United Business Media

2008 FRONT LINE AWARDS ARE HERE!

JANUARY 2009

game developer

THE LEADING GAME INDUSTRY MAGAZINE

POSTMORTEM
MEDIA MOLECULE'S LITTLE BIG PLANET

iPHONE SDK
STARTING OUT WITH APPLE'S HANDHELD

TOOL BOX
AUTODESK MAYA 2009 REVIEWED

PIXEL PUSHER
DESIGNING ARTIST-FRIENDLY SHADERS

Let us take care of your clothes.
havok
cloth.havok.com

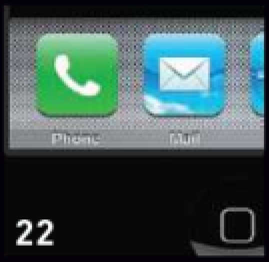
game developer

CONTENTS

JANUARY 2009
VOLUME 16, NUMBER 1

16

7



22

COVER ART: REX CROWLE

POSTMORTEMS AND PRODUCERS

LAST MONTH WE RAN AN ARTICLE CALLED *What Went Wrong*, highlighting common mistakes in game development as seen through postmortems. There was one area which I wanted to highlight, but didn't have the space for—when developers continue to make the same mistakes they've made before.

It comes up quite often. The author begins by saying something like "this is really important to us as a studio," but then goes on to say how they went ahead retread old ground. Here are some examples from past postmortems.

PENNY ARCADE ADVENTURES: ON THE RAIN SLICK PRECIPICE OF DARKNESS

"Being an experienced team meant a lot of us had developed certain 'best practices' that we used to maximize the quality of the final product. It was sobering near the end of the project to realize that despite knowing these things, we had simply failed to employ them."

—Joel DeYoung, HotHead Games

GUITAR HERO

"For the most part, we were successful in creating a complete and detailed schedule early in the project and sticking to it. However, there were a number of seemingly small and mundane features (such as the unlock store, the intro cut-scene, and the win sequence) that were either underspecified or didn't make it into the schedule at all, and they added up to quite a bit of work.

"This is a classic developer misstep, and one that we've made before. We thought we had learned our lessons and applied the necessary structure to avoid this problem, so it really stung when it cropped up again."

—Greg LoPiccolo, Daniel Sussman, Harmonix

AGE OF BOOTY

"One serious consequence was a breakdown in cross-disciplinary communication, something we take a lot of pride in, as evidenced by our completely open pit-style office."

—Max Hoberman, Certain Affinity

GUN

"With human resources in short supply, we let a critical Neversoft convention fall apart: game and mission reviews."

—Scott Pease, Chad Fidley, Neversoft

STRANGLEHOLD

"Quality of life is really important to us, and when we started we really tried to limit crunch but in the end we still failed miserably."

—Brian Eddy, Midway Chicago

PROBLEM OF PRODUCTION

These comments are all from accomplished developers who should know better—and in fact often do. Why then does this happen? It's one thing to simply have an in-house postmortem, but it's another entirely to actively try to learn from and fix the problems that are highlighted as a result.

A lot of these issues have to do with scheduling and communication, which in turn has a lot to do with the quality of producers. Proper producers seem to be sorely lacking in the game industry, and often wind up as glorified spreadsheet keepers.

A good producer should be fixing problems with the production pipeline—identifying areas where documentation is lacking, like in the GUITAR HERO example, and then making sure that gets done. Or in the case of GUN, making sure that a "critical convention" doesn't get lost in the shuffle.

As far as I can tell, this job confusion isn't solely the fault of the production team, but is also tied in to the fact that the structure of most companies is not conducive to producers actually doing production work. Due to improper job descriptions, or even a lack of understanding on a studio's part about what a producer should really do, general producers seem to spend more time worrying about the game's direction (which should be the job of a director or equivalent), or they become a conduit through which marketing communicates its ideas to the team. This is often the role they are given, and it seems a mistake to me.

Producers should be solving problems. In a sense, they should be the team's internal Q/A—making sure that all the communication, scheduling, and dare I say production bugs get smoothed out in a timely manner. The job goes beyond ordering pizza for the team and updating the Microsoft Project file. The issue is clearly deeper than I can get into here, and I don't mean to suggest that producers are the root of these problems—but they should be the canaries in the coal mines alerting the team to these issues. With some proactive "bug" hunting on the part of a production team, many of these repeated, known problems can be avoided.

