voronoi graph. This can be accomplished by adding a small amount of additional data to the very same input set that would be used to generate a Voronoi graph. The additional data is easily generated and takes little or no work on the part of an artist. For quick review, the electrical potential at a point in space is:

$$V_E = \frac{q}{4\pi\varepsilon_0 r}$$

Where q is a point charge, ε_0 is the electric constant (permittivity of free space), and r is the distance to the point charge. If there are multiple point charges in the field, they



FIGURES 2a and 2b show the effects of varying the charge of a point charge in two dimensions.
FIGURES 3a and 3b show the effects of polarity on a two-dimensional electric field.

obey superposition, and the potential at any point in space is simply the sum of the individual potentials.

$$V_E = \sum_{1}^{n} \frac{q_i}{4\pi\varepsilon_0 r_i}$$

It is now easy to subdivide the space by using the electrical potential field. Each point charge is assigned to own the region of space where its influence is greater than that of any other point charge. Using these equations and this one rule produces results identical to a Voronoi graph, assuming that all of the point charges in the field have uniform charge. Like a Voronoi graph, the electrical potential field is deterministic and will produce consistent results from a given input set.

The electrical potential field reaches beyond the limitations of the Voronoi graph by allowing for each point charge to have a different charge. This allows for the creation of curved fragments. Figure 2 shows the effects of varying the charge of a point charge in two dimensions. The red line shows part of the bounding line between subdivisions.

In the first case, both point charges have equal charge. In the second case, the point charge at the bottom has twice the charge of the point charge at the top. The boundary between the subdivisions moves and becomes curved under the increased influence of the point charge at the bottom. As this concept is extended to a large number of point charges and three dimensions, it provides for the creation of interesting and varied geometry. The result is that each point charge produces a manifold instead of a line.

An electrical potential field, working on an input set used for a Voronoi graph, can produce a nearly limitless number of results by simply modifying the charge of the individual point charges. This is a powerful feature that gains strength as other features are added to the system. For example, an artist could determine how many fragments are desired and place point charges to approximately locate the centroid of each fragment. The artist can then iterate on the final result by generating charge values pseudo-randomly, or by directly modifying them. This hybrid approach allows artists a great deal of control without spending large amounts of time on each object. It also allows for significantly more iterations than would be possible if the mesh were being fragmented by hand.

NORMALS

One major issue with the electrical potential field as used so far is that while it defines a bounding volume, it doesn't provide surface normals. Without normals, many geometric operations, such as meshing, become difficult. Surface normals can be produced using the vector form of the previous equation or the electric field equation.

$$E = \sum_{1}^{n} \frac{q_i}{4\pi\varepsilon_0 r_i^2} \hat{r}_i$$

Since the electric field equation introduces a direction to the field, polarity must now be dealt with. In the calculation of the potential field, it was easy to assume that all charges were positive in order to calculate the region of greatest influence. If the electric field were calculated in the same manner, all of the field vectors at the surface would lie tangent to the surface and not normal to it.

An easy remedy to this issue is to set the point charge of interest to be positive and all others to be negative. This directs the field so that the electric field vectors will be normal to the potential surface. Not only does this provide surface normals, but it also defines a full three-dimensional vector field throughout the volume of the electric field. Figure 3 shows the effects of polarity on a two-dimensional electric field.

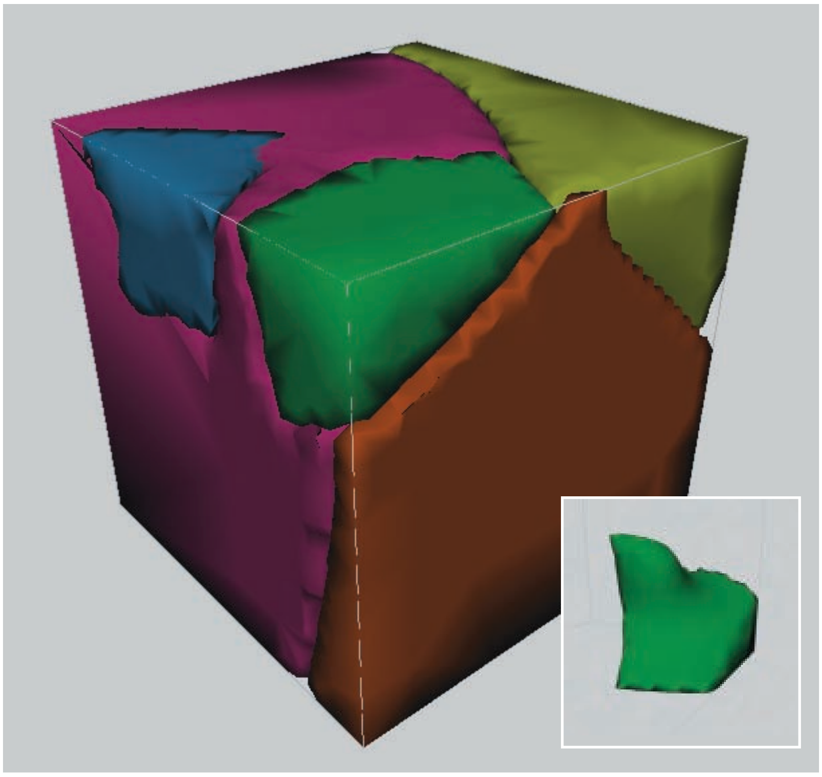


FIGURE 4a shows a cube fragmented using four splines. FIGURE 4b (inset) shows the interior detail on one of the fragments.

GROUPS ARE MORE POWERFUL THAN INDIVIDUALS

Electric fields provide increased flexibility over Voronoi, as they allow each point charge to have an individual charge that acts as a weight in addition to the packing of the points. This is a noticeable improvement, but the electric field has more potential to unlock (pun intended). Whereas, Voronoi graphs generally only deal with points—any arbitrary geometry can be electrically charged.

It stands to reason that seeding the subdivision with geometries other than points will create more interesting results. This turns out to be a valuable concept, as it allows for the creation of complex geometries from relatively simple ones. In order to treat arbitrary geometries through constructive analysis, each geometry will be considered a collection of a finite number of point charges. The resulting geometry is simply the superposition of the constituent point charges.

There are many useful geometries that can be generated in a pseudo-random manner to create interesting shapes. One geometry in particular that's both easy to generate and effective in creating a wide range of shapes is a spline. A cubic Bezier spline only requires the generation of four control points. Splines can be used to approximate a wide range of

LISTING 1

```
float Destruction::TransferUV::FaceDotDist(const Vector3& v1, const Vector3&  v2,
const Vector3&  v3, const Vector3&  point, const Vector3&  normal)
{
     Vector3 edge1 = v2 - v1;
     Vector3 edge2 = v3 - v1;

     Vector3 faceCentroid = v1 + v2 + v3;
     faceCentroid /= 3.0f;

     Vector3 faceNormal = edge1.Cross( edge2 );

     return (faceCentroid-point).Length()/fabs(normal.Dot(faceNormal));
}
```

LISTING 2

```
struct UV{
     float u,v;
};
struct ST{
     float s,t;
};
void Destruction::TransferUV::TriangleST_ToMeshUV( const UV& v1, const UV& v2, const
UV& v3, const ST& st, UV& o_uv )
{
     Vector2 edge1 = v2 - v1;
     Vector2 edge2 = v3 - v1;

     o_uv = v1 + st.s*edge1 + st.t*edge2;
}
```

geometries. A spline is sampled along its length to produce a group of point charges. The number of point charges used to represent the spline is a function of the desired resolution. Figure 4 shows a cube fragmented using four splines, including interior detail on one of the fragments.

Splines make the automated generation of smooth organic shapes easy and fast. They can also be authored by hand quickly, allowing artists control over the resulting shapes without spending time on details. They also provide fast and easy iteration. An artist can author splines to fragment a mesh and edit the splines afterwards, reprocessing until a satisfactory result is achieved.

IMPLEMENTATION

The implementation shown here is generally used as a rubble generator, and as such, produces rough and blocky seams. This is not a mathematical limitation of the electric field, but a practical limitation of the mesher that is used to create the rubble. Still, it provides a means to visualize the results. In this implementation, an entirely new mesh is generated using the union of the original mesh and the potential field. The resolution is intentionally limited, and this limitation contributes to the blockiness of the result. For the purposes of rubble, it works well.

UV MAPPING

Automated UV mapping is a crucial feature for the automated fragmentation of meshes. UV mapping can be terribly time consuming. Also, it can be highly obvious if the mapping on the fragments was rushed and doesn't match the original. If the meshes are very similar, the mapping can be copied from one mesh to the other. If the meshes are dissimilar, an interpolation scheme is required. Here is a simple method that works well with some limitations. In the described implementation, it's assumed that meshes contain only triangles and the vertices are unique to each face. The concepts can be extended to handle quads and other polygons as well as triangles.

Vertices in the new mesh are iterated over by face. Faces in the original mesh are ranked against vertices in the new mesh using values produced by a routine called `FaceDotDist()`. `FaceDotDist()` compares the distance from the vertex in the new mesh to the centroid of the face in the original mesh, and the normals of the faces in both meshes. The face with the lowest returned value is used for the interpolation. See listing 1.

Once a face in the original mesh is selected to interpolate from, the UV values at each of its vertices are used to perform the interpolation. The vertex from the new mesh is parametrized in the space of the face of the old mesh as a linear combination of two of the edges. See below:

$$p = se_1 + te_2$$

Where p is the vertex from the new mesh, and e_1 and e_2 are edges in the triangle from the original mesh, s and t are parameters along e_1 and e_2. This parametrization is commonly used in ray-triangle intersection tests and can also be used to verify that the vertex being parametrized is inside the triangle from the original. This verification is usually not necessary. While interpolation works best when the vertex falls inside the triangle, extrapolations a short distance outside the triangle are generally well behaved. The final UV value assignment is handled by the following function. Note that v1, v2, and v3 are UV values and not geometry vertices. See Listing 2.

This is an effective method for automatically transferring UV mappings between meshes. The primary limitation is that it cannot properly handle a face in the new mesh if it crosses a seam in the original mesh. In this case, it

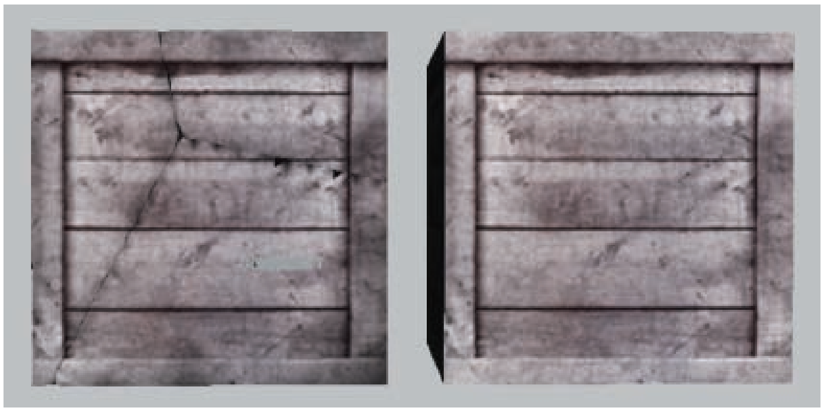


FIGURE 5 shows a fractured crate with interpolated UV mapping.

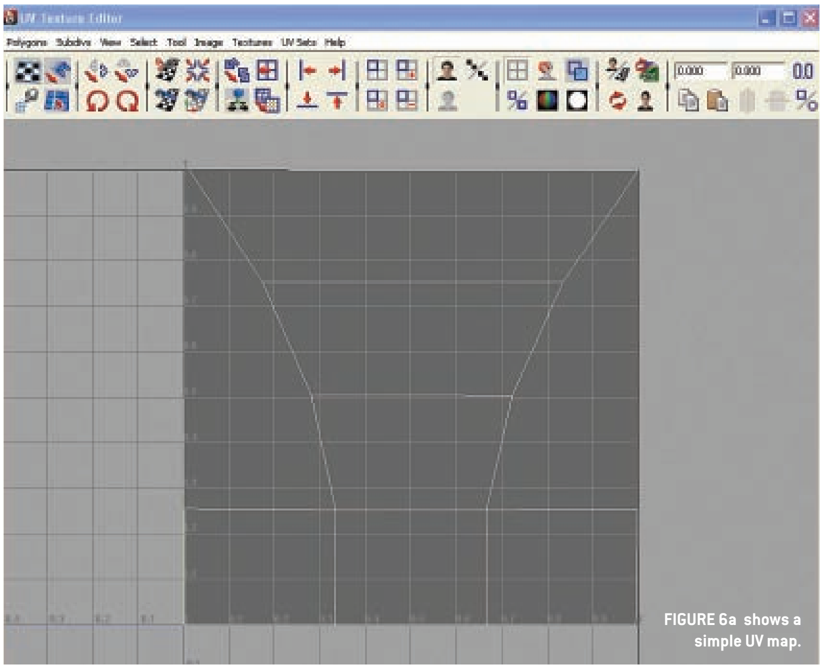

UV Texture Editor
Polygons Subdivs View Select Tool Image Textures UV Sets Help

FIGURE 6a shows a simple UV map.

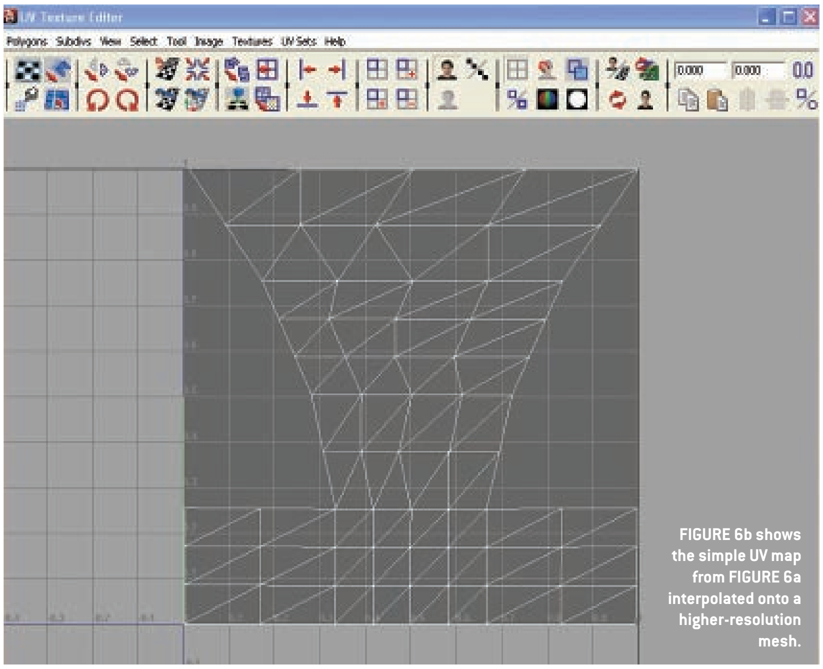

UV Texture Editor
Polygons Subdivs View Select Tool Image Textures UV Sets Help

FIGURE 6b shows the simple UV map from FIGURE 6a interpolated onto a higher-resolution mesh.

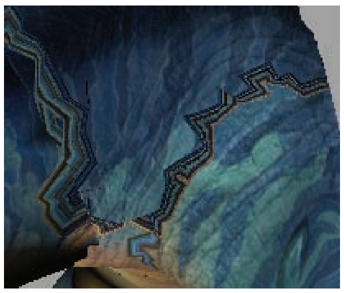


FIGURE 7 shows typical artifacts when faces in the new mesh cross seams in the original mesh.

produces obvious artifacts in the mapping. Problems at the seams could be corrected by identifying faces that cross seams and splitting them at the appropriate location. Also, it's an m×n order routine, where m and n are the number of faces in the respective meshes. For small face counts, this is fine, but as the meshes grow, run times can be a concern. This can be avoided through the use of a bounding volume hierarchy or similar construct to speed up the search through the original mesh. Also, if the new mesh is a lower resolution than the original, there are limitations as to how well the interpolation can cope with the reduced resolution. Figure 5 shows the results of a fractured crate with interpolated UV mapping. Figures 6a and 6b show a simple UV map and its interpolation onto a higher-resolution mesh.

This technique does not work in all cases, but it works in many. Particularly for world population items like barrels and crates, it tends to work well. It's also relatively easy to author them so that issues with seams can be avoided. In more complex cases, even though there can be issues with seams, the vast majority of the mapping can be properly transferred, leaving only small amounts of artist intervention to finish the process. This still yields a significant time savings. Figure 7 shows typical artifacts when faces in the new mesh cross seams in the original mesh.

FREEDOM TO DESTROY

Automated generation of fragmented meshes for destruction can not only reduce the costs associated with explosions in the game, it can do so while providing the opportunity to increase the variety and depth of what would otherwise be repetitive and potentially boring experiences. Increased iteration and variety with decreased production times are all possible by turning most of the work over to software. The price is that while there's a great deal of control over the process, it can be difficult to use if very precise results are required. The best fit is where the system can produce results that mimic a natural process.

In gaming, explosions do much to add interest, excitement, and drama to a game. But, for the artists tasked with creating realistic explosions, it can be difficult and costly, in terms of production and time, to fragment the meshes in unique and non-repetitive ways. Fortunately, new approaches now provide greater flexibility and control for creating unique, realistic fragmentation of meshes, thereby increasing game appeal, decreasing production time, and lowering the overhead typically associated with creating game explosions.

ROBERT PERRY *is a physics and simulation coder, and biking enthusiast.* **PETER WILKINS** *is an animation and rigging engineer at Dreamworks Animation. Both previously worked at Disney's Avalance Software.*

Image courtesy of Ninja Theory

NINJA THEORY

USES MORPHEME

“Today’s games require more than just realistic graphics – character animations need to be smooth, lifelike and convincing in order to really bring gamers into our worlds, and Morpheme delivers the level of control required for that aspect of game development.”

Mike Ball, Chief Technology Ninja

morpheme advanced animation system

www.naturalmotion.com

ILLUSTRATION BY JUAN RAMIREZ

FULL REACTIVE EYES ENTERTAINMENT INCORPORATING QUICK TIME EVENTS INTO GAMEPLAY

TIM ROGERS

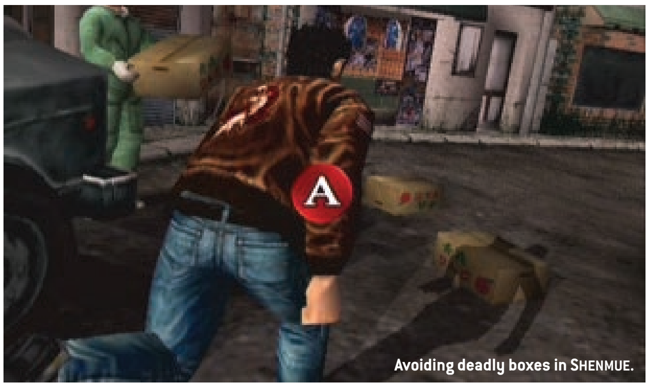
////////////////////// A "QUICK-TIME EVENT" (QTE) IS AN EVENT IN A GAME WHERE THE PLAYER MUST PRESS A BUTTON TO PERFORM A CINEMATIC ACTION THAT CAN OTHERWISE NOT BE PERFORMED IN THAT GAME IN AN ORDINARY CONTEXT.

Usually, when a QTE occurs in a game, normal controller inputs are overridden. If the X button on the PlayStation controller is usually used as the jump button, during a QTE, the X button can be substituted for any action the game designer requires. Using the X button in a QTE might result in the player character punching an enemy in the top of the head, dodging a bullet, or splitting an Intercontinental Ballistic Missile in half with a samurai sword.

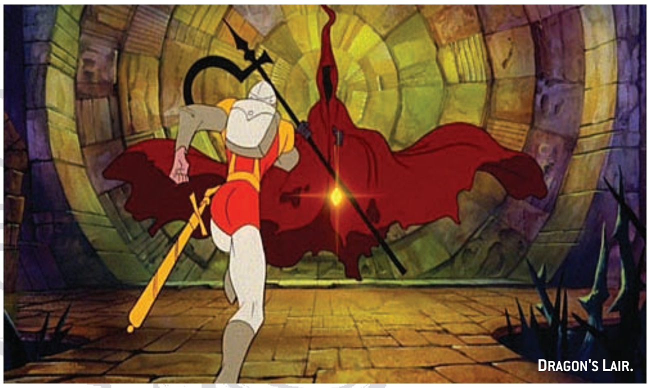
The sequence of a typical QTE involves normal controller input being taken away from the player for an instant before the on-screen action snaps into a cinematic camera angle. In an action game, this camera angle usually reveals an enemy threat. Next, a button icon appears somewhere on the screen. This icon stays in place only for an instant. If the player presses the button in time, the player character will avoid or neutralize the threat.

For example, in UNCHARTED (Naughty Dog, 2007), at one point, the hero Nathan Drake falls from a ledge and onto his back. Control is overridden; the camera angle swings up to show a large piece of rock breaking off the side of a cliff face and sliding toward the ground. A button icon appears. If the player presses the button, Drake rolls out of the way and the rock crashes violently onto the ground where he had just been. If the player doesn't press the button in time, a brief cutscene plays, portraying Drake's tragic death.

GEARS OF WAR's chainsaw attack is essentially a QTE.


Avoiding deadly boxes in SHENMUE.

GOD HAND's QTEs are contextual.


DRAGON'S LAIR.

More complicated QTEs might involve threat after threat raining upon the player. In this case, the player must press numerous buttons in sequence. Missing a button-press results in instant failure and possibly death. Some games, like NINJA BLADE (From Software, 2008), will allow the player to immediately restart the QTE upon failing once. Other games, such as SHENMUE (Sega, 1999), the game whose director Yu Suzuki coined the terms "Quick-Time Event" and "QTE," are not so forgiving. Failure at a QTE will result in player death and a game over condition.

Of Ninja Theory's game HEAVENLY SWORD (2007), in which QTEs are called "hero events," NINJA GAIDEN and DEAD OR ALIVE series director Tomonobu Itagaki told consumer magazine *EGM* that he had "never played a good game where the developers put a big icon of the button you're supposed to press onscreen." He said the game seemed "really half-assed, because it's asking you to do all these button-timing sequences," and the player is not "getting much payoff from it."

HEAVENLY SWORD director Kyle Shubel, in his game's defense, replied that "the intent of the hero sequences is to empower the player to experience events that would be nearly impossible to play in a natural platforming state ... for example, making the player run down ropes, leaping from rope to rope as they're being cut from underneath you, all while dodging other objects—that would be a frustrating experience to 99 percent of our users if we were to force them to do that manually."

This certainly seems to be the trend. QTEs replace actions that would otherwise be more complicated than any player, even the skilled ones, are able or willing to input with the basic methods allowed by an analog stick and a couple of buttons.

It's perhaps interesting to note that, despite their vocal stance on the virtue of QTEs, Ninja Theory's next game, ENSLAVED: ODYSSEY TO THE WEST (2010) employs not a single QTE, not even the kind where you hammer a button to open a heavy door. What happened?

CUTSCENE WITH A KNIFE

After hours of satisfying shooting at virus-infected high-speed-sprinting zombie-intelligence psycho-freaks, RESIDENT EVIL 4 (Capcom, 2003) climaxes in an extended knife fight between a hero and a villain. The exact nuances of a military-grade knife fight as seen in the climax of cinematic masterpiece *Under Siege* were, for certain, simply not expressible with then-modern video game control inputs. Knives are short blades, a fraction of the length of a human arm. The arm acts as a whip; the wrist rotates; the mind manipulates the blade to clash with the opponent's in defense, to feint, or to make a desperate stab. You really can't express this one-to-one in a video game using only buttons and analog sticks. The fight is long and elaborate—some critics might say too long, and fantastically elaborate.

The RESIDENT EVIL 4 knife fight takes place during a heated dialogue between the protagonist and antagonist. The dialogue involves the revelation of important story information—why who has been doing what to whom For All This Time, and what he wants to convince him to stop. Effectively, it serves a purpose of a cutscene. As a climactic moment in the story, it's a cutscene that players most likely wouldn't want to skip.

Players that do want to skip the cutscene are unable to though. The sequence contains a series of precise button-press prompts. If the player fails at inputting a button press, the antagonist kills the protagonist, and it's game over. The knife fight QTE is the most-hailed example of both the positives and negatives of the form. The suspense of the unraveling dialogue and story revelations place extra pressure on the antagonist's coming knife-lashes; the potential for quick death means the player may be forced to repeat the QTE, the cutscene, and the dialogue again from the start.

Other QTEs act to replace or supplement cutscenes. In SHENMUE, QTEs often occur at the height of a dramatic cutscene. Unlike the knife fight in RESIDENT EVIL 4, QTEs in SHENMUE are all action. In one scene, the hero (Ryo Hazuki) is chasing a group of biker gang members out of a bar and down an alley. The chase comes after a small conversation in which Ryo attempts to wrangle

The BURNOUT series took the fun of ROAD BLASTER and made it interactive.

information out of the gang members. The chase occurs as a spectacular action payoff. Unlike the knife fight in RESIDENT EVIL 4, the story revelations are over when this QTE begins. Also unlike the knife fight, if you miss a prompt during this QTE, you still have a chance to win. The QTE branches: at one point, the man you're chasing knocks over a box of fruit; if you don't dodge it, and instead trip, the QTE is effectively lengthened as you're offered opportunities in the form of more button prompt situations. In the context of the story, this means that the chase is longer, and the hero doesn't look as impressive as he would had he captured the character quickly.

Similarly, SHENMUE contains many QTE fight scenes full of intricately detailed karate maneuvers—grabs, holds, throws, dodges—that would be difficult to map to specific controller inputs. Miss a prompt, and the hero is punched in the face. That doesn't necessarily mean game over. The player has plenty more opportunities to win the fight. The fight grows long, the hero lands punches, misses punches, dodges punches, and takes punches. The longer the QTE, the more interesting, if not impressive, the fight. Of course, if you miss enough prompts, the hero goes down, and it's game over.

In both of these examples, the QTE is "replacing" a cutscene—in SHENMUE, it often replaces a cutscene that would follow another cutscene. The talking cutscene ends, and the punching QTE begins. This type of cutscene-replacement QTE is primarily a means for developers to impress players with dynamic action scenes. The knife fight scene in RESIDENT EVIL 4, on the other hand, is "enhancing" a cutscene. In an "enhancement" QTE, the developer is providing the player with a reason to invest himself in the story revelations of the cutscene.

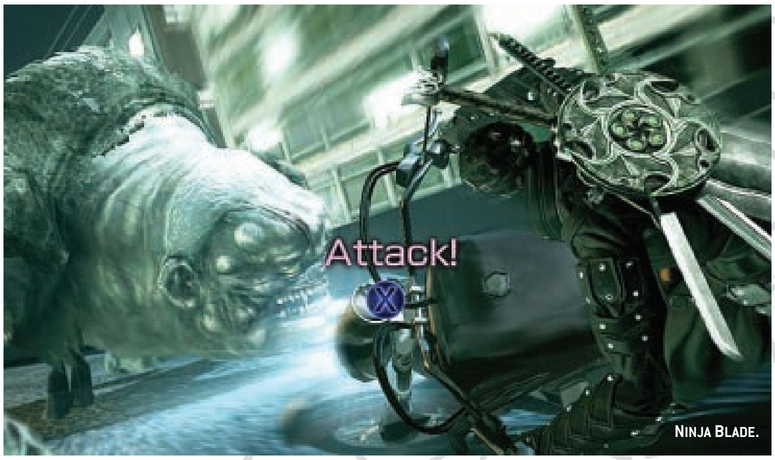
Other games, such as METAL GEAR SOLID 4 (Kojima Productions, 2008), will occasionally provide players with an on-screen prompt, which sometimes lasts no more than a fraction of a second. Press the action button during one of these prompts to view an alternate angle of the cutscene, or maybe view a piece of concept art of the character talking. In METAL GEAR SOLID: PEACE WALKER, succeeding at one such on-screen prompt results in the player being treated to a view of a female character's underwear. In this case, QTEs are rewards to the player for steadfastly paying attention to the game's narrative. In this way, perhaps QTE are used to safeguard against the common complaint that games like METAL GEAR SOLID feature too many cinematic sequences and not enough game-playing. This use of the QTE has created many critics' impressions that QTEs as a game-mechanic are interaction on the fringe of passivity.

FAILURE IS NOT AN OPTION

The very first QTEs were, in fact, a replacement for in-game action. The most obvious example is also the genesis of the modern concept of the QTE: the Laserdisc-based DRAGON'S LAIR (Cinematronics, 1983). In DRAGON'S LAIR, the player controls a knight on his quest to rescue a princess from a dragon. Though the story was common fare for any type of adventure story, the graphics were superbly unique compared to other games at the time. DRAGON'S LAIR was a lovingly hand-animated cartoon featuring the work of renowned animator Don Bluth. DRAGON'S LAIR's secret was that its data was stored on a Laserdisc. Player control inputs were limited to actions that, effectively, changed the chapter being played. At certain points in the action, an arrow or on-screen object flashes, either to the hero's left or right. The player has to respond in a split second. If we succeed, we see a brief animation detailing our hero's triumph. If we fail, we see our hero's demise. This game mechanic would be offensively shallow if it were the core of any games today. But at the time, with graphics so astounding, it worked. Part of DRAGON'S LAIR's appeal was that the hero's deaths—not just his triumphs—were unique animations. Dying is part of the game. Seeing each of the hero's deaths is as essential to earning encyclopedic knowledge of the game as seeing each of his triumphs.

We don't see games fully made up of QTEs anymore. However, we occasionally see games where the QTE becomes the main format of the game-action for an entire set piece.

BAYONETTA.



NINJA BLADE.

One infamous example of such a QTE usage comes in the game SHENMUE II (Sega, 2001)—the sequel to the game that brought the QTE abbreviation into the mainstream. At one point in the story, the hero and his buddy arrive at a dilapidated tenement building in Kowloon. The goal is to get to the tenth floor, where they have an appointment to meet someone. The hero goes ahead alone. Upon reaching the second floor, he finds that the floor is caved in, and the only way to get to the other side is to walk across a precariously positioned plank of wood. Step onto the wood, and the action QTE begins.

The camera is positioned just above the hero's shoulders as he stands on a thigh-wide wooden plank spanning a black hole in a gray-floored, brown-aired tenement building devoid of other life or sound. Arms held out at his sides, putting one foot in front of the other, he baby-steps forward across the plank. Every few steps, at randomly staggered times, he leans to one side or the other. An on-screen prompt urges you to press either left or right on the control pad. Soon, these prompts are coming in relentlessly. Miss just one, and the hero falls to his death. Falling to your death means game over. You reload the game, you endure the journey from your save point to the place of your death, and you try again. Succeeding at this particular mission flawlessly takes 10 grueling, palm-sweaty minutes of your life. Failing at it might take a dozen hours.

You have to walk across planks—sometimes two of them—on each of the ten floors of the building. This might be where you give up on the game, either because it's too difficult or because you've smashed your controller. If you succeed, the other character is waiting for you at the top. The hero, confused, asks how he got up there. He explains that he took the elevator.

The player has no option to take the elevator.

GOD OF BUTTONS

SHENMUE II's example of using QTE to replace game action, in theory, is purely out of DRAGON'S LAIR. In practice, it offers no neat graphical payoffs. Even death is unceremonious: the hero is swallowed into the void. It's a chore that must be completed to move forward in the game. QTEs are a powerful game mechanic in that they offer developers the opportunity to show the player something really cool—and that's why gamers play games: to see really cool things. Making a game sequence entirely out of QTEs means everything has to be very cool, and it's hard to make everything cool. It's like writing a sentence using only exclamation points. People get tired of that after a while.

The secret, then, is to use QTEs to enhance in-game action.

In GOD OF WAR (Sony, 2005), a shining example of enhancing in-game action, QTEs most often arise in the middle of climactic battles—not the cutscenes before or after said battles. In an early boss battle against a hydra, the player must dodge the enemy's attacks while attacking its weak points. Hit the weak points enough times, and you induce a vulnerable state. The player has a few moments to reach and attack the hydra head during this induced vulnerability. He must continue to dodge attacks while climbing the mast of a ship to reach the hydra's head.

Once in place, the player presses a button to initiate a QTE during which the hero lambastes the beast's head, swings around in an acrobatic arc, and ultimately pulls the head down with great force, impaling it on a broken, spiked wooden pole. Success at the QTE means destruction of one of the parts of the boss. If this were a game on the Nintendo Entertainment System, the boss's life meter would be made up of four rectangular segments: attacking the boss outside of the QTE would not decrease his life meter, while successfully completing the QTE would erase an entire segment.

The penalty for missing a prompt in the GOD OF WAR during-boss-fight QTE is ejection from the QTE: miss a move during the hydra fight, and Kratos plummets back to the deck of the ship. The hydra recovers his strength. You must now attack the boss as before until he's in a vulnerable enough position to initiate the QTE again.

FINISH HIM!

QTEs aren't just for bosses. They can happen in the middle of typical underling fights as well. GOD HAND (Capcom, 2006) employs QTEs of the button-mashing variety in the middle of standard fights. Sometimes, your hero will have an opportunity to get an enemy in a headlock. Press the button displayed on-screen with the proper timing to initiate the headlock. Now pound that button as hard as you can in the ensuing lock-up to inflict damage on the enemy. The faster you press the button while gripping the enemy, the faster the hero pummels, the more your controller vibrates, the more damage you do, and the more satisfied and inspired to pump your fist you become.

The crux of this kind of QTE is that it requires a timed button-press to initiate, and that that button is always the same button. In Japanese game development, all QTEs are most often referred to as "Action Button Events"—as in, you press a button, and you get action. Meanwhile, in games like THE LEGEND OF ZELDA: OCARINA OF TIME (NINTENDO, 1998), an all-purpose, context-sensitive button is called the "Action Button." For the GOD HAND flavor of QTE, where the same context-sensitive button is always used to enter these pummeling scenarios, the description "Action Button Event" is more apt than it may be elsewhere.

Compare the GOD HAND example with the "torture attacks" in BAYONETTA (Sega, 2009). The player plays the part of a witch who pummels angels to death with her hair in a fantasy realm. Every once in a while, during an angel-pummeling situation, a large button icon appears above the enemy. Press the button in time, and a large guillotine or iron maiden materializes surrounding the enemy. The blade falls, or the iron maiden snaps shut, and the enemy dies in a geyser of blood. It's like a MORTAL KOMBAT "Fatality," except it's happening during in-game action. It involves complicated machines materializing out of thin air, and it only requires a single button press. And that single button press is indicated on the screen, unlike MORTAL KOMBAT's arcane, mysterious, complicated Fatalities.

This reminds us that, even in the GOD HAND pummeling example, the button icon remains on the screen throughout the pummeling. This makes us realize why the button appears on the screen at the time of initiating the pummeling: otherwise, the player wouldn't know it was time to pummel. The enemies generally have no tells.

One game far ahead of its time with regard to this type of QTE was BERSERK: THE THOUSAND-YEAR OATH (Sammy, 2003). In that game, enemies have tells that indicate when the player can press the action button—normally the block button—to execute a spectacular parry and score a massive attack on the enemy. The tells are neither so subtle that you can only learn them through rote memorization (as in an old-school MEGA MAN game), nor are they so blatant that they see fit to throw a button icon up on the screen. Rather, they're near-subliminal: an ogre might raise a club above his head, and bring it crashing down toward you. If you're in range of receiving the attack, the screen action will freeze for the sticky, frictive instant before impact. This is your cue to press the block button and initiate the brutal, fast action button attack.

In light of this type of QTE, we could say that traditional QTEs, which halt the game to display button icons, are micro-tutorials. These micro-tutorials teach you how to do the precise thing the hero needs to do in the context of the current, complex situation a microsecond before he has to do it—or die. If the current situation calls for the hero to roll beneath a demon beast's blade before cartwheeling back in the opposite direction and then running up the blade toward the beast's face, the QTE-as-tutorial will instruct the player, for a moment, how to do that—with an alarming, screen-filling "X" button icon. This is directly in line with Ninja Theory's descriptor of QTEs as allowing the player to do things they couldn't do in regular game-action.

It might be construed that a QTE-as-tutorial is "introducing" a new action element to a game long after the traditional tutorial phase has ended. STAR WARS: THE FORCE UNLEASHED (LucasArts, 2008) often employs QTEs where the player character jumps about as high as he can jump in regular play, levitates objects about as heavy as he can levitate in regular play, or lightsaber-slashes about as ferociously as he can in regular play. As a tutorial, the QTE is merely "teaching" a nuance—one which will not be displayed elsewhere in the game.

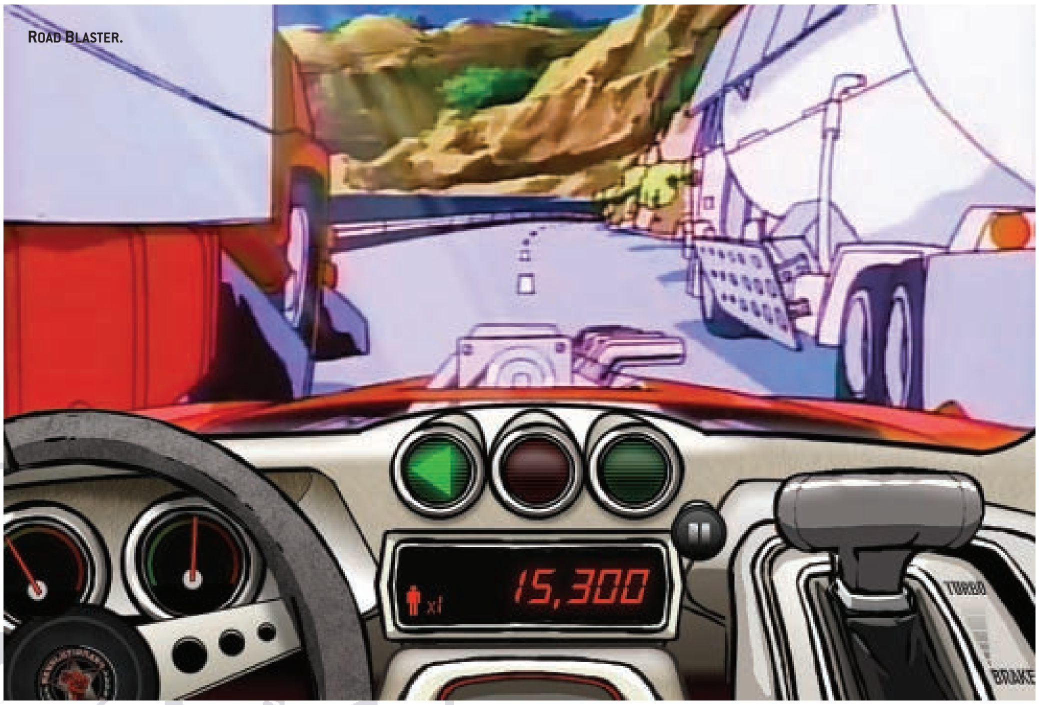
BUTTONS ON A STRING

It's possible to employ a "triggered" QTE like this simply, with great flair, with equal risk and reward, and in a manner that doesn't interrupt the game. A good example would be the chainsaw kills in GEARS OF WAR (Epic, 2006). You might not think of these as QTEs, but what else would you call them? In GEARS OF WAR, both you and the enemies have guns. You can shoot each other from a distance. Enemies can take four to five dozen bullets before dying. You and the enemies can crouch behind walls for cover from unfriendly gunfire. To arrange a chainsaw-kill, you need to orchestrate the situation. The entire level design is like an on-screen button prompt. Its geometry tells you where you need to be to stay out of enemy sight while your teammates suppress him. He's not going to stand up or leave cover, because that would be stupid. Jet-engine-volume gunfire echoing out your surround system, you sneak around, holding the chainsaw-revving button. You approach the enemy and are treated to a sudden, fast, furious, satisfying spray of blood.

Then we have the weird little disconnected button-mashing events. UNCHARTED has plenty of them: the player arrives at a door that cannot be opened. It's too heavy. It opens upward. The character puts his hands under the door. The character says "This door is heavy." Now a square-button icon appears on the screen. Hammer the square button, and our hero exerts himself until the door is open. The problem with these events is that they usually occur during a time of no conflict. The enemies are dead—or else we're in an undiscovered ancient ruin, and the developers feel the need to make the players push a whole lot of buttons furiously every once in a while or they'll get bored. (An aside to Naughty Dog: UNCHARTED's quiet parts are fascinating. It's cool; you don't need to punch them up.)

Then there are context-related situations, where a door requires cranks on either side to be rotated at once. This means you have to finish an ongoing fight in order to convince the non-player-controlled character to come to the door to aid in its opening. Once the fight is completed, the button-mash to open the door feels like dead air. The conflict is what kept you from doing it. Now that you can take your time, maybe the game should let you relish your victory instantly.

It's actually quite possible to place button-


ROAD BLASTER.

mashing QTEs in a strategic context. One fantastic example is in GEARS OF WAR 2 (Epic, 2008). Your characters are ambushed while standing in the middle of a circular elevator. The enemies are in the circular hall surrounding the elevator shaft. In the center of the elevator is a round wheel. Get to this wheel and hammer the action button to raise the elevator above the heads of the enemies. Now you have the high ground. The enemies, however, have access to a crank handle all their own, and they can pump it to lower your high ground. Now we have to keep our eye on the enemies' handle to pick off anyone trying to kill our advantage while also dealing with enemies around the circle. While dealing with those other enemies, one of them out of your sight range might get to the enemy crank and start lowering your elevator. It's a fast, maddening, excellent level design.

GOD HAND plays with the idea of opening a door with a QTE even while stopped at a conflict-free dead end. It does this by turning the event into a mini-game. Locked doors in GOD HAND often feature a large, smiley-faced button with a wide mustache made of maces. As you pummel the button, it turns from green to red. You can see its facial expression quaking. The face soon gets angry. This is your cue to tweak the right analog stick to dodge: its mace-mustache is about to clap your ears, doing big damage. Dodging forward or backward offers a slimmer margin for error than dodging left or right, though it is also quicker, and buys you more time to pummel the door-button. (Any time you're not pummeling the button, its color slowly fades from red to green.) Though the level design may be no more than hollow boxes full of enemies, the fights are fantastic, and even the simple act of opening a door includes unforgiving mini-games involving punching. It's no wonder scientists recently proclaimed GOD HAND the Best Game Ever. (Editor's note: they didn't actually.)

A pattern that emerges in the analysis of game-enhancing, progressive QTEs is that they involve using buttons on the controller for the same purpose that they're used in regular play. In the GOD HAND example above, the player uses the punch button to punch and the dodge button (actually an analog-stick swipe) to dodge.

Another excellent example is the first boss in NINJA BLADE, a game that otherwise features bland (if forgiving) cutscene-replacement QTEs. The first boss is a massive spider monster at the end of a corridor. The player must traverse that corridor, dodging left and right to avoid the shock waves the boss is spitting. If a shock wave hits the player, it hurts him, and knocks him back. The corridor is long and treacherous. When the player successfully reaches the end of the corridor, he can now attack the spider's face. He does this by pounding the attack button. Eventually, the boss doesn't like this, so he emits a super-powered shock wave that knocks the hero back with intensity. The camera zooms in to the hero's face. He's holding up his sword-edge against the shock wave. This is your cue to press the sword button rapidly to fight back the shock wave. No matter how many times you press the sword button, you're not going to conquer the shock wave. It's going to knock you back. The question is how much it's going to knock you back. With a less-than-stellar button mashing performance, you might be all the way back at the beginning of the corridor. With a great performance, you might only be 10 feet away.