—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*

EDITORIAL
PUBLISHER
Simon Carless *scarless@gdmag.com*
EDITOR-IN-CHIEF
Brandon Sheffield *bsheffield@gdmag.com*
PRODUCTION EDITOR
Jeffrey Fleming *jfleming@gdmag.com*
ART DIRECTOR
Joseph Mitch *jmitch@gdmag.com*
SENIOR CONTRIBUTING EDITOR
Jill Duffy *jduffy@gdmag.com*
CONTRIBUTING EDITORS
Jesse Harlin *jharlin@gdmag.com*
Steve Theodore *stheodore@gdmag.com*
Noel Llopis *nllopis@gdmag.com*
Soren Johnson *sjohnson@gdmag.com*
Damion Schubert *dschubert@gdmag.com*
ADVISORY BOARD
Hal Barwood Designer-at-Large
Ellen Guon Beeman Microsoft
Brad Bulkley Neversoft
Clinton Keith High Moon Studios
Ryan Lesser Harmonix
Mark DeLoura GreenScreen

ADVERTISING SALES
MEDIA ACCOUNT MANAGER
John Malik Watson e: *jmwatson@think-services.com* t: 415.947.6224
GLOBAL SALES MANAGER, RECRUITMENT & EDUCATION
Aaron Murawski e: *amurawski@think-services.com* t: 415.947.6227
ACCOUNT MANAGER, EDUCATION AND RECRUITMENT
Gina Gross e: *ggross@think-services.com* t: 415.947.6241
SR. EVENTS ACCOUNT MANAGER, SOUTHWEST
Jasmin Davé
ACCOUNT MANAGER, WESTERN CANADA, INDIA, AUSTRALIA, & ASIA
Amanda Mae Miller

ADVERTISING PRODUCTION
PRODUCTION MANAGER Pete C. Scibilia e: *pscibili@ubm-us.com*

REPRINTS
PARS INTERNATIONAL
Joe Nunziata t: 212.221.9595 e: *reprints@parsintl.com*

THINK SERVICES
CEO THINK SERVICES Philip Chapnick
GROUP DIRECTOR Kathy Schoback
CREATIVE DIRECTOR Cliff Scorso

AUDIENCE DEVELOPMENT
GROUP DIRECTOR Kathy Henry e: *khenry@techinsights.com*
DIRECTOR Kristi Cunningham e: *kcunningham@techinsights.com*
LIST RENTAL Merit Direct LLC t: 914.368.1000

MARKETING
SERVICES MARKETING COORDINATOR Laura Robison e: *lrobison@think-services.com*

UBM TECHNOLOGY MANAGEMENT
CHIEF EXECUTIVE OFFICER David Levin
CHIEF OPERATING OFFICER Scott Mozarsky
CHIEF FINANCIAL OFFICER David Wein
CHIEF INFORMATION OFFICER Kevin Prinz
CORPORATE SENIOR VP SALES Anne Marie Miller
SENIOR VP, STRATEGIC DEV. AND BUSINESS ADMIN. Pat Nohilly
SENIOR VP, MANUFACTURING Marie Myers
SENIOR VP, COMMUNICATIONS NORTH AMERICA Alexandra Raine

United Business Media

WWW.TSGAMEGROUP.COM

design. build. run.

REVOLUTIONARY DEVELOPMENT
FOR A REVOLUTIONARY DEVICE
UNITY3D.COM

unity

GDC New Summits

AS GDC RAMPS UP FOR ITS MARCH DEBUT, MORE INFORMATION has been revealed about the new summits and side-events taking place at the seminal developer conference (which is run by Think Services, which also runs *Game Developer*). GDC this year runs from March 23–27, 2009, and takes place at the Moscone Center in San Francisco as part of Game Developers Conference. All summits take place at the beginning of the show, from March 23–24.

BIGGER BRAINS

New this year is the AI summit, which has announced initial speakers and sessions for the two-day artificial intelligence summit, including notables from EA Maxis, Ubisoft Montreal, Rockstar Leeds, Nintendo and more. The event promises to give attendees an inside look at key AI architectures and issues within successful commercial games.

Key speakers already announced for the Summit include EA Maxis' Soren Johnson, AI programmer/designer for SPORE and AI professionals from EA Montreal and Ubisoft Montreal, as part of a panel called "exploring ways to manage the gap between designers and AI programmers to help establish better practices for this important (and inevitable) collaboration."