JOHN WOO'S STRANGLEHOLD (Midway, 2007), likewise, exclusively employs such progressive QTEs. The most striking of them are the standoff situations. A group of enemies surround the hero. They point guns at him. They tell him to negotiate. He's played by Chow Yun Fat and wearing sunglasses at night, so he is definitely not going to negotiate. The camera slides into a first-person view. Time slips into super slow motion. Using the right analog stick, we perform the usual right-analog-stick motion of aiming the hero's guns. We pull the right trigger, and it does what the right trigger always does: we fire our guns. In the first slow-motion microsecond, the enemies begin to fire their guns. The first-person camera snaps from attacker to attacker. The crosshair is always a bit off of the deadly pressure point. You move it manually, at just slower than its usual speed, as you savor the super slow motion reaction time of the enemy in front of you. You pull the trigger. The camera follows the bullet impact. The enemy flinches, deforms, crumples, or explodes backward with terrific physics calculated by the impact point of the bullet. This is as exciting as QTEs can possibly get: the action fits story context, character context, and game control context, and the payoff is visceral and instant. Much as games like GEARS OF WAR and HALF-LIFE phased out the cutscene by making the narrative "happen" in the world as the player plays, STRANGLEHOLD shows that QTEs can be part of a game and not be sudden, intrusive, demanding situations.

In STRANGLEHOLD, physics is the payoff. Everywhere you go, you're shooting neon signs and watching them fall onto enemies. Objects that can be shot glint at appropriate times. Shoot them, and they're bound to fall on an enemy position. The "glint" is the game's way of temporarily, instantaneously gifting the player with the hero character's superhuman skill of destruction-minded creative perception. Shoot at the glint, and something will happen. Shooting the glint is accomplished by aiming and shooting your hero's gun in the same way as you'd aim and shoot the gun in any other context.

For the moment, let's ignore the way STRANGLEHOLD jumps the shark one-sixteenth of the way through stage two, and say that it might just be the future of action games. STRANGLEHOLD presents a genre where the game world itself is a QTE.

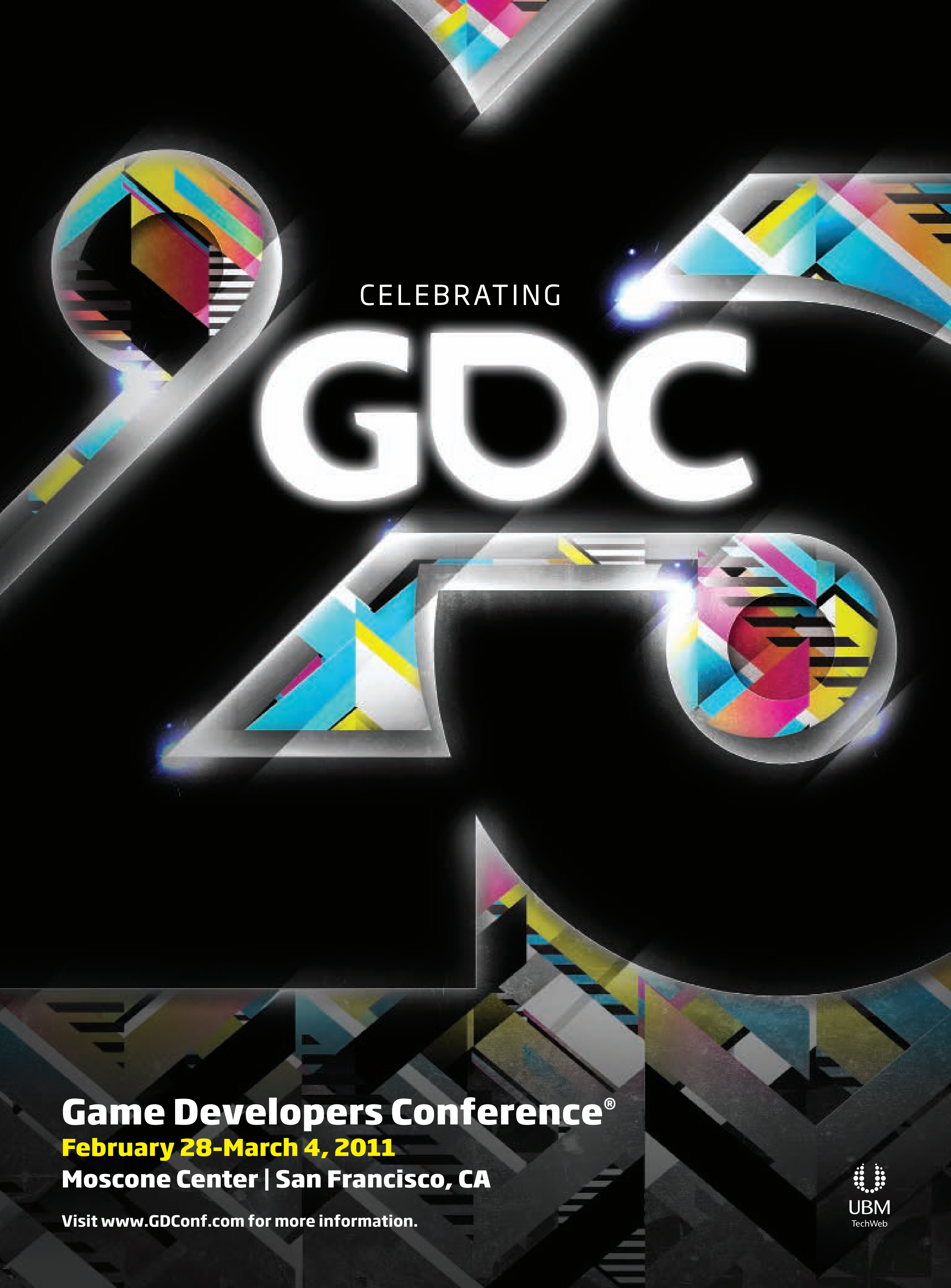
PRESS A TO DIE

So we've come full circle. Game graphics today are incredibly impressive, even if the things we do with them are something obtrusive and weird. Eight-year-olds who gawked at DRAGON'S LAIR in 1983, if shown STRANGLEHOLD, would likely scream until they spontaneously combusted. The amazing thing, way back then, was that games could look this good while simultaneously portraying complex, dynamic, cinematic action on the screen. We've evolved much since then. We've learned how to make graphics equally as impressive as those cartoons of the 1980s, and we've learned how to make games so incredibly interesting to play that we're willing to get online and play them with profane 12-year-olds, if we have to.

Consider ROAD BLASTER (Data East, 1985), which depicts high-velocity cartoon car chases from a driver's seat view. In ROAD BLASTER, your only input is pressing right or left on the controller at excruciatingly specific times. Your reward for enough precise inputs is to watch an enemy car fly off the road, smack into a mountain wall, and explode in a ball of fire—or to watch your own car drive up a ramp and fly over some impossible ravine. A decade and a half later, we have BURNOUT 3 and BURNOUT PARADISE (EA, 2004 and 2008), games about driving at criminally insane speeds and performing ridiculous maneuvers, where the central play mechanic involves knocking cars off the road to their death. What we've done in this modern age is perfectly recreate the thrill of piloting a speeding automobile, and married it seamlessly with the crazy action of sideswiping some dude off the road and into a mountainside.

Unlike ROAD BLASTER, BURNOUT, using only its vehicle native controls and no on-screen button icons, lets us finely control the velocity of our car and minutely consider the angle and ferocity of our approach. And when we succeed in our favorite in-game action of death-delivery, all kicks into slow motion and the camera swivels to bring the road behind us into view to show our soon-to-be-late rival slowly careening toward some form of demise, the physics of his flight perfectly calculated uniquely, just for our current performance. Compare that to the game-length QTE that is ROAD BLASTER. (Please ignore ROAD BLASTER's killer soundtrack and wicked-sweet character designs.) Which one is more exhilarating to play? Be honest. If QTEs are a "problem," we might be millimeters away from a global solution.

TIM ROGERS *is a freelance writer and game designer. He recently worked on* NO MORE HEROES 2.



Game Developers Conference®

February 28-March 4, 2011

Moscone Center | San Francisco, CA

Visit www.GDConf.com for more information.

UBM
TechWeb

electric ey pattern recognitio reality

CÉSAR BOTANA

////////////////////// IN SIMPLE TERMS, AUGMENTED REALITY IS THE PROJECTION OF AN INTERACTIVE GRAPHIC OVER A VIDEO FEED OF A REAL-WORLD PHYSICAL ENVIRONMENT. RECENT TITLES SUCH AS EYEPET AND EYE OF JUDGEMENT FOR THE PLAYSTATION 3, OR INVIZIMALS FOR THE PSP ARE EXAMPLES OF GAMES THAT USE AUGMENTED REALITY TECHNIQUES AS THE BASIS FOR PLAY.

An important key to these games is the process of pattern recognition. Using a camera, a pattern printed on a card can be recognized by the game, and on the card some play elements can be superimposed, such as the pet from EYEPET. Cards can be used to place virtual characters, or used to define a playfield in which virtual elements from a game are kept oriented to the real world surface they are projected over.

Here we'll examine how to recognize a pattern on a card, and how to detect its position and orientation. To make this task easier, we'll make use of the OpenCV library. This library has many functions for solving problems related to computer vision, but we will only use a small portion for our interests. OpenCV is written in C and includes the source code. Because OpenCV is BSD licensed, it is free in principle to port to almost any platform, as well as for use in commercial projects.

CREATING THE PATTERN

As shown in Figure 1, the basic pattern card which we want the computer to recognize is a simple image on a white background surrounded by a black border. The black border and white background is common to all pattern cards, and the image in the center will be what differentiates one from another. Some considerations to take into account when creating a pattern card are:

» It must be a white square with a black border (the width must be equal to the height).
» The image in the center needs to be grayscale, and it's preferable to use only the absolute white and black. It's possible to use shades of gray that are extremely close to these values, but avoid middle tones, or use them in a very specific way (such as at the edges of the image) to avoid aliasing.
» The interior image should not be too close to the edges. If the pattern card is moved, the image may blur, making it impossible for the computer to differentiate between the interior drawing and the pattern card's black border.

Once we've created a pattern card, we need to define the pattern associated with it, so it can be recognized by the computer. Begin by dividing the space within the black borders into *NxN* cells so that each cell can have its average color calculated.

To calculate the average color, add up the color values of all pixels inside a given cell, then divide by the number of pixels within that cell. If the value is less than 127, we will call the cell black. An average equal to or higher than 127 will be considered white. As shown in Figure 2, at the end of the process, we will have a pixilated image, in pure black and white, of the original image.

The maximum number of patterns that can be defined, and the ease with which we will be able to detect them depends on the choice of *N*. If we choose a lower resolution *N* such as eight, which results in only sixty-four squares, we will have little variety when it comes to defining, patterns and different images can look similar when pixilated. However, there is an advantage in averaging

ue

n in augmented

a large number of pixels per cell, as individual pixels will have very little influence on the total—two erroneous pixels are nothing compared to one hundred valid ones.

On the contrary, if we choose a higher resolution *N*—let's say eighty—we have a greater number of cells with which to generate different patterns. The disadvantage is that a single pixel now counts very much towards the calculation of the value of a cell—two erroneous pixels out of eight can cause the average value of the cell to be miscalculated.

We will have to repeat this process with the original image rotated 90, 180, and 270 degrees. Each of these will be called a rotations pose. Because a pattern card can face any direction in the real world, we need to know which of the four poses it's in if we want to calculate its orientation correctly.

DETECTING THE PATTERN

For every image we want to analyze through the camera, we'll have to follow the same operations:

First, we convert the image to absolute black and white. To do this, simply call the function **OpenCV cvCvtColor (OriginalImage, GrayImage, CV_BGR2GRAY)** to convert the image to grayscale. Once converted to grayscale, we call the function **cvThreshold (GrayImage, BinaryImage, 127, 255, CV_THRESH_BINARY)** to convert it to a binary image. These parameters indicate that if the color of the pixel is greater than 127 (remember that we are working on the grayscale image, which is over eight bits), the pixel will be considered white (255); otherwise, it will be considered black (0).

Keep in mind that environmental lighting can modify the view that the camera has of the pattern card. If the card is in shade, white can be taken as black, or if a light shines directly on the pattern, the brightness on a black area may be understood as white. Because of this, it is recommended to work with patterns that are absolutely black and white in order to minimize this problem.

Next, we calculate the contours of the image. A contour is only a list of points that indicate the border between black and white areas. To do this we call the function **cvFindContours ()** and indicate through the flag **CV_RETR_TREE** that we want the contours to be listed hierarchically. If a contour is inside another, it will appear as a child on the list. This is useful because our original image's black border and its internal white area creates another four-sided contour which is a child of the previous one. Use **cvApproxPoly ()** to convert a "rough" contour into a straight line, "poly" contour (see Figure 3).

The points defining a contour can be ordered in the list either clockwise or counter-clockwise. This will depend on whether the contour marks the boundary between a white area and a black one (clockwise) or between a black area and a white one (counter-clockwise). As shown in Figure 4, there are two contours highlighted in red. The arrow indicates the direction in which points are defined.

Once we have the list of contours, we can exclude those that do not fit the contour of the pattern we are seeking to recognize. A simple test is to check that the contour has four sides and the points that define it are clockwise. Also it should have a child (remember that you obtained a hierarchal list of contours) with four sides, and the points of its child should be counter-clockwise. Our pattern will meet these conditions because the outer black border will create a four-sided clockwise contour and its inner white square will generate a four-sided counter-clockwise contour. Additionally, we can check whether the adjacent sides of the contour create an angle of approximately 90 degrees.

At this point, we will have a list of contours that may fit the pattern we're looking for. For each possible contour, we will perform a series of actions to finally recognize it.

First, from the points that make up the contour, we build another list in which the points appear, ordered so that they are always clockwise, and the first point on the list corresponds to the top left corner of the image. This is necessary because the contour that **OpenCV** returns may be clockwise or counter-clockwise, and the first point on the contour is dependent on its orientation. When we recreate the list of points that compose the contour, we should always know how it is defined, as this makes working with them easier. For example, Figure 4 represents the position of the contours before reordering—the blue dots refer to the first point on the list.

Next, check the captured image to determine whether the area within the coordinates of the contour matches the pattern we are trying to recognize. To do this, we will represent this area using a homography matrix. In computer vision, homography is the process of projecting the mapping of a plane to a different plane. For example, the flat 2D view that the camera obtains of a real-world 3D object is an example of plane homography. We can calculate a matrix that relates these two planes by taking the image from the camera's plane of vision and converting it to one of our pattern's four rotation poses (see Figure 5).

To calculate the homography matrix, we begin by creating two lists of points. In the first list we will introduce the eight vertices of the contour we are working with (the four from the black edge and the four from the inner white square). This defines our origin plane. The second list will have the points of the pattern at the destination plane we seek—that is, the actual positions and dimensions in pixels of the black border and the inner white square from the original image pattern.

It is necessary to enter a coordinate Z in the second list, and we should always set it to 0 (Figure 6 shows the eight vertex values). Call the function **cvFindHomography ()** and as parameters, give it the two lists of points we've created, and a matrix in which to store the result.

Once the matrix has been calculated, we apply it to the binary image captured by the camera (see the results of the operation in Figure 5). On the image generated from homography, we should do the same calculations we did when creating the pattern. We divide the inner white square into *NxN* cells and for each cell we calculate its absolute color (black or white). Once we have the calculated values, we simply compare them against the black cells of the four poses of the pattern that we have loaded into memory. We only need to compare against black cells because the white cells define the background of the pattern.

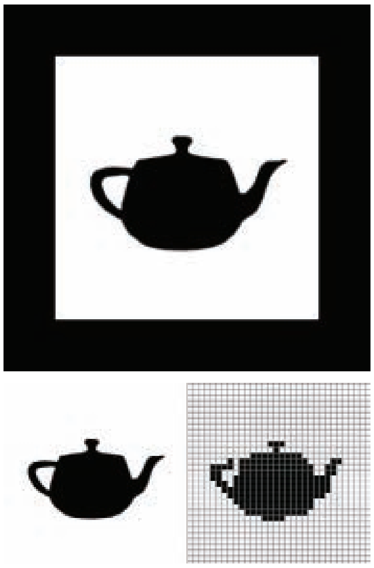


FIGURE 1 (top) shows the format for a basic pattern card. FIGURE 2 (bottom left) shows the original image and its binary representation (bottom right).

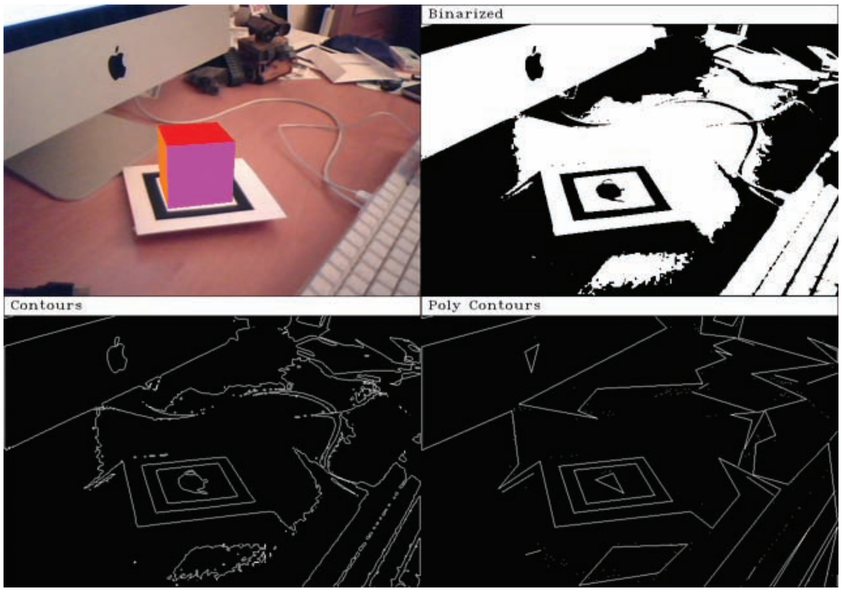


FIGURE 3 shows the detection process starting with the camera image (top left), the binary image (top right), the rough contour (bottom left), and its converted "poly" contour (bottom right).

Once the black cells are compared, we will get a percentage of similarity (number of coincidences / number of black cells). Then, we check to see if this percentage is greater than a certain threshold that we have defined (such as 90 percent) and if so, the pattern has been recognized. We always need to compare against a threshold, because in practice we are never going to obtain a 100 percent correspondence. When calculating the homography, the results will depend on variables such as size, orientation, lighting, and the image on the pattern card.

We can also employ a heuristic to decide which pose of the pattern to compare first. We can start by checking against the pose the pattern had the last time it was recognized, and if the pattern is not found to be using that pose currently, it can be compared against the preceding and following poses.

At this point, we will have detected the pattern in the frame of the camera. In addition, we have its base pose that will be needed to properly calculate its orientation. We will only need to get the positions and orientation of the pattern in 3D coordinates.

CAMERA EYE

Rays of light colliding with the objects around us form our view of the world. In such collisions, some light is absorbed and some is reflected. This reflected light is collected by our retina and defines the color that we see reflected from an object.

A camera functions very similar to the eye—light enters through the lens of the camera and is projected onto a plane that collects the image. To project a 3D point from the world onto the 2D plane of a camera, two matrixes are used to transform the origin coordinates.

The first matrix is the camera matrix, or the intrinsic parameters matrix. In it, the values for the focal length are defined—meaning the distance from the point where the light enters the camera, and the plane where the image is recorded.

There is a focal length for *x* and another for *y*. This is normally because the pixels of the camera plane where the image is recorded are rectangular rather than square. The other two values that are used are the displacement of *x* and *y*, from the center of the projection plane to the center of the camera. Under perfect conditions both would be 0, but low-cost cameras rarely have the millimetric accuracy needed to correctly place the projection plane right in the center of the focal point. These values are given in pixels, so if we calculate the values for a resolution of 640x480, and we change the camera resolution to 1280x960, we need to multiply the values by 2 for them to be valid.

The second matrix is a matrix of extrinsic parameters that are the product of a rotation matrix and a translation matrix. These indicate the rotation and translation from the coordinate origin of an object in the real world toward the coordinate origin of the camera.

Initially, each camera has some intrinsic parameters that differ from model to model. To obtain these values, it is necessary to

sample applications

To get more insight into pattern recognition, we've provided sample programs available at www.gdmag.com/resources/code.htm that use OpenCV for computer vision and OpenGL for graphic representation. They are compiled using Visual Studio 2005 Express for use with Windows. The window initialization and the event management of the application are set for Windows, so you will need to make some changes in this regard if you want to port to other platforms.The first sample is a command line program called Creator. The first parameter introduced should be the name of the file with the image pattern, and the second parameter should be the name of the file where the pattern information will be stored once it's processed. Visual feedback in Creator shows how the program processes the image so you can see the original image, its contours, and the four poses of the image converted to cells.

The other sample program is called Recognizer. When running, it will display a window that shows what the camera is capturing. By using the example pattern that comes with the program, Recognizer will place a virtual cube over the pattern that adjusts to position and rotation.

In Recognizer's debug window, you can see the steps that the application follows in order to detect a pattern. Press F1 with the focus on the recognition window, and you can see how the binary image is calculated, followed by the contours, the homography of the possible contour, and finally the pattern in the form of cells. In addition, at the bottom, a percentage indicates the degree of similarity found between the pattern taken from the frame of the camera and the pattern read from disk.

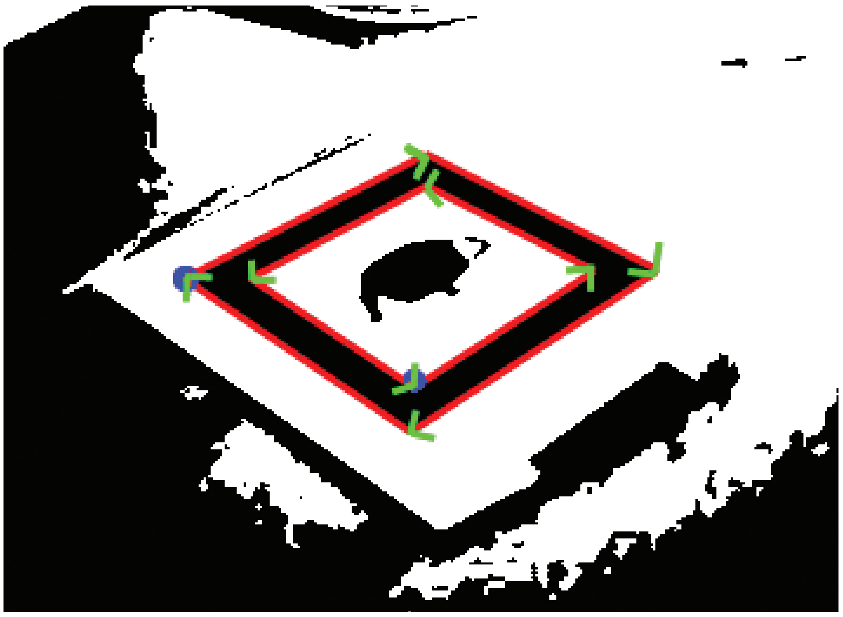
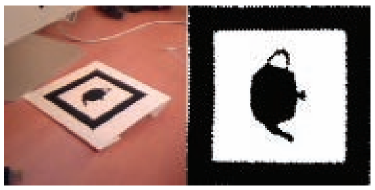


FIGURE 4 shows the two contours highlighted in red. The arrows indicate the direction in which points are defined.



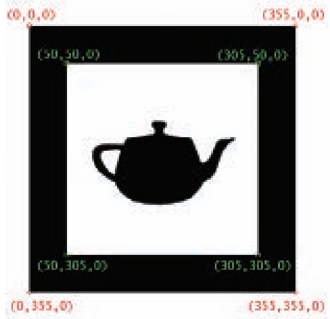


FIGURE 5 (top) To calculate a matrix that relates these two images, take the image from the camera's plane of vision and convert it to one of the pattern's four rotation poses. FIGURE 6 (bottom) shows the eight vertex values.

calibrate the camera. Using a well-defined pattern such as a checkerboard, images of the pattern in different positions and rotations are taken by the camera. Then by using the `cvCalibrateCamera2 ()` function, we obtain the intrinsic matrix.

CALCULATE POSITION AND ORIENTATION IN THE PATTERN'S 3D COORDINATES

In order to overlay 3D graphics onto the camera feed, we need to know the position and orientation of the pattern relative to the camera. This allows us to, for example, draw a 3D model on top of the card and have it look like it fits into the scene.

The first thing to do is to build two lists of points in the same way we did when calculating the homography matrix. The values of the points are the same as in the previous case; the only difference is the order in which we define the points of the pattern in the frame. When we calculated the homography matrix, the first point was always located at the top left of the contour, and the others were clockwise.

The first point is given by the pose in which the pattern has been found. If the pose is 0 (original image without rotation), the order of the points is the same as when we calculated the homography matrix. If the pose is 1 (image rotated 90 degrees), the first point on the list will be the last point of pose 0. If the pose is 2 (image rotated 180 degrees), the first point will be the last point of pose 1. For pose 3, which is the image rotated 270 degrees, the first point will be last point of pose 2. For the second list, we use the list of points that we built when calculating the homography matrix.

Once we have the two lists of points and a matrix of intrinsic values, we call the function `cvFindExtrinsic ()` and pass these values to it. It will return rotation and translation vectors for the object we have been analyzing that relates to the coordinate's origin (the camera's focal point). The units in which these values are given are the same ones we used to indicate the coordinates in the second list of points.

Remember that we introduced the values in pixels, so the values we obtain in return will also be expressed in pixels. In this way we can get the distance in depth between the camera and the pattern, and if we divide by the size of the inner white square, it will give us the depth in units of the world where we would place a 3D model.

For now, we only have the depth of the 3D object. To calculate the remaining two coordinates, we pass the *x* and *y* position from the center of the pattern on-screen (simply add the values of the four vertices and divide by four) to 3D, and throw a ray according to the orientation of the camera to the depth we have calculated. With this operation we will have the three world coordinates.

The orientation is given in a 3x1 vector. The three values indicate an axis of rotation, and the magnitude of the vector represents the angle of rotation. To make these values easy to use, OpenCV provides a function to convert this rotation vector to a classic 3x3 rotation matrix. This function is `cvRodrigues2 (RotationVector, RotationMatrix)`. If we are working with OpenGL, we should reverse the rotation in `X` before calling the function, and after calling it, we will have to transpose the matrix.

These are the steps necessary to detect a pattern and obtain its position and orientation in the 3D world.

Throughout this article, I used less than a dozen OpenCV functions. If you want to continue working with augmented reality, it is worth investigating this fantastic library. For example, you could obtain the brightness of a camera frame, and depending on the lightness or darkness, apply a shade to your 3D models so that they integrate better with the real world. Another example could be tracking the user's head movement in order to rotate our models, as well as detecting hand movements and gestures. There is a world of possibilities within this library.

CÉSAR BOTANA *has been working as a game programmer since 2001 in companies such as Pyro Studios and Zinkia Entertainment. He is now lead programmer at Tequila Works.*

resources

OpenCV http://opencv.willowgarage.com/wiki

Learning OpenCV: Computer Vision with the OpenCV Library
by Gary Bradski and Adrian Kaehler, O'Reilly Media, 2008

Splinte

Conviction

BY PATRICK REDDING, ALEX PARIZEAU, AND MAXIME BELAND

er Cell

////////////////////// WITH THE 2002 RELEASE OF TOM CLANCY'S SPLINTER CELL, UBISOFT MONTREAL ANNOUNCED ITSELF AS A STUDIO, AND UNDER UBISOFT, PRESENTED ITS INDUSTRY CREDENTIALS AS A DEVELOPER-PUBLISHER OF SERIOUS AAA TITLES. TWO SEQUELS FOLLOWED IN RAPID SUCCESSION, WHICH SOLIDIFIED THE FRANCHISE AS A LEADER IN THE STEALTH-ACTION GENRE.

After the fourth game in the series—SPLINTER CELL: DOUBLE AGENT (2006)—a new thematic direction had been established. Grizzled intelligence operative Sam Fisher was on the run and at odds with his former employers in the wake of a devastating loss. This reset offered both promise and peril for the next production, which wound up becoming SPLINTER CELL: CONVICTION. The SPLINTER CELL team set out to redefine the core gameplay away from traditional light and shadow stealth mechanics, and toward disguise, crowd concealment, improvisation, and dynamic environmental action. When the results fell short of the publisher's standards for its flagship titles, it meant the binning of over two years worth of work and the restructuring of the team.

This was the situation that confronted the title's new producer and creative director at the start of 2008. Having previously worked together on TOM CLANCY'S RAINBOW SIX VEGAS, Alex Parizeau and Maxime Beland understood the challenges of reinvigorating an established brand with extensive history and fiercely passionate fans. They were also starting their involvement at a time when the clock had already been run down by two years, representing a body of work that would be largely invisible to an end consumer. This was on top of the quality expectations that would come with a second outing on the current generation of consoles.

At the request of the company's Paris HQ, Beland steered the game back to its roots, and at the same time recalibrated the working model of shadow stealth to give the player faster movement, more visceral

Splinter

action, and explicit, presentation-layer cues for preying on AI enemies. Parizeau drew talent from Ubisoft's entire operation to grow the team, which at its peak numbered 200 developers in Montreal, Bucharest, and Paris. Ramping up through an accelerated pre-production, they delivered a vertical slice in November 2008, and pitched a full game walkthrough that deviated from the shipped content by only a single scripted event.

Reaction to CONVICTION upon its release in April 2010 reflected both the players' hunger for variety in the modern action-adventure milieu, and the relatively hardcore tastes of many bloggers and reviewers. SPLINTER CELL: CONVICTION dropped as an Xbox 360/PC exclusive and was the best-selling console game in the U.S. for that month, according to NPD.

WHAT WENT RIGHT

1) CLEAR AND FOCUSED DIRECTION (ONCE THE VISION WAS SET). Beland established a standard language for describing the target experience, and that message was applied consistently both internally within Ubisoft and externally during the marketing of the game. Sam would move and behave like the proverbial "panther." This was the reference point in the character's animations, and for new game mechanics designed to empower the player with a credible approximation of Fisher's ruthless super-spy efficiency.

Clear player experience targets gave the team the confidence to make decisions and to tweak the game's systems in support of Sam's predatory aspect. The result was a player fantasy that lived as much in the depth of CONVICTION's systems as it did in the fiction surrounding Sam's work for Third Echelon.

But that fiction also did its part to shift the game's emotional center out of the geopolitical spaghetti of most CLANCY titles, by focusing on the apparent death of Sam's daughter in a drunk-driving incident. For a character that had previously been defined as a self-aware chess piece, this made his concerns and anxieties immediately more relatable.

The decision to ship CONVICTION as an Xbox 360-exclusive title brought increased support from Microsoft. This working relationship proved vital when submission schedules were shifted around late in production to allow for more polish time. Having been briefed on the overarching intentions for the title, Microsoft was in a better position to offer flexibility. By the time the game was revealed on the Microsoft stage at E3 2009, content for the single-player campaign had been precisely scoped. The demo itself showcased real systems and data. What was shown was what shipped.

2) THE COOPERATIVE MODE STOOD ON ITS OWN MERITS. A late addition to the game's official planning, co-op became the focus of a dedicated team that was quickly assembled and given a mandate to deliver within less than a year. The result was several game modes, a two-player story, and new characters that fit the SPLINTER CELL universe (providing some of the classic mission-based structure loved by fans of the earlier games) while setting up background for the solo campaign.

Because key time constraints and the needed autonomy were established up front, the co-op group could pick its battles judiciously. Associate producer Sebastien Ebacher brought on Patrick Redding as game director, and the two of them collaborated closely to set quality benchmarks and dial the scope of content up or down as needed.

The designers focused on the core mechanics of single-player that best scaled up to two players, and they avoided overextending the design. Since they weren't locked into using existing single-player map geometry or assets, designers were able to tailor the level design blueprint to the unique needs of two-person multiplayer. The artists had the freedom to develop a unique visual aesthetic for the campaign, and were given room in the production schedule to iterate on level art and assets.

The resulting work extended the overall replay value of the game, and drew substantial praise from critics and players. Ubisoft leveraged this through a DLC strategy that delivered new DENIABLE OPS maps for months after the game's initial release.

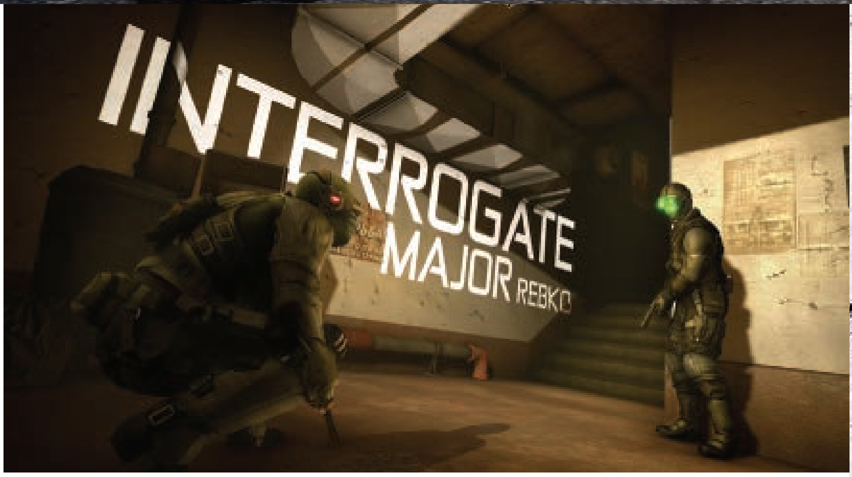
3) THE TEAM INCLUDED THE RIGHT MIX OF EXPERIENCE. At the time of the game's shift

PUBLISHER
Ubisoft
DEVELOPER
Ubisoft Montreal
NUMBER OF DEVELOPERS
200 at peak
RELEASE DATES
April 13th (Worldwide Xbox 360), April 27th (Worldwide PC), 2010
PLATFORM
Xbox 360, PC

in direction, there were developers assigned to the project who had worked on all four of the previous SPLINTER CELLs and had already devoted over two years to CONVICTION. The new team retained a number of these veterans, particularly those with engine experience. Others arrived off lengthy productions such as ASSASSIN'S CREED, RAINBOW SIX VEGAS, and FAR CRY 2. As the team grew to full capacity, it absorbed a substantial share of new talent arriving at the Montreal Studio, particularly programmers, artists, level designers, and level scripters.

Of particular note, game design lead Steven Masters helped develop player character actions for RSV, and was part of the fight team for ASSASSIN'S CREED. There were key contributions from ASSASSIN's alumni in gameplay animation, giving Sam's "predatory" athletic moves, takedowns, and 3D navigation some Altair-style grace. CONVICTION's ad hoc co-op group was a major beneficiary of this mix: lead designer JP Cambiotti and level design director Jason Arsenault previously worked on RAINBOW SIX: VEGAS and RAINBOW SIX: VEGAS 2, where they built their expertise at creating environments and gameplay around two-man units.

Because the Montreal studio is "engine agnostic," the most experienced of the CONVICTION newcomers had already worked with the SPLINTER CELL tech and tools, or their recent forerunners. This helped to reduce the ramp-up time, especially if new developers could be placed near those who had been involved since the beginning. The management team worked especially hard to retain the knowledge of the original CONVICTION crew, shifting people as needed to reduce project fatigue and opening up new roles whenever possible.



4) THERE WAS A CLEAR PIPELINE IN PLACE FOR DELIVERING FEATURES AND CONTENT. The path from concept to completion was understood well, and understood early, before the team was fully ramped up. Once pre-production began, CONVICTION's direction did not change. We delivered a publishable vertical slice halfway through the new production schedule, which contained all the features that shipped in the final game, in part because the team had already spent a year of rapid prototyping and knew how to plan and what to include.

Content development was streamlined through a blueprint process that allowed the level designers to begin iterating on their maps even while the gameplay team was doing the same for mechanics. The approach was unabashedly top-down, producing a timeline of the game's timeline in layers of increasing granularity, from story beats through locations, down to individual blocks of action.

CONVICTION also inherited a philosophy of design documentation that originated with ASSASSIN'S CREED and was put to good use on RAINBOW SIX. Feature Sign-Offs (FSOs) did away with unread, bloated Word docs and required designers to rigorously define the game's features in terms of testable requirements. The FSOs continued to live well past Alpha, and became the basis for the game's final closing plans.

5) THE PRODUCTION SUCCEEDED IN INNOVATING ON IN-GAME PRESENTATION ELEMENTS. Graphics were a key pillar of SPLINTER CELL from the very start of the series. In the intervening years, graphical fidelity has become a commodity of AAA games, essentially the price of admission for any title that hopes to compete. This raised the bar for the CONVICTION team as we worked to push the production values on scripted events and other story-driven moments in the game.

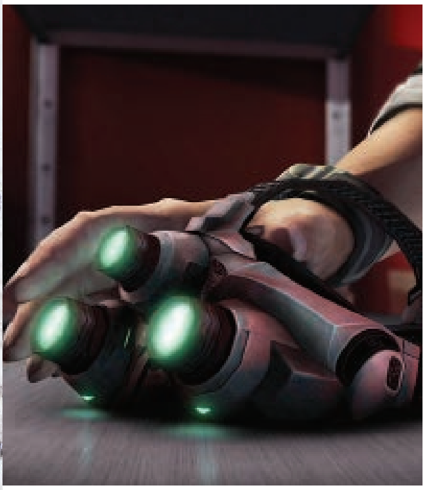
CONVICTION's visual signature is probably the use of projected media—movie clips and text splashed across in-game environments for the player to see at the right moment. Projections allow the game to show important dramatic beats without resorting to non-interactive cutscenes.

This technique also helped preserve the seamlessness of in-game continuity, further supported by delivering the game without cuts, hiding load-times behind continuous camera transitions that follow Sam from location to location, or offering a preview of the next chapter's objectives. This fed stylistically into the game's relentless forward momentum, reinforcing the theme of Sam's personal mission and the aggressive, strike-from-the-shadows dynamics.

WHAT WENT WRONG

1) THE GAME'S MECHANICS WERE RESTARTED FROM SCRATCH. CONVICTION's original direction, while intriguing as a concept, proved to be unworkable as a fully-featured game experience. When Parizeau and Beland joined the team in early 2008, the game as it existed then was so far removed from the core mechanics of the series that Ubisoft felt one of its key franchises was in potential jeopardy.

Splinter



Even though the "panther" concept was envisioned and documented relatively quickly, implementation could only be accelerated by so much. Many features that are a given in a SPLINTER CELL game, such as dynamic lighting, two-handed weapon firing, and gadget management literally needed to be recoded from scratch. Any new mechanics needed to sit on top of these must-have features, putting further pressure on the timetable and creating considerable bug risk.

For example, the black and white filter that tells the player they are hidden from nearby AI was functionally dependent on the restoration of the light and shadow system. It was impossible to make changes to ambient lighting or the nuances of shadow gradation without impacting the logic of the black and white filter, which made it harder to debug and prevented us from giving the effect the degree of visual polish we wanted.

Unfortunately there wasn't time to rebuild everything. Analog movement speed was abandoned in favor of a simplified run-walk system. The ability to pick up and hide dead bodies was never recreated, nor was lockpicking, nor the full variety of door-entry mechanics.

2) TARGETING ACCESSIBILITY PUT THE GAME IN CONTENTION WITH HARDCORE PLAYERS. Stealth games in general occupy a relatively narrow niche, and Ubisoft needed to expand the franchise's appeal or risk it falling between the cracks, pleasing no one. Earlier chapters in the SPLINTER CELL series had emphasized a fairly punishing model of stealth play that required painstaking observation and concealment, and which generally ended badly if the enemies' suspicions were ever aroused to the point of violence.

Opening the series up to new players who were wary of its reputation meant devising completely new mechanics and streamlining much of the complexity from the original systems. Unfortunately, some much-appreciated features—like the ability to move dead enemies, or use a knife for close-quarters kills—fell by the wayside because they were never included in the original design requirements, and there wasn't sufficient time left to reintegrate them and polish them to the standard of the earlier games.

Among other things, the compounding development pressures left the team without enough time to implement and polish a true "realistic-hardcore" difficulty setting that would have better satisfied the desire of some players to tackle the game in a more traditional way.

3) CHANGING DESIGN, TEAM, AND TECH AT THE SAME TIME CREATED SIGNIFICANT PRODUCTION RISK. As experienced as the team was, most of us were new to the franchise and needed a crash course in the engine and tools at a stage when our time was at a premium. This was further complicated by the fact that the technology itself was in flux, and engineers had to respond to the revised vision by integrating significant changes to 3D, rendering, animation systems, and AI.

Feature-specific teams were forced to make hard prioritization calls even before the project entered full production, in order to meet the twin goals of updating the mechanics and restoring key gameplay pillars. Rather than risk shipping legacy features in a half-baked form, the team chose to limit scope. The must-have Spies Vs Mercs adversarial mode became one of the first casualties in this process, which undoubtedly alienated core fans of the series.

Changing all three points on the "iron triangle" introduced continuous stress into the working lives of the entire team. The new direction still needed to be presented, explained, and evangelized to the team, and unsurprisingly, not everyone on CONVICTION understood the intention immediately. People were frequently blocked by a lack of stability in the engine and/or tools, since the responsible groups were scrambling to meet their new requirements within a compressed timeframe. And finally, the team wasn't afforded enough time to become acquainted with each other, and to adapt to their respective strengths and weaknesses.

4) THE ENTIRE LIGHTING SYSTEM NEEDED TO BE RECREATED FROM SQUARE ONE. If any one system represents a non-negotiable pillar of the SPLINTER CELL series, it has to be light and shadow, which is a critical affordance for stealth play. Without the shadows, Sam loses his most powerful weapon, and the experience begins to resemble a generic third-person shooter. When this was restored to the game's direction, there was no cheap fix.

CONVICTION's more kinetic play style also had an impact. Previous implementations of light and shadow worked well in small spaces and at slower movement speeds, but a heightened pace of action demanded larger spaces and more enemies to engage the player. Those two factors stressed the budget of dynamic light sources in many areas.

Furthermore, as runtime effects go, dynamic lighting is processing-intensive but not necessarily visually spectacular. In 2010, it's a stretch to position "light and shadow" as a quality differentiator. By eliminating this type of lighting, CONVICTION's original direction planned to rely heavily on ambient occlusion and static bounce illumination to create a more nuanced look for outdoor and daytime settings.

With shadow stealth back in play, level teams needed to light first and foremost for aesthetics, and then iterate extensively in response to playtests to get the setups working, both for the players and for the designers' gameplay intentions. Since the player has the ability to shoot out lights, ambient lighting was kept subtle, to avoid prohibitive runtime recalculations. Ultimately, the 3D group had to rework many renderer internals to support moving dynamic lights like flashlights, and enable large-scale, directional outdoor shadows. Because the game's stealth system determined the player's visibility based on actual illumination thresholds, small tweaks to the map lighting could easily break gameplay, and the group spent a frustrating amount of time creating debugging aids for artists and the AI programmers.

5) THE GAME'S EXISTENCE WAS REVEALED TOO EARLY. After the UbiDays 2007 reveal, the web was flooded with images of Sam in his tousled-haired "fugitive hobo" guise, a character design that was ultimately discarded in the context of the new direction. This created an automatic disconnect in the minds of players and journalists when the revised game was publicly announced.

During the major press and publicity events in support of CONVICTION, Ubisoft reps would routinely spend half of an interview explaining the checkered history of the project, time that might have been better showcasing key features. In the final analysis, the game might have been best served by giving it a completely new title to further distance it from its muddled origins.