Another notable lecture features Rockstar Leeds' Brett Laming, discussing "From the Ground Up: AI Architecture and Design Patterns," and focusing on the "multi-title, multi-genre architecture that now adds GTA CHINATOWN WARS to its history."

Other sessions include Crystal Dynamics' Daniel Kline alongside EALA's LMNO lead AI programmer Borut Pfeifer and others, discussing "practical approaches to pushing the boundaries of character AI, past successes and ideas for the future," plus former HALO 3 AI lead Damian Isla and MIT Media Lab's Peter Gorniak with a talk called "Beyond Behavior: An Introduction to Knowledge Representation."

The principal advisor and a speaker for the AI Summit is Nintendo's Steve Rabin, who explains of this inaugural summit: "What's truly exciting is that AI has the greatest potential of any technology to create brand new gameplay experiences and to broaden the market. With unparalleled AI industry experience, the AI Summit at GDC is an insider's look at how AI will impact the future of game development."

TOWER OF BABEL

The other major new summit this year concerns localization, which has gained increasing importance over the years. This is supported and organized by the IGDA Game Localization SIG, and is aimed at helping game professionals understand how to plan for localization in order to minimize bugs and maximize effectiveness.

Localization into America isn't the only concern, as the demand for entertainment software is coming from a growing number of countries and emerging markets around the world, which has prompted game publishers to partially or fully localize products to maximize their ROI.

The game industry is often generating as much, if not more, revenue from their localized versions. The old approach of simply translating a few phrases embedded in the game code at the end of the process can only devalue the title. Game localization is becoming an unavoidable, mission-critical aspect that must be fully integrated into game development life cycles.

THE MAGIC TOUCH

This year GDC Mobile is also part of the standard suite of summits, no longer a separate event as it was in years prior. Now developers can move freely within summits using a single summit pass, rather than buying specific passes for each.

This year's Mobile Summit has a focus on emerging platforms such as Apple's iPhone and Google's Android, and boasts initial speakers from EA and Indiagames. The program will cover six tracks encompassing the entirety of mobile gaming: New Platforms, Game Design, Programming, Deals & Distribution, Production, and Original Innovation.

The lectures announced so far include "The iPhone Bag Of Tricks," a hands-on session covering the day-to-day aspects of iPhone development, presented by G3 Studios CEO Guido Henkel, and "Social Games for Android, iPhone, Java, C++ and Objective C? Where do you fit in?" presented by Pick Up And Play president and CEO Paul Foster.

Other notable speakers announced for the event include EA Mobile Europe marketing director Tim Harrison, Indiagames founder and CEO Vishal Gondal, Amplified Games president and CEO Tom Hubina, and School of Interactive Computing at Georgia Tech associate professor Blair MacIntyre.

In addition, the IGF Mobile competition, which is giving out $30,000 to the most innovative independent mobile games, including a new $10,000 Best iPhone Game award, will have its awards during GDC Mobile once again this year.

For more information on all the GDC summits, which also include the Education Summit, Independent Games Summit, Game Outsourcing Summit, and Casual Games Summit, please visit http://gdconf.com/conference/summits.html

All summits take place from March 23rd to the 24th.

—Staff

PANDORA'S NEW BOX

AS OF ITS RELEASE, THE NEW PANDORA SHOULD BE THE MOST powerful commercially-produced handheld in the world. The console is the brainchild of the western GP2X community (an open source console from South Korean company Gamepark), which wanted to see more functionality from that machine, but decided it would be better to strike out on its own. The platform is fully open-source, and was created in large part off the back of forum suggestions.

The Pandora is created primarily as a homebrew and emulation machine, and has some pretty impressive specs. With Wi-Fi, touchscreen, dial analog and digital pads, and a full keypad, it seems, at least on paper, like a rather interesting piece of tech. Videos have already been shown of the console running QUAKE II, alongside various emulators and a full Linux desktop and office suite.

The console is pricey at $330, but even so the initial run of 3,000 units sold through immediately, with another run planned soon. Pandora's application for commercial game sales are basically nil, but as a hobbyist curiosity, it's pretty much tops.

—Brandon Sheffield

TECH SPECS:

- ◊ ARM Cortex-A8 600MHz+ CPU running Linux
- ◊ 430-MHz TMS320C64x+ DSP Core
- ◊ PowerVR SGX OpenGL 2.0 ES compliant 3D hardware
- ◊ 800x480 4.3" 16.7 million color touchscreen LCD
- ◊ Wi-Fi 802.11b/g, Bluetooth & High Speed USB 2.0 Host
- ◊ Dual SDHC card slots & S-Video TV output
- ◊ Dual Analog and Digital gaming controls
- ◊ 43 button QWERTY and numeric keypad
- ◊ Around 10+ hours battery life

STORY OR GAME?