SINGLE CELL ORCHESTRA

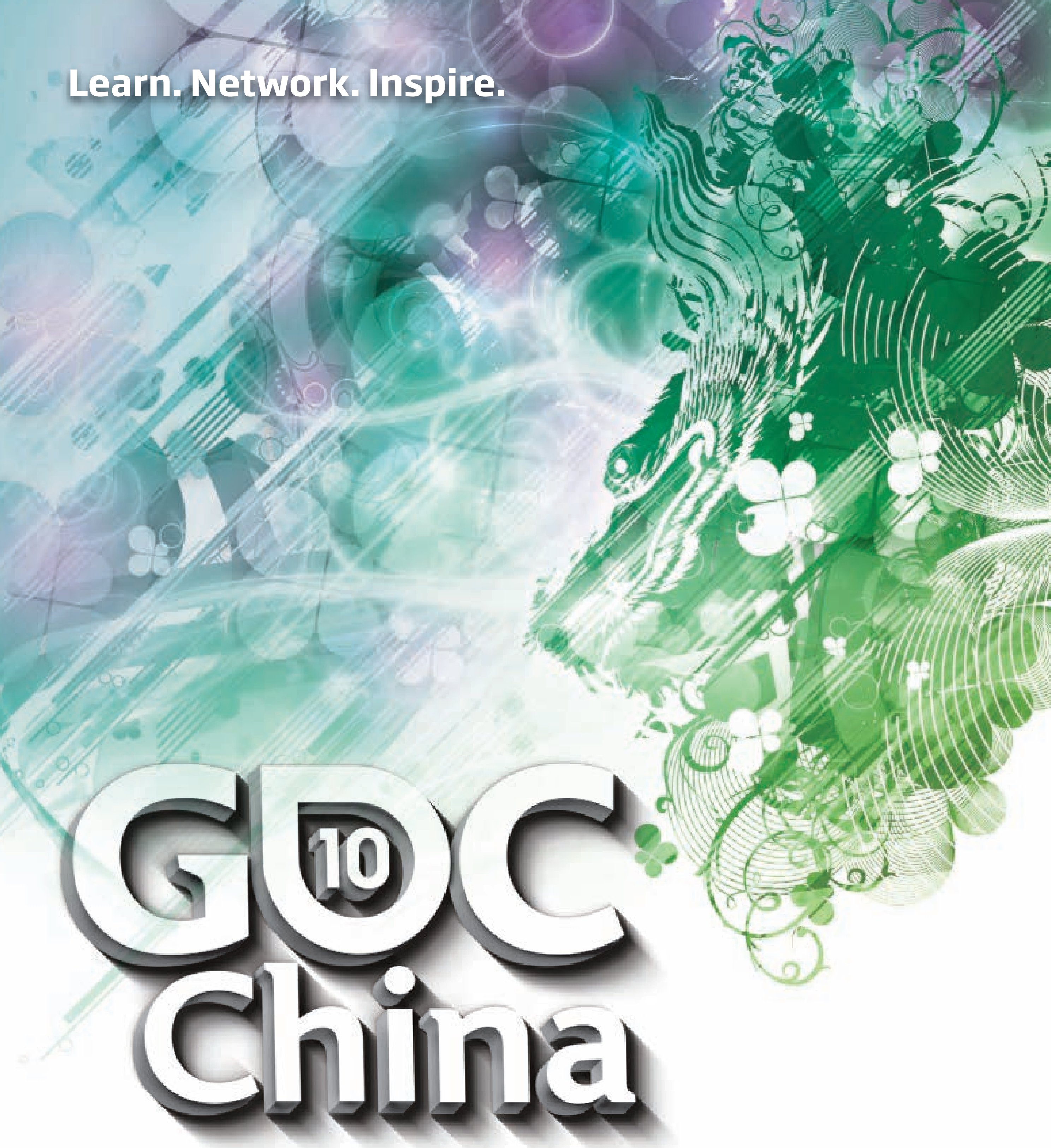
We, as a team, are extremely proud of what was accomplished in just two years.

In spite of the challenges, SPLINTER CELL: CONVICTION ultimately released with solid marketing buzz and consistently positive reviews. Post-launch survey data suggests that established SPLINTER CELL fans still made up the majority of players, and that their appreciation of the new mechanics was a lot better than had been originally expected. We realized that the franchise can be moved in a more accessible direction without losing longstanding fans of the series. With our approach vindicated, it now falls to us to take it further and reach out to new players.

More than anything else, CONVICTION represents a solid technical foundation on which to continue to build. The mistakes, missteps, and glitches have been accurately accounted for and can be resolved with very little risk. More importantly, we now have the opportunity to iterate on an already successful game by focusing on content, variety of play, and enhanced player choice.

The next game in the series is now under development at Ubisoft's new Toronto studio, with many of the same development leads. The direction and technology are in place, so the challenge for the moment is to build up a new team of experienced and passionate developers and continue to evolve the tools and methodologies that proved successful on CONVICTION.

By **PATRICK REDDING, ALEX PARIZEAU,** *and* **MAXIME BELAND.**



Game Developers Conference™ China

December 5-7, 2010

Shanghai International Convention Center | Shanghai, China

Visit **www.GDCChina.com** for more information

UBM
TechWeb

REVIEW BY TOM CARROLL

AUTODESK 3DS MAX 2011

WHILE 3DS MAX HAS SEEN revisions on an almost yearly basis, it's important to say right up front that 3ds Max 2011 is a significant update. So often software companies get into releasing new revisions when they're simply not worth the time or the money, like buying a new Lexus when really, only the trim and the taillights have changed.

This isn't the case with 3ds Max 2011. To continue with the automobile references (because I can), the release ships with a new user interface (UI) that is as intuitive as a cup holder, dynamic workflow streamlining, luxurious additions to the modeling tools, a new and improved system for building materials, integration of CAT (that's Max's Character Animation Toolkit, not videos of adorable felines playing with yarn), and even a best of class 3D/2D paint system.

And about that UI ...

Lest I begin the review sounding like the curmudgeon I am not, who provided guidance on that color scheme for the native viewports? Darth Vader? I can hear it now in the production meetings:

Minion: "Lord Vader ... what colors should the interface be?"
Vader: "Geev them any color as long as it's blaaack!"
Minion: "But Lord Vader ... black is ... well, black."
Vader: "I find your lack of confidence in dark color schemes ... disturbing. Make it charcoal gray then ... dark charcoal."

While I can anticipate the avalanche of emails wondering if my own 2011 viewports have been redecorated to conform to *Dora the Explorer* or *My Little Pony*, let me just counter that if the worst negative you can find about 3ds Max 2011 is the viewport color scheme, the rest of it must be pretty damned good. It is.

The first note is that it is possible to save assets back to the 2010 version of 3ds Max. This means that teams that might have been skittish about upgrading to a new level can now do so without any qualms (if they're using 2010, that is). And there are a lot of reasons teams would want to do so. For example, 3ds Max 2011's Object Paint tool and updated Viewport Canvas give game developers Toyota hybrid economy with Lamborghini styling. By themselves, these two features will help game artists save a ton of time, and that's no lie.

Let's see why ...

OBJECT PAINTING

>> I find that Object Painting is one of the best new features in 3ds Max 2011. It's tops because it allows you to do something you've always wanted to do, and it does it very, very well. And that is to paint parts of your scene with objects.

The Object Painting tab is on the Ribbon. To paint with objects, you pick a 3D object and paint into your scene with your mouse. The controls let you adjust scale and spacing on all three axes, and as objects are painted into the scene, they can be adjusted on the fly to prevent them from being too uniform. You can even paint with more than one object at a time. Painted objects remain live (you can continue to tweak them until your middle mouse button falls off) until you commit them within the scene. Artists will thank the 3D gods in the sky.

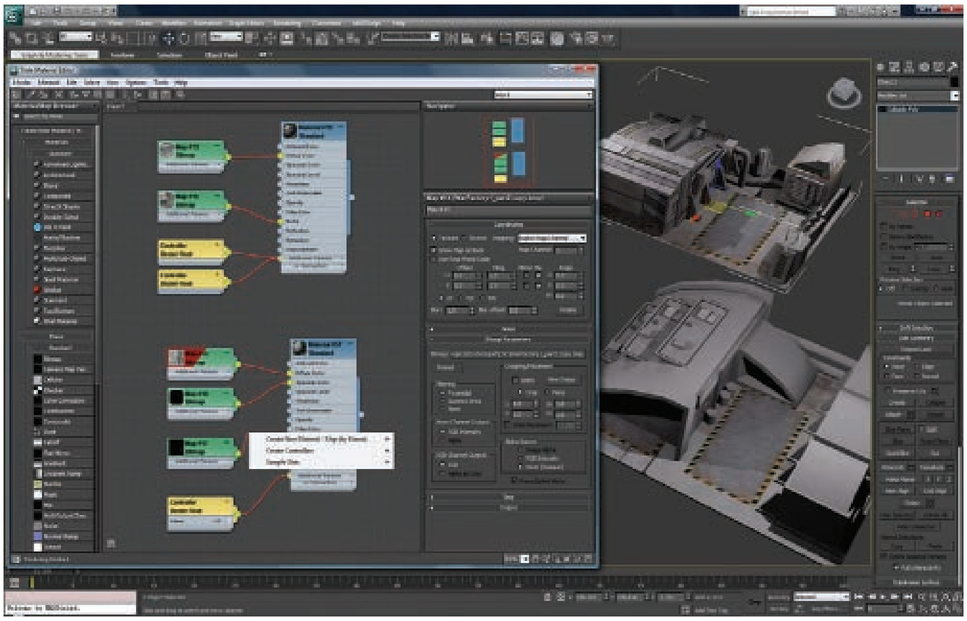
Common environment tasks such as placing foliage, patches of grass, and various sizes of rocks, bricks, tombstones, fence posts, bollards, etc., will not only become less repetitive, but may actually allow artists to get their art groove on.

Object Painting also allows for various kinds of object filling. For instance, let's say you're working on some kind of killer steampunk zeppelin and you want to put low poly rivets along the structural lines on the outside of the covering. In the past, you might have toiled away in relative obscurity, eating cold pizza and drinking Mountain Dew for a day and a night while placing those rivets by hand, or you might have taken a shortcut by painting them into a normal map, and then tried to make them appear in the render.

With Object Painting in 3ds Max 2011, simply select an edge or a loop on the zeppelin, turn the edge into an editable object, then pick a rivet object and use Object Fill to place rivets along the edge. The rivets adjust automatically, so the amount you preselected via the Ribbon will be spaced out nice and even.

VIEWPORT CANVAS

>> Viewpoint Canvas was a part of 3ds Max 2010, and it allowed you to paint on the diffuse channel of objects directly in the viewport.


The Slate Material Editor in 3ds Max 2011.

In 2011, it has been 'roided up a bit. Without question, the biggest change is being able to paint on any map type, including on bump maps directly in the viewport. The Canvas palette also includes options for loading and painting with bitmap images and masks, as well as with colors.

A new layer manager in Viewport Canvas mimics the layers you find in Photoshop, and what could be better than that? This layer manager lets you create new layers (and turn them on and off as needed), adjust the blending mode and opacity of each layer, and apply image adjustments such as brightness, contrast, levels, and color balance.

Viewport Canvas also has several new brushes to Clone, Fill, Gradient, Blur, Sharpen, Contrast, Dodge, Burn, and Smudge; also, the brush radius, opacity, spacing, and color can be randomized to add essential variety to your work.

CAT-TASTIC!!

>> 3ds Max's Character development tools have always been advanced and developer friendly. When Character Studio first appeared, it was "the bomb." Automatically creating walk cycles by placing footsteps quickly became the industry standard. But it's tough to stay on top, and the complex and convoluted nature of Character Studio became its Achilles' heel.

Now, 3ds Max 2011 has integrated CAT into the tool suite and it's a breath of fresh air.

This CAT is really simple and effective. Creating and animating rigs of various standard sizes and complexities is now super simple. If you're an animator, you now have the ability to change a rig to match another character, and you can do it even if the new character has a different number of legs, arms, tails, tentacles, you name it. CAT's wide variety of default rigs are ideal for a wide range of human characters, and also for an even wider range of malformed trolls, creepy monsters, or scary creatures.

CAT is also smart. You can use standard transform tools to position and orient bones, and CAT automatically knows what section of the character you are working on. For instance, if you create and position one arm, then create a second arm, CAT positions the second arm automatically as a mirrored copy of the first. CAT also correctly names all rig bones by default, and if you change the adult name, it correctly modifies the names for all children bones as well.

Part of CAT's brilliance is that it retains a lot of the best Max features so today's animators don't have to relearn the whole thing. I could go on and on, but animators should find a way to experience the new version, I'm sure they'll see the clear advantages.

AUTODESK 3DS MAX 2011

Autodesk, Inc., 111 McInnis Parkway, San Rafael, CA 94903
http://usa.autodesk.com

PRICE

- 3ds Max 2011: $3,495
- Upgrade from 2009 or 2010: $1,745

SYSTEM REQUIREMENTS

Microsoft Windows 7 Professional, Microsoft Windows Vista Business (SP2 or higher), or Microsoft Windows XP Professional (SP2 or higher). Intel Pentium 4 1.4 GHz or equivalent or AMD processor with SSE2 technology.
2 GB RAM, 2 GB swap space, 3 GB free hard drive space. Direct3D 10 technology, Direct3D 9, or OpenGL-capable graphics card.

PROS

1. Ability to save assets back to the 3ds Max 2010 level.
2. Object Painting is a real time saver.
3. Viewport Canvas toolset promotes the artist within 3D art.

CONS

1. 3ds Max 2011 seems a little slower than 2010 and before.
2. The charcoal color scheme. How fast can one adjust that?
3. Extrapolating on rev level naming sees 2012 hitting the market in about a month.

product news

New Version of NavPower

BABELFLUX
www.babelflux.com
/// Middleware developer BabelFlux announced a new version of its NavPower AI pathfinding software, which will be used in upcoming games such as EA's DEAD SPACE 2, DARKSPORE, and Square Enix's DUNGEON SIEGE III.

DUNGEON SIEGE III.

Pathfinding algorithms in the new version allow for AI-controlled characters to negotiate moving platforms, fly over buildings or through windows, and navigate crowded MMO environments, according to BabelFlux.

The update also optimizes the middleware's performance on the PlayStation 3, using the system's multiple cores more efficiently to reduce memory consumption and improve response time, the company says.

3D Flash Game Hardware Acceleration

ADOBE SYSTEMS
www.adobe.com
/// Adobe announced a new set of APIs enabling large-scale, hardware-based 3D acceleration on its popular Flash and AIR platforms. Code named "Molehill," the company's new APIs will be available as a public beta starting in the first half of 2011.

They will provide low-level programmable shader-based engine features including z-buffering, stencil color buffering, fragment and vertex shaders, and cube textures, all of which will use the GPU "where possible" for "significant performance gains."

Developers are told to expect "hundreds of thousands of z-buffered triangles to be rendered at HD resolution in full screen at around 60 Hz" under the new APIs, compared to "thousands" of un-z-buffered, 30Hz triangles under the current Flash Player 10.1.

The acceleration will rely on DirectX 9 standards on Windows, OpenGL ES 1.3 on Macs, and OpenGL ES 2.0 on mobile platforms, and potentially puts Flash more directly into competition with 3D-centric web game engines such as Unity.

Beast Lighting Middleware

AUTODESK
www.autodesk.com
/// The Autodesk Beast lighting middleware is now shipping and is the first release of the product under Autodesk since the company acquired Beast creator Illuminate Labs in July this year.

Autodesk Beast allows for light bouncing, color bleeding, soft shadows, and other lighting-related effects in games.

The company claims that Beast streamlines the lighting process in game development with its LiquidLight Global Illumination Engine and real-time lighting visualizer eRnsT. The tool's DistriBeast distribution engine can be used to quickly distribute renders on multiple machines.

Other key features of Autodesk Beast include precomputed global illumination, the ability to adjust overall scene lighting without bounce lights or ambient fills, and an API that allows users to more easily integrate Beast into a game engine.

SLATE MATERIAL EDITOR

» Next in the conga line of new features in 3ds Max 2011 is the Slate Material Editor. I place it second after Object Painting because of the increasingly complex nature of today's materials, even more so for video game work. Slate is an expansion of the Compact Material Editor (previously known simply as the Material Editor). While Slate incorporates portions of the old system, it is miles ahead because of its highly visual node-based material editor. The fact that you can see at a glance the numerous components comprising a material, the relationships between them, and each shader channel puts the old system to shame.

Another huge addition under Slate is the inclusion of the Autodesk Material Library within the Material/Map Browser. Thousands of ready-made materials can be immediately applied to objects, which is especially useful for environment artists that need to start working right away with various real-world architectural materials like tile, ground, metal, water, and glass.

INTERACTIVE TEXTURE DISPLAY

» In a tie for third with the Slate Material Editor is the ability in 3ds Max 2011 to view many 3ds Max texture maps and materials within the viewport to help develop and refine scenes in a higher-fidelity interactive display environment without the constant need to re-render. Tell me that you enjoy sitting at your desk at 11:59 PM, trying to dial in those texture UVs on your model but constantly struggling with the low-res nature of the display. 3ds Max 2011 gives you the poison dart to take that beast down. The more I talk about it, the more I want to bump this feature into second place, but I'll move on.

QUICKSILVER

» The Quicksilver hardware renderer is a new multithreaded rendering engine that uses both the central processing unit (CPU) and the graphics processing unit (GPU) to deliver rendering PDQ (Pretty Damned Quick). The new setup achieves up to 10 times faster rendering than traditional techniques on common graphics cards. You can't necessarily say that this is a benefit to video game developers because many cutscenes that used to be rendered are now done within the game with real-time processing. However, with such an unsteady job market, no one in the industry will begrudge being able to more quickly render out asset turnarounds, animation cycles, and full scenes to appear in their demo reels and websites. At the same time, in terms of overall performance, 3ds Max 2011 seems a little slower than 2010 and before.

TO UPGRADE OR NOT TO UPGRADE ...

» Overall, I've found that Object Painting and Viewport Canvas are worth the upgrade fees by themselves. Add in everything else, and while you may not quite be willing to relegate that legacy copy of Max 2010 to the scrap heap (like your old Pinto), it makes the decision not to upgrade a lot tougher than it usually is. But if you're working with characters, then the integration of CAT will deliver a beat down like Rey Mysterio driving through your studio in a monster truck.

And when have you ever seen that metaphor in a software review?

TOM CARROLL *is a freelance video game artist and a contributor to myIPD.com, an intellectual property portal.*

GAME CAREERGUIDE.COM

UBM TechWeb

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
- MESSAGE BOARDS

DIGITALCOUNSELOR

I'LL MATCH YOUR INTERESTS AND GOALS WITH THE RIGHT GAME RELATED PROGRAMS AND SCHOOLS FROM AROUND THE WORLD.

WWW.GAMECAREERGUIDE.COM

FIND OUT HOW THEY GOT THERE.
Download and Play the latest student Games!
GAME CAREER GUIDE
Need Help Choosing the Right School?
DIGITAL COUNSELOR
GETTING STARTED
FEATURES
Earn your bachelor's degree in three years.
DigiPen

DOWNLOAD YOUR FREE DIGITAL EDITION OF THE 2010 gamedeveloper GAME CAREER GUIDE ONLINE AT: WWW.GAMECAREERGUIDE.COM

THE BALKANS

LIVING AND WORKING IN A FRAGMENTED MEDIUM

NEARLY 10 YEARS INTO THE seventh console generation, we've gradually become accustomed to a lot of things that once seemed exotic and scary. There aren't a lot of we-don't-need-no-steenking-normal-maps holdouts anymore, and plenty of artists can toss off terms like "spherical harmonics" with the kind of casual nerdiness that once belonged only to the ensign Wesley Crushers of the multiverse. It's kind of shocking nowadays to roll back the clock and look at titles that represented the peak of graphics in 2002 or 2003; without the gauzy haze of fond gaming memories, many of those AAA stalwarts would look more at home on a handheld or a smartphone.

One of the peculiarities of this generation of tech is that the advances seem to come in fits and starts. One game has really swanky effects shaders, with water ripples or caustic lighting or buttery smooth subsurface scattering. Another has a brilliant lighting model, with a huge dynamic range and ultra-sophisticated radiosity mapping that captures all the subtleties of indirect lighting. A third may have real-time soft shadows or ambient occlusion helping to define its forms and tie its disparate pieces together. But no game really does it all (which is probably good news; would you want to compete against the game that really did everything?).

A COMPROMISING POSITION

>> Modern hardware can do remarkable things, but the hard limits of memory and processing power still force us into some awkward compromises. You might, for example, go with a deferred renderer that gives you lots of lights, but doesn't handle multiple layers of transparency well. Or, you could plug in a fancy offline radiosity solution and store tons of spherical harmonic light probes for a really rich indirect lighting solution, but you might not have enough memory left over for anti-aliasing at 1080p, or the ability to change the time of day dynamically.

Obviously, the tradeoffs that define any engine are (or let's say they should be) driven by the nature of the game you're trying to make. A moody film noir mystery game should obviously be investing more in dynamic shadowing than a flight sim, while a tunnel shooter can probably rely a lot more heavily on pre-baked lighting than an open-world game with weather effects.

Sometimes these choices are simply budgetary ones: will dynamic reflection maps leave enough memory in the budget for a fancy texture-based skin shader? Sometimes the choices are architectural: If you want to have a lot of dynamic lights, you're probably not going to get MSAA or cool transparency effects. One way or another, it's the game artist's eternal pride and perennial curse: you can't have it all.

BALKAN DEATH GRIP

>> The interesting side effect of this overstocked buffet of graphics techniques is balkanization: every project comes with its own unique way of doing things. Many artists who've finally settled down into a comfortable co-existence with the "next" generation of graphics don't even realize how little their particular brand of "next-gen" resembles those of other games and other platforms. Context is everything in art, and when the context is some crazy one-off proprietary tech, your art must struggle to adapt.

Consider how even something simple like environment texturing interacts with the complex weave of lighting and shaders that make up an engine. Perhaps I can rely on screen-space ambient occlusion: this means I get a lot of the visual definition in my environment for free, so I don't have to bother with a lot of extra tricks for darkening corners and emphasizing intersecting forms. You, on the other hand, spend most of your time finessing the way your materials bounce photons around to get precisely the subtle radiosity effect you need, because your fancy lighting makes mine look like crap. (But hey, at least I don't have to wait two hours for the light farm to process my changes whenever I move a light!) Meanwhile, the driving game team up the hall uses simple direct lights because they need dynamic shadows for night races, and they can't spare the memory for lots of ambient textures. So they kick it 1999 style, with a ton of baked-in lighting in their textures.

Now, imagine that management has reassigned us all. How long will it take us to be productive in each other's working environment? And that's just for texturing!

MIGRANT LABOR

» For the working artist, this is not just an annoyance. If you're too tied to the ins and outs of one particular engine, your career mobility can be at risk. When the rules of the game vary so much from studio to studio, your work can be devalued by viewers who don't understand its technical constraints. It's hardly surprising, either, that candidates who've worked with familiar tools and techniques have a leg up over those with very different experience, but it's not much fun to miss out on an interesting project because you know the "wrong" lighting model or software package.

Technical balkanization exaggerates the industry's tendency to value computer bullet points over the artistic side of art. How many of us have sailed through a portfolio check and a phone screen only to run smack into something like "of course, you know we're really looking for somebody who's done Mudbox sculpting" or "we loved your work but we need somebody who has more experience in HDR lighting."

Now, to be fair, this kind of job-search triage isn't just random cruelty. The art director or producer looking to fill a hole on a production team has a legitimate reason for preferring candidates who'll be ready to go from the day they show up instead of requiring weeks or months of ramp-up time. We all know that it's the artist, not the tools—but too few of us feel like waiting for said artist to actually learn the tools. It's the same thinking that has led a lot of us to half-consciously segregate ourselves into Max- or Maya-centric career tracks. It's the path of least resistance.

Unfortunately, the logic may be impeccable but it comes with some serious costs. For individuals, the costs can be pretty brutal: losing out on a job that you know you could ace solely because you haven't spent a few weeks with a particular set of sliders is pretty galling. Technical decisions that your studio make to service a particular project (or satisfy a publisher's whims) can have a serious but silent impact on your career.

UNIVERSAL STUDIOS?

» For the industry, the costs are more subtle, but also more far-reaching. Our heterogeneous ways of making things reinforce our business model, even when a lot of people think that model is in need of an overhaul. We've spent years trying to figure out whether we ought to organize in Hollywood fashion, combining small core teams with a cloud of freelancers and work-for-hire production houses rather than our long-term studio model (see Pixel Pusher's March 2010 column "Zombie Apocalypse"). Love it or hate it, the Hollywood model has sustained several generations of huge franchises successfully, while our business has struggled awkwardly to handle the huge content costs of modern graphics.

There are many reasons why games haven't gone Hollywood yet, from culture to unionization to geographical concentration. The fragmentation of our art pipelines and engine technologies also plays an important role. If you want to build a production team from scratch every time you spin up a project, standardization has obvious attractions. The perennial complaint that Hollywood "doesn't have to re-invent the camera for every movie" doesn't just explain our penchant for missing deadlines, it also explains why we stick with a studio model that looks more like the 1930s than the 21st century.

It doesn't take much effort to prove that the crazy welter of different techniques and tools we use is a hassle. Whether it has interfered with your career plans or simply forced you to waste time learning a succession of quirky approaches to common problems, you already know that life would be a lot easier if we didn't have such a bewildering variety of tools and techniques to keep up with.

WHAT STANDARDS?

» Unfortunately, standardization may be attractive but it's not going to happen any time soon. As the current console generation matures, we will probably see some progress toward common ways of getting things done.

In the early days of any tech, there's a lot of experimentation. The early days of normal mapping, for example, involved lots of custom tools and incompatibilities. Who can forget their first time trying every one of ZBrush's annoying tangent-space and UV-winding options to find the magical combination that would actually produce a working result?

Over time, though, the experimental processes tend to simplify and become more robust. Nowadays, it's pretty much a given that any working artist can figure out how to crank out a normal map without special instructions and a PhD in math. Other technologies, from motion capture to high-level shading languages, show a similar progression from the exotic to the merely mundane. Hopefully, in our gradually aging console generation, we'll see more tech becoming de-mystified.

Nonetheless, the main reason for our lack of standards isn't going away. That is to say, our ambitions are, and probably always will be, far grander than our hardware can support. We're constantly trying to strike a balance between the shifting, conflicting demands of gameplay, graphics technology, and art. Our need to make things cooler means we'll always be trying new ways to get more and better art out of those damn machines, and inevitably this will mean we strike out in different directions and make different, incompatible discoveries.

EXTRA-CURRICULAR ACTIVITY

» Because the career impacts of all this helter-skelter technological change can be pretty severe, it's important to have at least a basic familiarity with the way things work outside your particular studio setup and pipeline. *Game Developer* postmortems, Gamasutra, and particularly the GDC are great ways to take the pulse of how the production landscape is changing. And of course, in the Internet era, you can always find somebody out there who's willing to do a step-by-step tutorial on every new trick in graphics or content creation.

You can also help yourself by doing a little work on the side. Artists who freelance in their spare time pick up valuable intelligence on different studio cultures and techniques as well as a little extra cash. Even experimenting at home with techniques you don't get to use at the office is valuable. If nothing else, a few hours of learning the ropes on a new process is great for handling awkward interviews: "We don't use it on my team, but I've done a lot of work with it in my side projects" is often enough to nudge a wavering interviewer back onto more positive topics like your portfolio.

If this sounds like a lot of hassle ... well ... it is. But it's also a critical consideration for the long-term health of your career. Technologies, even companies, come and go. The only thing you can be sure of is the life of a modern games artist demands you be a jack of all trades as well as a master of one.

STEVE THEODORE *has been pushing pixels for more than a dozen years. His credits include* Mech Commander, Half-Life, Team Fortress, Counter-Strike, *and* Halo 3. *He's been a modeler, animator, and technical artist, as well as a frequent speaker at industry conferences. He's currently the technical art director at Seattle's Undead Labs.*

GAME CONFIGURATION AT CRYSTAL LAKE

LOADING AND PROCESSING CONFIGURATION DATA WITH JSON

Game configuration is a problem that is both ubiquitous and arbitrary. Every studio needs to configure some aspects of its games, and the data used to configure each aspect can come in various shapes and sizes. This can lead to customized solutions for loading and parsing different types of configuration data. The Excel JSON Exporter is meant to help simplify and unify how our configuration data is loaded and processed.

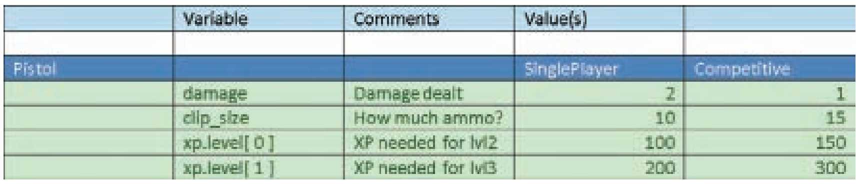
JavaScript Object Notation (JSON) is an open-standard text format for describing objects. It's basically text that defines a set of key-value pairs, where values can either be scalars, arrays of values, or other objects. Because it is an open standard, a wide range of tools are available for use with JSON. Our new toolset makes heavy use of JSON for asset metadata, game object properties, and so on, which made JSON a natural fit for storing our game configuration data. The JSON Exporter is an add-in for Excel that converts a specially-formatted spreadsheet into a JSON text file for use in our game.

A NEW BEGINNING

>> When determining the configuration data pipeline in our new tools and engine, one of the first decisions we needed to make was where the source data would come from. Would configuration data be stored in an internal format and edited using a proprietary tool, or would it be imported from some third-party application? In our case, the answer was straightforward enough—our current method for getting configuration data into the game was to use Excel as the source, convert it to CSV, and then parse it in the game (with a special binary version of the data used in final builds). Overall, designers like using Excel, and it's available on all our machines, so using Excel requires no additional installation process for us. While both writing a proprietary tool and using Excel have their pros and cons, a custom tool that is basically a spreadsheet (which is the user interface the designers want) would require more effort in the short term and take longer to mature than writing an add-in for Excel, which has a well-known interface with plenty of features.

Once we decided on Excel as the source of our configuration data, the next step was to determine how it would make it into the game. The use of JSON with the configuration data stems from our use of JSON in other aspects of the codebase. At Insomniac, we're currently creating a new toolset in Flash to provide our team with a richer user interface for development, and JSON provides a convenient method for sending data from Flash/ActionScript into our C/C++ code and back. To support this on the C/C++ side, we created several utilities (the main one is called Data Description Language, or DDL) that provide functionality to convert JSON both to and from an arbitrary C/C++ structure.

On the Flash/ActionScript side of the code, ActionScript objects can be converted both to and from JSON in JavaScript, and thus shared with the C/C++ code. It is this data path from ActionScript to C/C++ and back that forms one of the backbones of our new toolset. Because JSON serialization is widely supported in our C/C++ code via the DDL utility, we adopted its use for a variety of other tasks. For example, game object properties are specified in the level editor and parsed at runtime into our game object structure using JSON. Our level data structure is saved as JSON. We use JSON to store our asset metadata on disk and to store asset changes in our asset editor's undo queue. We use JSON to send generic events in our event system. We even use JSON from Perl scripts to help automate a few processes in the new toolset. Thus, it seemed only natural to use JSON as our method for getting configuration data from Excel into the game.



	Variable	Comments	Value(s)	
Pistol			SinglePlayer	Competitive
	damage	Damage dealt	2	1
	clip_size	How much ammo?	10	15
	xp.level[0]	XP needed for lvl2	100	150
	xp.level[1]	XP needed for lvl3	200	300

FIGURE 1 shows the configuration data for a Pistol weapon.

JSON LIVES

>> Writing an Excel add-in was not without its (minor) challenges. The first thing to address was the choice of language—a C/C++ plug-in or a Visual Basic add-in. Although I'm far more familiar with C/C++ and generally prefer it over Visual Basic, the process for getting a C/C++ plug-in into Excel is daunting, and involves writing some sort of plug-in shim. In the past, we wrote an Excel plug-in for localization (and its corresponding plug-in shim) using C/C++, but it had its share of problems. It was complicated to set up the environment, was difficult to debug (the debug build of the plug-in would not load on some systems), and every update required the users to re-install the plug-in. After having to deal with those issues, I have been wary of shimmying anything into anything else—it just sounds bad.

In contrast, writing the plug-in as a Visual Basic add-in made the setup as simple as saving the Excel file as an Excel Add-In (.xla/.xlam) file. Updating the plug-in only required that we replace the .xla/.xlam file (provided Excel is not running at the time of update). However, I still had to deal with the occasional cryptic Visual Basic error message, such as "Only User Defined Types Defined In Public Object Modules Can Be Coerced To Or From A Variant Or Passed To Late Bound Functions."

It's difficult to export an arbitrary Excel spreadsheet to some sort of structured and meaningful JSON, so to narrow the scope of this problem, we placed certain rules on how the Excel spreadsheet needs to be structured. The spreadsheet structure was created to suit the needs of our designers, and while it can change from game to game, it usually does not. Most of these rules already existed in the previous system (which used Excel and exported to CSV) and are things the designers are already familiar with. The basic rules are as follows:

- The name of the sheet will be the name of the top-level object.
- A main header row must be present to indicate the start of the configuration data. This header row must contain "Variable," "Comments," and "Value(s)" in the second, third, and fourth columns respectively.
- At least one object header row must be present, and is determined by having the name of the object in the first column. In addition, if a data member can have more than one value, the names of those values start at the fourth column.
- Each row after the header row will specify object data members and their values. If a data member is also a nested object, its members can be accessed using a C-like syntax.

An example set of configuration data for a Pistol weapon can be seen in Figure 1. The light blue row is the main header row. The main header row is determined by columns two, three, and four having the values "Variable," "Comments," and "Value(s)" respectively. The "Comments" column is ignored by our format, and designers can do whatever they want with it. The "Value(s)" column is used to indicate the starting column for all the values for a given row. Everything above the main header row is ignored by the parser; this gives designers the ability to put notes at the top of the file. The dark blue row is the object header row, which indicates that the current object is "Pistol," and that it will have values for both "SinglePlayer" and "Competitive." The object header row is determined by the name of the class being in the first column (and of course being after the main header). The green rows represent the data members of "Pistol," such as "Pistol.damage" and "Pistol.clip_size." Member variable rows are determined by having the member variable name in the second column (and nothing in the first column), which is placed after a valid object header. All other rows are ignored.

THE NEW BLOOD

>> In practice, the spreadsheet in Figure 1 may contain several sets of rows for additional weapons such as the Shotgun, Rocket Launcher, and so on. Each weapon may have a different set of member variables as well; there are no limitations on what objects can have what data members. Note that cell color and other cell formatting have no effect on how the cells are interpreted—they are formatted such that the spreadsheet is easier for designers to read. The meaning of a cell is derived from its row number, column number, and the most recent object header.

The Excel JSON Exporter converts this specifically-formatted spreadsheet to JSON. It works by processing each row one at a time and then converting it into a string containing a C-like syntax that describes the member variable to be set. This string is then used to add a node to a tree that describes the final JSON data. In particular, when the exporter encounters a new object header row, it remembers the object name. In Figure 1, the object name is "Pistol." For each row beyond that until the next object header row, it generates the C-like string by concatenating the sheet name, each value type, the object name, and the member name. For the Pistol example, if the Excel worksheet were named "Weapons," the C-like string generated for each row would look like this:

```
Weapons.SinglePlayer.Pistol.damage
Weapons.Competitive.Pistol.damage
Weapons.SinglePlayer.Pistol.clip_size
Weapons.Competitive.Pistol.clip_size
Weapons.SinglePlayer.Pistol.xp.level[ 0 ]
Weapons.Competitive.Pistol.xp.level[ 0 ]
Weapons.SinglePlayer.Pistol.xp.level[ 1 ]
Weapons.Competitive.Pistol.xp.level[ 1 ]
```

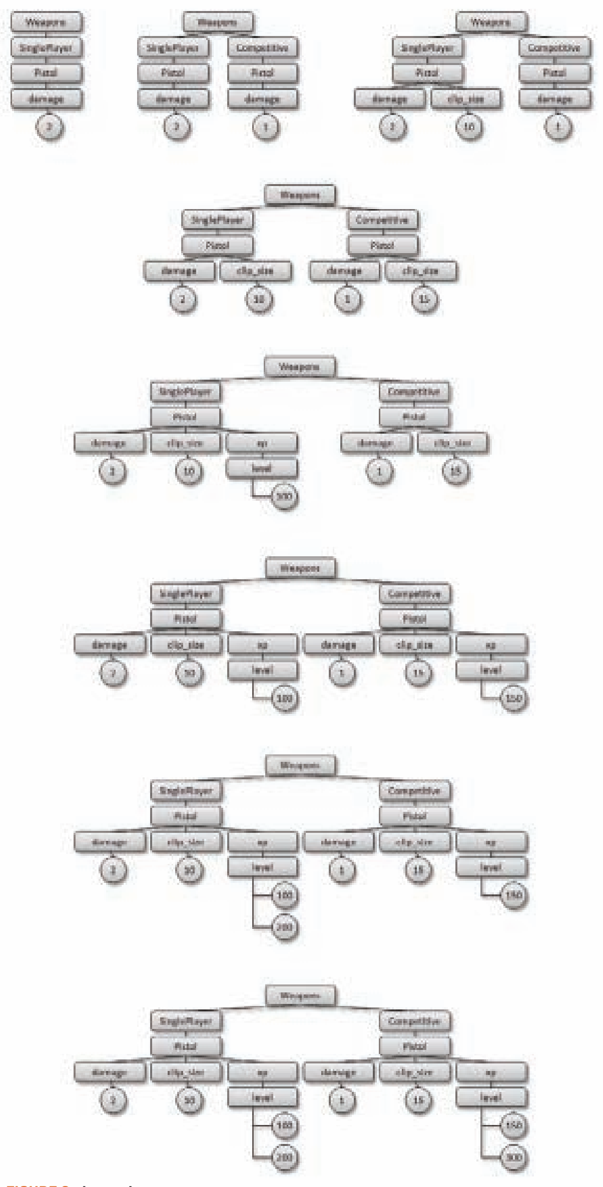


FIGURE 2 shows the string to node process.

The values do not need to be part of this string since they can be read directly from the appropriate cell. Once this string is generated, it's parsed into a tree structure that is grown over the course of the export. In the example above, after parsing "Weapons.SinglePlayer.Pistol.damage," the tree would contain four nodes. The root node would represent an object named "Weapons," and it would have a single child—a node that represents an object named "SinglePlayer." A node beneath the "SinglePlayer" node would be a child node representing the object named "Pistol." Finally, "Pistol" would have a single child node named "damage," which would contain the scalar value "2" (extracted directly from the cell), which represents how much damage the pistol does in a single player game.

As subsequent lines are parsed, nodes are added accordingly to the tree in a similar fashion. This process can be seen in Figure 2. After all lines are parsed, the final tree is traversed and JSON text is output to a file specified by the user. The resulting JSON for the tree in Figure 2 can be seen in Listing 1. This JSON is a representation of how the Pistol weapon should be configured for both the single player and competitive modes of the game. The value types "SinglePlayer" and "Competitive" are grouped in this manner so that it is easy to switch between weapon configuration sets during the game.

After the JSON text is exported, it must be loaded into the game. Fortunately, as previously described, our C/C++ code provides functionality for loading JSON data into structures (the main utility, which we call DDL, is outside the scope of this article). The key to loading JSON data into a C/C++ structure is to make sure the fields in the structure match up with the JSON data accordingly. The C/C++ code, if desired, will use default values (specified by a programmer) for any field that is not present in the JSON data. Furthermore, fields present in the JSON data—but not present in the C/C++ structure—are ignored.

We use [JSON] for loading a variety of data in our new toolset because it's so simple and straightforward, so using JSON as our format for game configuration data came naturally.

Listing 2 shows the C/C++ structures that can be used with the JSON in Listing 1. In practice, programmers usually create these structures first, and then tell designers which fields are available for configuration. Once these structures are in place, programmers can use the C/C++ functionality to load JSON, and the member variables will be set (this is basically a function call on the structure). In general, we have two ways of processing the JSON data—initializing and updating. When initializing a structure, any member variable that is not specified in the JSON data will receive a default value specified by the programmer. When updating a structure, any variable that is not specified in the JSON data is simply not updated at all. This makes it simple to pass changes to objects back and forth in the tools and game as JSON containing a partial description of the object. One thing that stands out when comparing the structures and the JSON data is that all structure member variables are prefaced with "m_," whereas in the exported JSON data (and the Excel spreadsheet) they are not. This is intentionally built into our C/C++ utility; it automatically appends "m_" to variable names to keep the JSON data (and thus Excel configuration data) more human-readable while allowing the C/C++ structures to adhere to our coding conventions.

THE FINAL CHAPTER

>> While it is still in its infancy, the Excel JSON Exporter plug-in has proven useful, and has a few advantages over our previous CSV-based system. The previous system used the Excel built-in CSV exporter. On rare occasions, Excel files would get filled with an unreasonably large amount of empty rows and columns, which made the CSV files take excessively long to load. While the solution is simple enough (just get rid of the excess rows and columns), it's annoying to have to re-export the data and restart the game.

The JSON Exporter deals with this problem in two ways. First, it only parses a fixed number of columns. The JSON Exporter knows how many columns it needs to process based on the header row. Secondly, it will stop processing rows after it hits an arbitrary (but high enough) consecutive number of blank rows. While this is not the greatest solution ever (and may actually be the most embarrassing), it does avoid the problem of Excel sometimes keeping around unused rows and columns.

Another advantage over the previous system is that the CSV data needed to be loaded and parsed by a special system at runtime that used Lua. While we are still parsing data in the new system, the system used to parse JSON is generic and written in C/C++, so the process of loading the configuration data becomes greatly simplified, and most likely faster during day-to-day development since we are not running any Lua scripts in this process.

JSON is a widely supported open standard text format. We use it for loading a variety of data in our new toolset because it's so simple and straightforward, that using JSON as our format for game configuration data came naturally. Our loading code for game configuration data is greatly simplified by our C/C++ utility that matches fields from JSON data to C/C++ structures, and the JSON Exporter for Excel enables us to unify our configuration loading code while allowing designers to work in the environment they prefer.

Excel files and sample data for JSON are available for download at http://www.gdmag.com/resources/code.htm. A version of this technique was previously published as part of Insomniac's Nocturnal Initiative.

GIACOMINO VELTRI *is a programmer on the core team at Insomniac Games. His recent work includes the animation editor in the new toolset and prototyping gameplay in the new engine. Giac is also a graduate of UCLA—go Bruins! [Editor's note: go Trojans]*

LISTING 1

```
{
  "Weapons" :
  {
    "SinglePlayer" :
    {
      "Pistol" :
      {
        "damage" : "2",
        "clip_size" : "10",
        "xp" :
        {
          "level" :
          [
            "100",
            "200"
          ]
        }
      }
    },
    "Competitive" :
    {
      "Pistol" :
      {
        "damage" : "1",
        "clip_size" : "15",
        "xp" :
        {
          "level" :
          [
            "150",
            "300"
          ]
        }
      }
    }
  }
}
```

LISTING 2

```
struct XpData
{
 int m_level[ 2 ];
};

struct WeaponData
{
 int m_damage;
 int m_clip_size;
 XpData m_xp;
};

struct WeaponSet
{
 WeaponData m_Pistol;
 // Any other weapons, such as:
 // WeaponData m_Shotgun;
 // WeaponData m_RocketLauncher;
};

struct WeaponConfig
{
 WeaponSet m_SinglePlayer;
 WeaponSet m_Competitive;
};

struct Configs
{
 WeaponConfig m_Weapons;
 // Other config data
};
```

LISTING 2 C/C++ Structures that correspond to the JSON data in Listing 1.

THE WEIGHT OF SILENCE

HOW SILENCE CAN INDICATE A CHARACTER'S IMPORTANCE

NONE OF THE WORLDS that we deal with exist. Our artist colleagues carve dynamic spaces out of thin digital air. Our engineering brethren wrap those spaces in Newtonian physics and breathe artificial life into an artificial populous. These digital universes, rich with puzzles, platforms, and protagonists though they may be, are 100 percent silent without us.

Silence is negative space, and just as in fine art, negative space is an essential tool of contrast. In interactive audio, we can divide sound into two categories: sounds that happen because of the player and sounds that happen in spite of the player. The way sound designers handle the balance between silence and these two categories of sound can be a useful tool in helping to establish the weight of the player character's importance within the game world.

LESS IS MORE

» UNCHARTED 2: AMONG THIEVES has become one of those must-play games in our industry, and with it, Nathan Drake has arguably become one of this generation of gaming's most iconic characters. Drake's snarky banter is a well-implemented counterpoint to the pop of his M4. Take another detailed listen to the game, though, and an interesting portrait emerges. Despite crumbling buildings and dangling train cars, Drake's world is actually very quiet.

In UNCHARTED 2, not only is Nathan Drake the center of the game's story, he's also the focal point of nearly all the game's sound. Listen to the way urban combat is handled in the Nepalese section of the game. As Drake and Cloe attempt a full-frontal assault on an occupied temple, there are three primary elements to the mix: weapons fire, music, and dialogue, in descending order of importance.

Drake's weapons cut through everything with a Hollywood punch. There's never any doubt as to when the player is firing their weapon. Nor is there any doubt as to when Drake is being fired upon, as enemy weapons have a similar bite. The music is interactive and kicks into action as Drake does. Accompanying the gunfire and music is the interweaving dialogue. AI speech is focused on either taunting Drake or death utterances. For his part, Drake has his own set of utterances as well as the intermittent requisite snark. If Cloe speaks at all, it's to give the player critical gameplay dialogue. Everything on-screen is either happening to or because of Nathan Drake.

The moment combat stops, however, silence begins to intrude. Music drops out quickly without threats to Drake. Some initial dialogue points the player in the story's next direction, but then Drake and Cloe spend large sections without speaking. As the player is allowed to explore the scene and treasure hunt, it becomes apparent that almost all of the sound comes solely from Drake's actions.

Footsteps and foley are quite loud and take center stage alongside the vocal exertions of exploration. If Drake stands still, the world goes quiet. There are short bursts of distant gunfire, but these are infrequent and also quiet. There are occasional point source emitters for effects like small fires, but these are very quiet as well, even when beside them. It's as if the world waits for Nathan Drake, and as such, all of this silent negative space isolates him as the single most important element of the game.

CALL OF DUTY: MODERN WARFARE 2.

MORE IS LESS

» For the sake of comparison, listen to a different take on urban warfare. At the start of CALL OF DUTY: MODERN WARFARE 2, the player steps into the combat boots of Pvt. Joseph Allen as he fights his way through the war-ravaged streets of Afghanistan. Although he is fighting in close quarters with automatic weapons just like Nathan Drake, Pvt. Allen's world is one of complete chaos. Pvt. Allen is literally a faceless grunt, intentionally indistinguishable from the rest of his platoon.

When Allen stands still, his world remains a constant buzz of weapons fire—both near and far—and ever-present radio chatter. Rocket-propelled grenades fire and explode. Jet aircraft scream through the skies. Characters shout dialogue at each other from all sides, occasionally including orders for Allen. Platoon members engage the Afghani militia in a constant hail of AK-47 rounds. The player character's own actions are frequently lost within the din. Foley, reload sounds, and even the faloomp of Allen's grenade launcher can be nearly inaudible beside the chaos of the world around him.

At its calmest, there is no silence for Pvt. Allen. Instead, Pvt. Allen finds himself awash in the sounds of a world fighting around him and in spite of him. If he takes part and adds to the sounds of combat, it barely affects the mix. Only between levels is there a sense of tense quiet in the form of secret backroom conversations, but even then, Allen is merely a cog in the larger machinations of politicians and generals.

Who controls the silence controls their own destiny, and sound designers shouldn't be afraid to use silence—or the lack thereof—as a tool to help indicate a character's importance within their own world.

JESSE HARLIN has been composing music for games since 1999. He is currently the staff composer for LucasArts.

STOP MAKING SENSE

MATCHING THEME TO MECHANICS

Some of our industry's most beloved games make precious little sense. Why, for example, do players battle the trolls, goblins, and skeletons of PUZZLE QUEST by challenging them to a two-player version of BEJEWELED? Similarly, success in PROFESSOR LAYTON's world seems to revolve disproportionately around one's ability to solve classic logic and deduction puzzles, no matter the reason.

Game stories have fared no better. MARIO's canonical plot sounds like nonsense from Lewis Carroll: the plumber punches bricks to find magic mushrooms that double his size, so that he can battle an evil turtle who has kidnapped the kingdom's princess. The less said about the METAL GEAR SOLID franchise's various twists and turns—including the infamous possession of Revolver Ocelot's mind by Liquid Snake's old arm—the better.

Still, games have their own internal logic, which is much more important than whether the game's story makes sense, or even whether the game's mechanics hold together logically. The traditional concepts of levels, lives, and respawns are ultimately constructs that support a designer's vision, regardless of whether they have any logical real-world parallel or thematic metaphor.

Why, for example, should players respawn—come back to life—after being killed in a team-based shooter? Shouldn't players expect their dead character to stay dead after being killed? The reason is that the respawn mechanic matches the inviting tone the game's designer wishes to strike. By softening the blow of death, gamers are free to play aggressively, which rewards risk and even experimentation.

A place exists for games which do not allow respawning—COUNTER-STRIKE being the most successful example—but the designer chooses this mechanic not in pursuit of realism, but to strike a different tone. When characters stay dead, players feel more tension during the match, which encourages them to play more carefully and with greater precision. Thus, games without respawns simply occupy a different location on the play spectrum.

One great advantage of not worrying about a game making sense is that designers are free to use the theme which best matches the game's mechanics.

BE TRUE TO THE GAME

» Sometimes these imaginary design constructs are necessary for the existence of entire genres. The classic real-time strategy design pattern, with peons, base-building, and rush/turtle/boom dynamics, bears little resemblance to actual warfare, even when ignoring the common fantastical themes. In what type of war does each side construct army barracks to train troops—and even research labs to discover technologies—on the very field of battle? Indeed, why is every scientific breakthrough forgotten between each scenario of a fictional campaign?

Ultimately, these questions are subsumed by the genre's needs. Strategy games work because players are forced to make tough choices between a number of options, each with its own set of trade-offs. Although the environments of most real-time strategy battles often contain nonsensical elements, such as economic infrastructure and research facilities, these elements each create important mechanics that increase strategic depth.

Creating infrastructure gives the player an actual location on the map to defend; without it, armies could roam freely across the map with no consequences for abandoning a certain location. Discovering technologies creates short-vs-long-term trade-offs for the player to balance—should resources be invested in science for a long-term payoff of stronger units or spent on new units to attack the enemy and press an early advantage?

These trade-offs make sense in a fundamental way—players understand that location should matter and that making long-term investments should succeed under the right circumstances. Therefore, the gameplay itself makes sense, even if the game's world does not—workers planting farms within sight of a pitched battle and all.

TOO MUCH CONSISTENCY

» Indeed, designers who worry too much about a consistent world can often hamstring their own work. In STARCRAFT, the designers had no qualms allowing Terran players to team up with the Zerg in multiplayer, even if fighting against other Terrans. However, COMPANY OF HEROES only allows matches with the Axis on one side and the Allies on the other. Clearly, this decision makes sense thematically, but does it make sense that players never get to pit identical sets of virtual army men against each other?

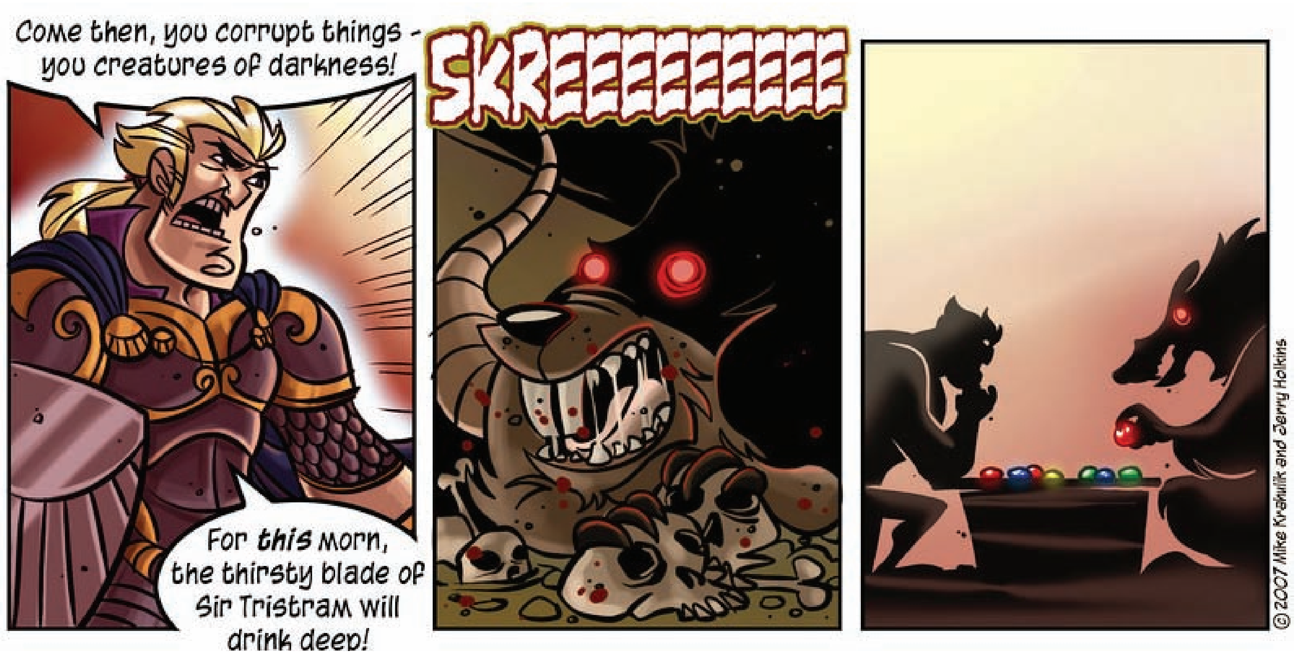
ASSASSIN'S CREED famously went to great lengths to cover up as many standard game conventions as possible. A frame story put the player in the shoes not of a 12th-century Middle Eastern assassin (as the game's advertisements featured) but of his 21st-century descendant who is somehow reliving the former's life with advanced memory reconstruction technology.

This conceit aims to explain a number of typical design constructs. Discrete game levels are simply different memories, while all character deaths must be false memories. The assassin's movements are mapped to a physical gamepad because he is actually the puppet of a latter-day character trying to relive his memories.

Did these rationales broaden the game's appeal by explaining supposedly arbitrary gaming cliches? Or did they unnecessarily burden the game's narrative with a convoluted and unnecessary frame story that distanced players from the fantasy of being a medieval assassin? Surely, the average console owner would not be surprised that the game required controlling the character with a gamepad.

Indeed, the early arcade industry was a font of creativity largely because the games were not expected to make any sense—think of the dot-eating PAC-MAN, or the cube-jumping Q*BERT, or the ray-running TEMPEST. As graphics became more realistic, almost all arcade cabinets were ghettoized into just a few concrete categories —racing,

ILLUSTRATION COURTESY OF PENNY ARCADE


Come then, you corrupt things - you creatures of darkness!
For this morn, the thirsty blade of Sir Tristram will drink deep!
SKREEEEEEEEEE
©2007 Mike Krahulik and Jerry Holkins

fighting, and shooting—because the higher-resolutions discouraged bizarre, abstract games. Only now that downloadable, mobile, and Web-based gaming have brought back the use of lower resolutions is the old eccentric energy returning.

GO YOUR OWN WAY

>> Sometimes, manipulating a game's story to paper over unusual design concepts can work. Certainly, the Dagger of Time's ability to rewind time for a few seconds in PRINCE OF PERSIA: THE SANDS OF TIME was an elegant way to integrate a quick-save system into the game's core functionality. In the recent TORCHLIGHT, the character's pet can run back to town to sell loot, nicely shortening a time-consuming element of most action-RPGs while also staying within the game's fiction.

Still, designers should feel comfortable going their own way if a mechanic makes sense for the game they want to make. SHIREN THE WANDERER is a roguelike dungeon crawler, which means that all character deaths are permanent as progress cannot be saved. Roguelikes are meant to be played repeatedly, with the player improving purely through increased knowledge of the game's rules.

However, SHIREN does allow a very unusual type of progress by letting the player stash loot—including powerful weapons and armor—in various caches found throughout the game that have persistence between sessions. Thus, although a character might die an unlucky death, he still contributes to advancing the game by leaving a supply of potions for the next character's playthrough.

This strange mechanic, where most (but not all) of the world resets on death, has few parallels either inside or outside of gaming, and the story makes no attempt to explain it. Truly, no explanation is necessary because the game is being true to itself; the designers wanted a game that combined the tense atmosphere of permadeath with a touch of power progression from a traditional RPG.

BIOSHOCK is another game which gave no explanation for an absurd element—the audio diaries which are littered about the underwater city of Rapture. These bits of recorded speech from the game's main characters provide important backstory for this Objectivist dystopia. Still, what type of person would, after putting their personal thoughts onto tape, decide to break up the tape into pieces and then scatter those pieces around the world like junk?

That the player discovers these scattered bits of audio in roughly linear order allows the designer to tell the story without relying on stodgy cutscenes, but their placement in the world simply doesn't make sense. This doesn't mean that the designers made the wrong choice; perhaps a more elegant solution was possible, but better to allow a little inelegance than to turn the player into a non-interactive viewer who must be force-fed the story.

THE PERFECT THEME

>> One great advantage of not worrying about a game making sense is that designers are free to use the theme which best matches the game's mechanics. The tower defense genre emerged from user-created scenarios designed for real-time strategy games like STARCRAFT and WARCRAFT III.

The limitations of these platforms gave the genre a distinct set of conventions—stationary defenses vs. mobile "creeps"—which had little narrative justification. Why must all defenses be static? Why are the creeps so slow and mindless? If only a thematic environment existed which matched this set of game mechanics.

In fact, one did, but the designers just needed the confidence to pull it out of thin air. What type of life-form can grow but can't move? Plants! What type shambles along slowly in a straight line without a brain? Zombies! Naturally, the answer was to pit these two groups against each other.

With PLANTS VS. ZOMBIES, PopCap found the perfect theme for a tower defense game. The fact that it completely defied common sense was beside the point. Why are players battling zombies with mutant plants, after all? That doesn't matter; the important thing is that even someone who is unfamiliar with the tower defense genre would have an intuitive understanding of what to expect simply from the game's title, all because the designer wasn't afraid to stop making sense.

SOREN JOHNSON *is a designer/programmer at EA2D, working on web-based gaming with strategystation.com and DRAGON AGE LEGENDS. He was the lead designer of CIVILIZATION IV and the co-designer of CIVILIZATION III. Read more of his thoughts on game design at www.designer-notes.com.*

FROM LARA CROFT TO LAME CASTLE

CRYSTAL DYNAMICS VETERAN GOES INDIE

Brad Johnson went from lowly tester, to basic scripter, to full fledged programmer, fighting his way up through the ranks in classic style. But after completing work on LARA CROFT: GUARDIAN OF LIGHT, he realized it was time for a change, and went independent with his new company Be-Rad Entertainment. We spoke with Johnson about his leap to freedom.

***Game Developer:* What made you go from LARA CROFT: GUARDIAN OF LIGHT to the iPhone/Android space?**

Bradley Johnson: We made a great game, but it wasn't a cakewalk getting there. Overtime coupled with a 45 minute commute each way began to drain my soul. I'd come home unhappy and pissed off at the world. I was tired of feeling that way and always wanted to make my own games, so I left and started my own company.

My goal with Be-Rad Entertainment is to make a larger scale game for XBLA/PSN/Steam, but the way I'm going to get there is to take components that the larger game will use, create the component, and then make a smaller game out of it. This way I can work out any kinks in the system, and if it's designed correctly it will allow me to easily plug it into another game. On top of that, I'll have a finished product to show for it, which will hopefully help fund the big one. Once I create all the small components necessary for the larger game I'll start cranking it out. Until then I'm really enjoying the freedom being an indie gives me—creating my own schedules and developing whatever games I want to work on.

***Game Developer:* Did you already know what you wanted to make, or did you just know you wanted to go indie?**

BJ: Sort of not really! But it came together fast. About two weeks after I quit my job to go indie I started working on a prototype for an iPhone game. This game involved a ninja and lots of jumping. After about two weeks of work my buddy sent me an email with a link to a PSP ad and wrote, "We should make this game." In the ad they're making fun of what looks like an iPhone. The guy in the ad is holding his phone with a game called LAME CASTLE on it. Once I saw that I wrote back and said, "Yes, we're going to make that game."

That PSP ad gathered a decent amount of exposure on the web from various blog sites so I knew if we made the game we could leverage that to our marketing advantage. I immediately squatted on the iphone app name and domain name.

We wanted to pump a game out quick so that people wouldn't forget about the ad, so we had to act fast. Since I already had a prototype for another game we decided to use what was there and turn it into what's become LAME CASTLE.

A week into development we had a pretty decent prototype. A week after that we found an artist and a sound guy and started plugging everything in. A couple weeks after that we were in the final stages of polish. It's been a very quick development cycle, but it's been quite smooth.

***Game Developer:* Any rough spots in moving from being part of a team to full development on your own?**

BJ: The best thing we did was to get it in peoples' hands for playtesting. Watching people play the game for the first time made us quickly realize we had several things in the game that people just didn't understand. One of them was a chicken leg pickup that gave you boost power. When we first put it in we thought it would be funny to have a "MEAT-er" for your power meter. Get it? But people just didn't make the connection, so we turned the pickup into a lance with a little explosion on the tip, and now people get it.

Making iPhone games is quite a bit different than console games. If the audience can't pick up your game and figure out what's going on in 30 seconds then you've probably already lost 95 percent of your audience. That's why playtesting has been so important, so that we can recognize how people play the game and fix the problem areas.

***Game Developer:* Any advice for those thinking of making the jump?**

BJ: Looking back I can easily say I wish I had gone indie sooner.

new studios

Continuing its rapid expansion, FARMVILLE developer Zynga is opening a Seattle office and hiring a team of web engineers for the new location.

Following news of Zynga's plans to found a Seattle studio, the developer of FARMVILLE and MAFIA WARS said it is continuing to expand its operations into Ireland.

Following his recent departure from Namco Bandai, KATAMARI DAMACY creator Keita Takahashi has announced his new studio, Uvula, which offers services in art, music, and video games.

THQ has opened a new development studio in Montreal, including talent such as Ubisoft veteran Patrice Desilets, the former creative director of the ASSASSIN'S CREED franchise.

who went where

Neonga, the free-to-play game company formed by Frogster's ex-CEO, has added another Frogster veteran, Stefan Hinz, to its staff as chief marketing officer.

Just after the launch of his latest project, VANQUISH, renowned RESIDENT EVIL creator Shinji Mikami became part of the ZeniMax Media family—the Bethesda Softworks parent has acquired his new Tokyo-based development house, Tango Gameworks.

Microsoft executive Scott Henson has assumed the role of studio manager of Microsoft-owned UK developer Rare.

Hoping to travel in Asia with his family, Take-Two CEO Ben Feder has stepped down from his executive role at the GRAND THEFT AUTO publisher, and the company's chairman, Strauss Zelnick, will assume the chief executive role.

MEGA MAN creator and 23-year Capcom veteran Keiji Inafune has left his position as head of global production at the company, and has yet to announce his plans for the future.

The Academy of Interactive Arts and Sciences announced today that Joseph Olin has stepped down as president of the organization, and taking his place is former Sunleaf Studios CEO Martin Rae.

2011 Independent Games Festival

THE ORGANIZERS OF THE 13TH ANNUAL INDEPENDENT GAMES FESTIVAL—THE LONGEST-RUNNING AND LARGEST FESTIVAL RELATING TO INDEPENDENT GAMES WORLDWIDE—ANNOUNCED ANOTHER YEAR OF RECORD ENTRY NUMBERS FOR IGF 2011'S MAIN COMPETITION.

In total, this year's Main Competition took in just under 400 game entries—many of them new titles from leading indie developers—across all platforms.

This includes 150 entries for mobile hardware like the iPhone, iPad, DS, PSP and Android devices, with all mobile entries now eligible for all IGF 2011 prizes, including a unique Best Mobile Game award.

In-depth information and entrant-provided screenshots and videos on each of the IGF Main Competition entries are now available on IGF.com, a feature unique to the contest.

Some of the titles entered in the IGF Main Competition this year include SuperMono's real-life RPG tasklist EPICWIN, Monobanda's zen-like BOHM, indie party game hits like Copenhagen Game Collective's B.U.T.T.O.N. and Messhof's NIDHOGG, Vblank Entertainment's parodic 8-bit revival RETRO CITY RAMPAGE, and Matt Gilgenbach's A MOBIUS PROPOSAL, a game created specifically to (successfully!) propose to his girlfriend.

In addition, a number of returning developers previously honored at the Independent Games Festival have entered new games, including LIFE/DEATH/ISLAND, the latest from 2010's Nuovo Award winner Cactus, and both KOMETEN and SHOT SHOT SHOOT from 2009 Grand Prize winner Erik Svedang. This year also sees a number of prior IGF Mobile winners and finalists joining the main festival—with entries like Steph Thirion's FARAWAY, Mobigame's PERFECT CELL, and Gaijin Games' new console port of Different Cloth's LILT LINE.

Some of the other games previously known to the indie community and entering this year include two entries in the BIT.TRIP series by Gaijin Games, ROLANDO creator HandCircus' first non-iDevice game, OKABU, Nicalis's re-imagined version of foundational indie title CAVE STORY, and Mojang's recent headline-grabbing surprise-hit MINECRAFT.

In addition, several teams made up of formerly 'mainstream' developers have also chosen the 2011 Independent Games Festival to debut new indie works—with SuperGiant's BASTION, Haunted Temple's SKULLS OF THE SHOGUN and former Maxis developer Chris Hecker's SPY PARTY being just a few of the entered titles.

These titles are just a fraction of the games that are debuting for the first time as an Independent Games Festival submission. In fact, history has shown that some of the most notable and award-winning games—from AUDIOSURF through WORLD OF GOO and beyond—were relatively unknown at the time of their submission, so indie game aficionados should carefully browse all titles to find the many hidden gems.

"I'm thrilled with both the growth and the diversity that the Independent Games Festival has shown in its 13th year," said festival chairman Brandon Boyer. "This year's entrants happily cover the entire spectrum from more polished and commercial works to smaller, more personal and artistic statements to entries geared toward the resurgence of more social, new-arcade-type play. We're all looking forward to sitting down with each game and starting the conversation as we determine finalists!"

This year's IGF entries will be distributed to more than 150 notable industry judges for evaluation, and their highest recommendations passed on to a set of elite discipline-specific juries for each award, who will debate and vote on their favorites, before finalists are announced in January 2011.

In turn, winners will be awarded at the IGF ceremony during the Game Developers Conference 2011 in San Francisco next March, and all finalists in the Main Competition (including the art-centric Nuovo Award) and the Student Showcase (which is due for submission by November 1st) will be showcased on the GDC Expo Floor from March 2nd–4th, immediately following the 4th Annual Independent Games Summit on February 28th and March 1st.

2010 Independent Games Festival China Finalists

The organizers of the second annual Independent Games Festival China have revealed finalists for both the Main and Student competitions, which included submissions from across Asia.

High-quality submissions for the second iteration of the event—a newly formed sister competition to the main yearly Independent Games Festival in San Francisco—were received from multiple Chinese provinces, Hong Kong, Taiwan, Singapore, South Korea, Australia, New Zealand, Iran, India and beyond.

Finalists were chosen by a panel of distinguished local judges, including representatives from Shanda Games, Tencent, IGDA Shanghai, TipCat Interactive and more.

IGF China finalists are invited to Shanghai for the Game Developers Conference China event from December 5–7, where they will be showing their games at a special Pavilion on the Expo Floor, open to all GDC China attendees.

In addition, finalists are eligible to win up to RMB 61,000 ($9,100) in cash prizes, as well as specially created awards and All Access Passes to GDC San Francisco 2011 worth thousands of dollars.

The Main Competition finalists for the 2010 Independent Games Festival China are:

SUGAR CUBE (Turtle Cream, South Korea)
HAZARD: THE JOURNEY OF LIFE (Alexander Bruce, Australia)
TRAIN CONDUCTOR 2 (The Voxel Agents, Australia)
CUT & PASTE (Turtle Cream, South Korea)
SKILLZ: THE DJ GAME (Playpen Studios, Hong Kong)
BUTAVX: JUSTICE FIGHTER (Nekomura Games, Singapore)
CROSSOUT (Coconut Island Studio, China)

The Student Competition finalists for this year's IGF China event are as follows:

ZONELINK (Huazhong University Of Science And Technology, China)
DEAD STEEL (Media Design School, Auckland, New Zealand)
AFTERLAND (Singapore-MIT GAMBIT Game Lab, Singapore)
THE WHITE LABORATORY (Huazhong University of Science & Technology, China)
PONLAI (National Yunlin University of Science and Technology, Taiwan)

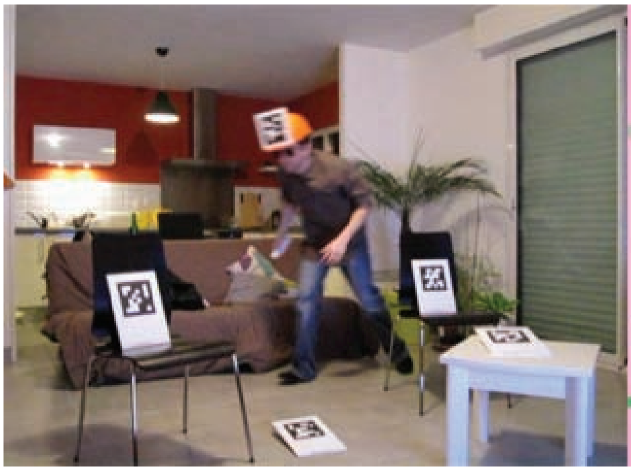
The winners of IGF China in categories including Best Game, Mobile Best Game, Excellence in Art Direction, Excellence in Visual Arts, Technical Excellence, and Student awards will be announced during the 2010 Game Developers Conference China at a special IGF Awards ceremony.

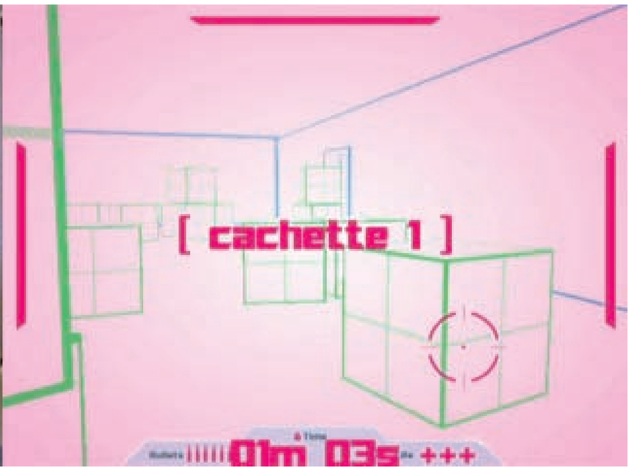
The Independent Games Festival's outreach into Asia is part of GDC China, which returns to the Shanghai International Convention Center on December 5-7th, and early registration is open until November 5th. Further information on IGF China can be found at the event's official website.

immersive rail shooter

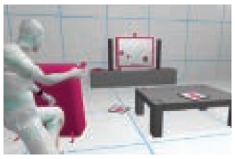
BUILDING A VIRTUAL PILLOW FORT

DAVID ARENOU'S IMMERSIVE RAIL SHOOTER IS AN AUGMENTED REALITY PROJECT THAT CLEVERLY ALLOWS PLAYERS TO DUCK BEHIND THEIR LIVING ROOM FURNITURE IN ORDER TO AVOID INCOMING FIRE FROM THEIR VIDEO GAME ENEMIES. USING A VARIETY OF OFF-THE-SHELF COMPONENTS AND OPEN SOURCE LIBRARIES, ARENOU CREATED IMMERSIVE RAIL SHOOTER FOR HIS DIPLOMA PROJECT AT L'ECOLE DE DESIGN NANTES ATLANTIQUE IN NANTES, FRANCE. THE RESULT IS A FASCINATING EXAMPLE OF HOW LOW-BUDGET, EXISTING TECHNOLOGY CAN BE COMBINED TO CREATE RADICALLY NEW GAME DESIGNS.





Jeffrey Fleming: How quickly were you able to go from concept to a working prototype?
David Arenou: It took me almost four months from the birth of IMMERSIVE RAIL SHOOTER to the end of the programming. I wrote the concept basics for two months. Then, I spent a month and a half in full-time to program the prototype. Let me add that before these two phases, I had two other months where I did a lot of research and analysis in order to position my diploma proposal.



JF: What tools did you use to create IMMERSIVE RAIL SHOOTER?
DA: I'm used to working with 3DVIA Virtools, an environment that allows you to create 3D real-time applications. It's easy to get convincing results with it, so I naturally headed toward it to make the core of my game.

JF: Tell me about some of the technology driving IMMERSIVE RAIL SHOOTER. Are you using any special cameras?
DA: As an interaction designer, I'm used to doing rapid prototyping and taking advantage of simple and already working technologies. So I didn't use any special cameras or algorithms written by myself.

JF: How does the computer recognize the marker cards that are placed in the play area?
DA: I used an existing library, ARToolKit, which detects and tracks the position of special patterns that you print out and stick to objects in your environment. With a simple webcam, the library, and Virtools, half of the work was already done!

JF: What about the Wii remote? How is that integrated?
DA: Thanks to Bluetooth and the GlovePIE software, I was able to connect my Wiimote to my PC.

Finally, I bought a cheap infrared sensor bar so I could use the remote as a pointer. I could build it myself, but it was cheaper and way faster to order it from Amazon!

Building IMMERSIVE RAIL SHOOTER wasn't a question of extremely complicated technologies, but more a matter of composing accessible stuff.

JF: The possibilities in IMMERSIVE RAIL SHOOTER for versus play against another person seem like a lot of fun. How far along are you toward implementing that?
DA: Versus and cooperation modes are pretty exciting mods! Unfortunately, these ideas were just a post proposal to my main concept. I considered them during the design phase of the project in order to present a full experience that answers to both solo and multiplayer play, but in the end, I put them aside for the purpose of focusing on the single-player prototype.

JF: We used to hear a lot about virtual reality but most of that work was never really suited for the masses. However, augmented reality seems to be much more accessible. What do you think is driving that?
DA: Indeed, augmented reality is more accessible than virtual reality because AR systems are really light and efficient: a camera and a good algorithm are enough to create the expected results. I'm not saying that you don't need good ideas anymore, but from a technology point of view, you will rapidly have nice feedback with AR.

Furthermore, people like AR because they really are in the center of the application. They see their own environment and can interact directly with any kind of virtual object. It's easy to understand, immersive, and fun!

—Jeffrey Fleming

IMMERSIVE RAIL SHOOTER
http://portfolio.davidarenou.com/video-game/immersive-rail-shooter
3DVIA Virtools
www.3dvia.com/products/3dvia-virtools
ARToolKit
www.hitl.washington.edu/artoolkit
GlovePIE
http://sites.google.com/site/carlkenner/glovepie

TURN YOUR PASSION FOR GAMING INTO A CAREER

Game Art
Bachelor's Degree Program
Campus & Online

Game Design
Master's Degree Program
Campus

Game Development
Bachelor's Degree Program
Campus

Game Design
Bachelor's Degree Program
Online

Campus Degrees

Master's
Entertainment Business
Game Design

Bachelor's
Computer Animation
Digital Arts & Design
Entertainment Business
Film
Game Art
Game Development
Music Business
Recording Arts
Show Production
Web Design & Development

Associate's
Graphic Design
Recording Engineering

Online Degrees

Master's
Creative Writing
Education Media Design & Technology
Entertainment Business
Entertainment Business: *with a Sports Management Elective Track*
Internet Marketing
Media Design

Bachelor's
Computer Animation
Entertainment Business
Game Art
Game Design
Graphic Design
Internet Marketing
Music Business
Music Production
Web Design & Development

FULL SAIL UNIVERSITY

fullsail.edu

Winter Park, FL
800.226.7625 • 3300 University Boulevard
Financial aid available to those who qualify • Career development assistance
Accredited University, ACCSC

ADVERTISER INDEX

COMPANY NAME	PAGE
Epic Games	8
Full Sail Real World Education	47
Havok	C3
Natural Motion	14
Rad Game Tools	C4
Scaleform Corporation	C2
Transgaming	3

Game Developer (ISSN 1073-922X) is published monthly by United Business 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 United Business 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. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. 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/United Business Media LLC, unless otherwise indicated. Don't steal any of it.

OUR LAST, BEST HOPE

THE PROJECT THAT WILL SAVE OUR STUDIO

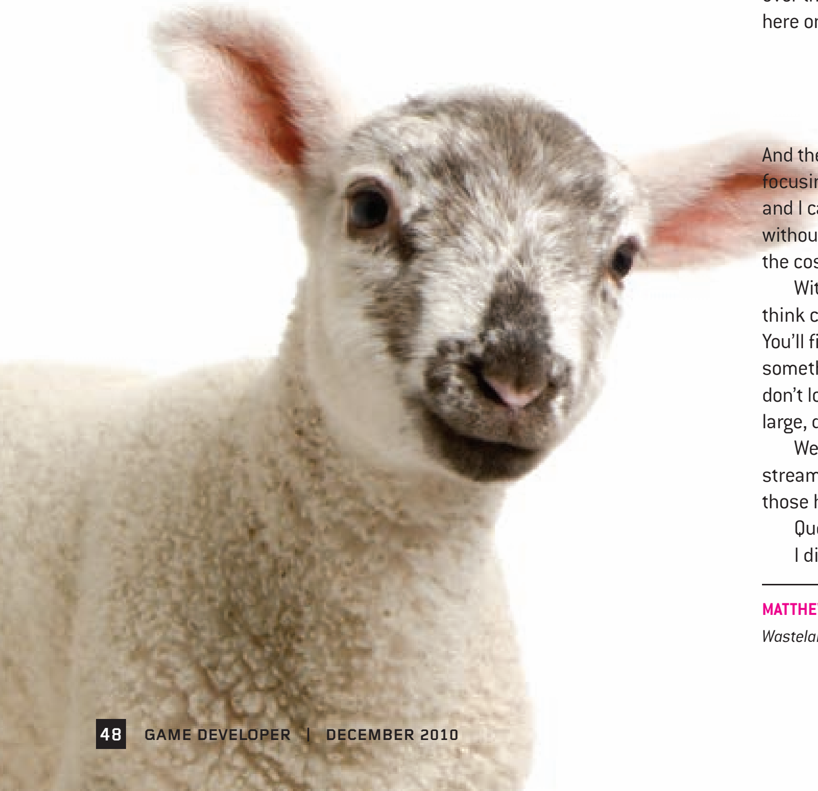
HI EVERYONE, AND WELCOME TO THE COMPANY MEETING. YOU COULD CALL this the fun-size company meeting, I guess. But hey, look on the bright side: if we hadn't had those layoffs last week, we wouldn't all fit into this conference room right now.

Rough crowd! Okay ... Well, let's just get right down to brass tacks then, shall we? I know a lot of you have questions on your mind, like, are we going out of business, what's going on, why did my paycheck bounce, and so on. I'll take a brief moment to address those rumors: they aren't true. We're still going strong and charging ahead at full speed on our new project. I'll get to that in a moment.

I know some of the recent turbulence might have given you a bad feeling regarding our future. I won't lie to you, we're going through a challenging period. But we're not alone: it's everyone. That's right, the whole game industry is suffering right now. Valve is laying off employees left and right. Blizzard is close to shutting down. And over at Zynga, they're boiling grass and wood chips just to stay alive. That's secret information, so don't tell anyone I said that.

My point is that there's nothing we could have done. Even if we had the best management team in the world, which, I dare say, we are pretty close to having—right, Fred?—you can't avoid layoffs once in a while. That's just how the system works! If it were up to me, I'd have given everyone a big raise. Honest!

On that note, it's time to get to the exciting part of the meeting: the new project. There's been a lot of buzz building about it amongst the team members, and now we're finally ready to reveal the full plan. Now, I don't like to exaggerate, but this game is going to rocket us out of these tough times and straight to the top of the heap. If you could get the lights, Fred? Some d-bag made off with the projector—oughtta sue his ass—uh, so I'll just turn my laptop around here, and if you could all sort of crowd in together? Let me load up the presentation.



Here we go: WORLD OF ALIEN FARM CRIME. As you probably already guessed from the name, it's a massively multiplayer, open-world farm simulator set against the backdrop of a brutal alien invasion. Let me read from the pitch document a bit: "You, along with hundreds of thousands of your friends, are humanity's last hope. Complete quests, commit crime sprees, grow crops, and participate in a dynamic, constantly changing interstellar war!"

I can see from the looks on your faces you're already completely amazed. But it gets better: "WORLD OF ALIEN FARM CRIME will have the grand scope of an MMO, the balanced competitive multiplayer of an FPS, and the open-world freedom of, uh ... of an open-world game!" Hmm, should have re-written that. "It has co-op, downloadable content, social networking, a guitar peripheral, and motion control. The game boasts 19 distinct, completely unique races, 43 character classes, and a dialogue script that is 3,000 times as long as the 1974 edition of the *Encyclopedia Britannica*."

Guys, you would not even believe how excited our publishing partners got for this pitch. I told them to imagine how much money HALO has made, and then multiply that figure by how much money WORLD OF WARCRAFT has made, and that's how much money WORLD OF ALIEN FARM CRIME will make. Then, to clinch the deal, I said we could make it for half the price and in half the time of our nearest competitor. Because I know we can.

Needless to say, they signed us up on the spot. Fastest deal I've ever done, in fact. Now we'll move on to—sorry, a question? Technology, you say? Well ... tech is, you know, just one of those bridges that we'll cross when we get to it. You know? I'm sure the tech team here will work out all the details, because they're a really talented, great group of guys. Actually, all of you are, everyone at this company. With your talent, your dedication, your drive, anything is possible. I mean it.

Where was I? Right. So the technology will sort itself out, but what about the important stuff—the corporate philosophy that will guide us to victory over the big boys? As it happens, I have an answer for you, and that's right here on the next slide. Pay attention, please.

EFFICIENCY

And there we go. "Efficiency" is the key concept that we're going to be focusing on from here on out. I've been thinking long and hard about this, and I came to the realization that as long as we're efficient, we can cut costs without cutting corners. We can develop big, triple-A games at a fraction of the cost and time spent by our soon-to-be dinosaur competitors.

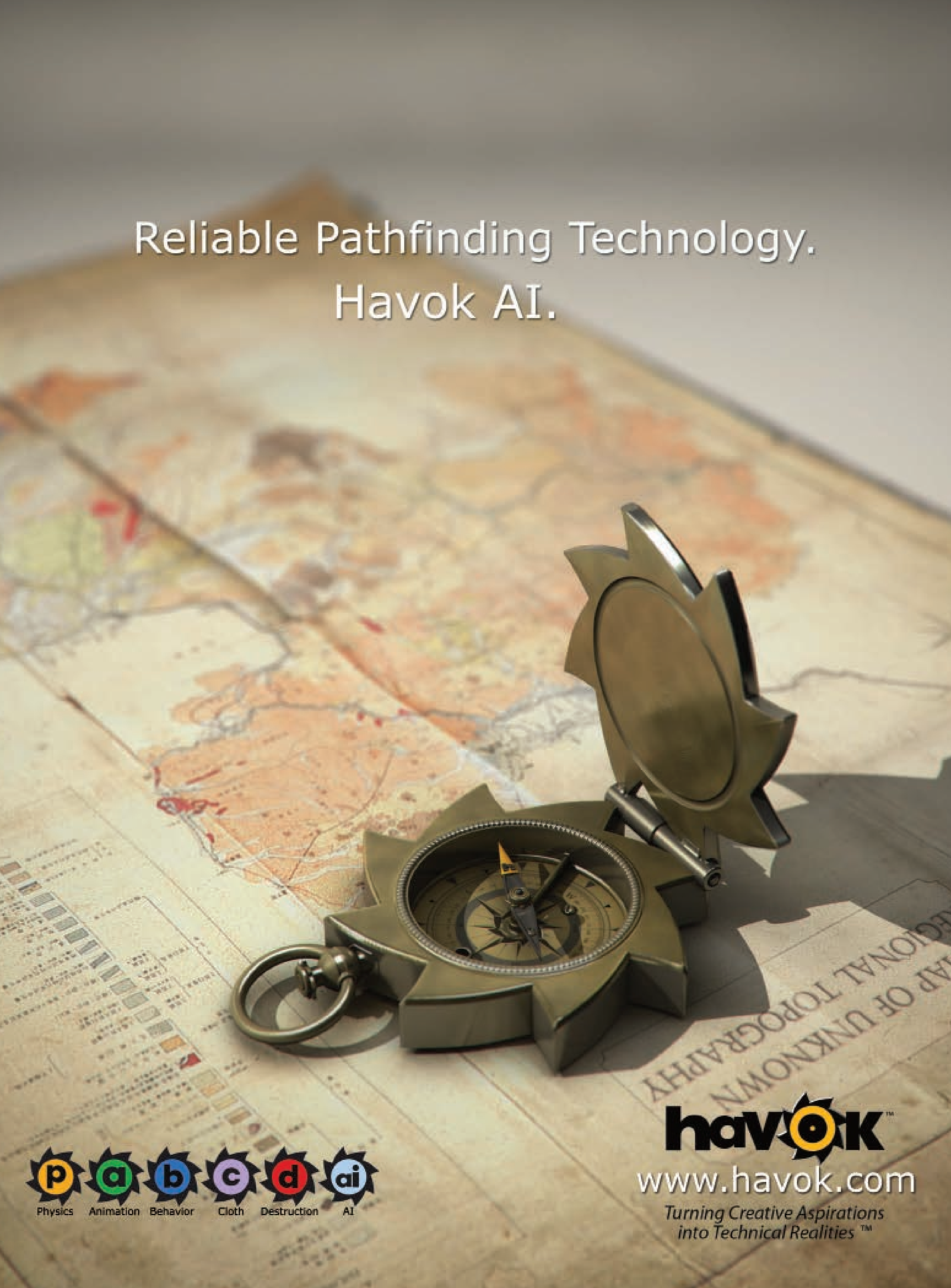
With that in mind, starting today, I want each and every one of you to think constantly about how we can improve efficiency across the company. You'll find that there's efficiency to be gained everywhere. It could be something as simple as moving the trash can nearer to your desk, so you don't lose time walking over to it. Or it could be developing a way to create large, detailed, and optimized level environments with the touch of a button.

We're already part of the way toward realizing the goal with our new streamlined, refocused team size. So all we need to do now is batten down those hatches, make WORLD OF ALIEN FARM CRIME, and ship that sucker.

Questions? Concerns?

I didn't think so. Let's get to work!

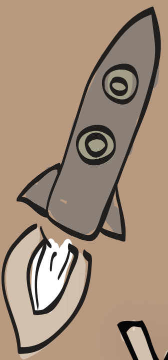
MATTHEW WASTELAND *writes about games and game development at his blog, Magical Wasteland (www.magicalwasteland.com).*


Reliable Pathfinding Technology.
Havok AI.
REGIONAL TOPOGRAPHY
MAP OF UNKNOWN
Physics
Animation
Behavior
Cloth
Destruction
AI
havok
www.havok.com
Turning Creative Aspirations into Technical Realities ™

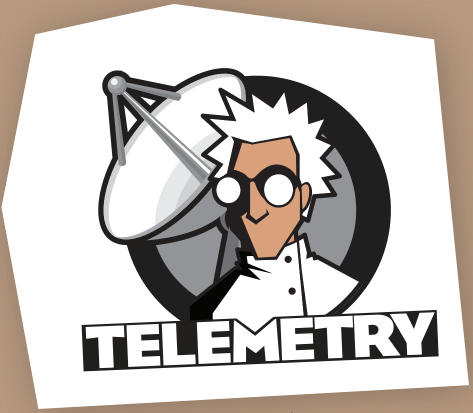
THE NEWEST RAD TOOL IS

yes, it's NEW

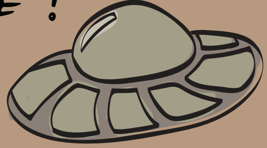
NOW SHIPPING



OUT OF THIS WORLD



TELEMETRY is a programming library and set of tools for instrumenting, profiling, tuning and visualizing application PERFORMANCE!



VISUALIZE real-time game performance,

see **WHEN** things happen — not merely **WHAT** happened!

PROBE the hierarchical display —

see thread interactions, context switches, and mutex locking!

THIS ISN'T JUST rocket science, THIS IS rad!

RAD© GAME TOOLS

www.radgametools.com/telemetry
(425) 893-4300