IN THE LONG DESIGN DEBATE OVER narrative versus gameplay few genres are as polarizing as the visual novel. While the visual novel is primarily popular in Japan with little exposure in the West, its basic concepts are familiar to anyone who has read a *Choose Your Own Adventure* novel as a child. Visual novels offer essentially linear narratives that include opportunities for players to branch the story line resulting in a variety of endings. Largely text-based (though often enhanced with lush artwork and sound) and offering minimal interactivity, visual novels stretch the definition of what constitutes a "game" and can appear mystifying to players and designers steeped in the concepts of shooting and jumping. These games are created by and for a sub-niche audience and it doesn't help that their content can often come across to the uninitiated as self-referential and adolescent. Still, recent Japanese exports such as the ACE ATTORNEY series, HOTEL DUSK, and LOST ODYSSEY (for its dream sequences) have raised the profile of visual novels in the West, demonstrating that the form has many creative possibilities.

Developers who want to explore the potential of visual novel authoring can look to the fan translation group Insani (http://nscripter.insani.org/index.html) which is working on an English language SDK and documentation for Naoki Takahashi's NScripter engine, a commonly-used visual novel scripting tool. While its use has declined in recent years within the Japanese game industry, the Windows-based NScripter remains a powerful tool that has the advantage of being free for personal and commercial development. For a good example of its capabilities check out Insani's translation of stage-nana's visual novel NARCISSU at http://narcissu.insani.org/index.html. An open-source, English language version of NScripter called ONScripter-EN is also available at http://dev.haeleth.net/onscripter.shtml and enables both Windows and MacOS X users to run visual novels built with NScripter.

—Jeffrey Fleming

CALENDAR

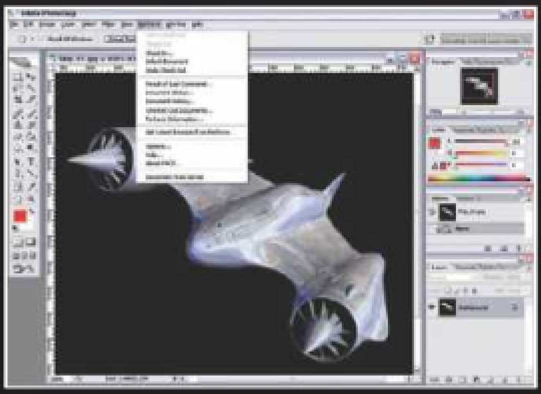
SXSW Interactive Festival
Austin, TX
March 13–17
Price: $325–1,145
http://2009.sxsw.com/interactive

Game Developers Conference
San Francisco, CA
March 23–27
Price: $850–2,200
www.gdconf.com

Perforce | *Fast* Software Configuration Management

Introducing P4GT,

a productivity feature of Perforce SCM.



P4GT

The Perforce Plug-in for Graphical **Tools,** P4GT, makes version **control** painless by **seamlessly integrating Perforce** with **leading graphical tools. Drop-down menus allow access to Perforce from within 3ds Max, Maya, Softimage XSI, and Adobe Photoshop.**

Art and development teams can standardize on Perforce to version and manage both source code and digital assets. Enhanced collaboration during the design process helps teams to work together in real time to release small patches or create whole new worlds.

P4GT is just one of the many productivity tools that comes with the Perforce SCM System.

PERFORCE SOFTWARE

Download a free copy of Perforce, no questions asked, from www.perforce.com. Free technical support is available throughout your evaluation.

All trademarks and registered trademarks are property of their respective owners. Adobe screen shot reprinted with permission from Adobe Systems Incorporated.

GAME DEVELOPER FRONT LINE AWARDS 08

>> jeffrey fleming

FOR MORE THAN A DECADE *GAME DEVELOPER* HAS BEEN SINGLING OUT THE BEST IN game development tools with our annual Front Line Awards. Here we take a moment to honor the products that are most effective at helping developers do great work. Whether in the trenches or on the cutting edge, these are the tools that artists, designers, and engineers rely on. While the tool business can be hotly competitive with the next greatest thing always just around the corner, we also want to pay special tribute with our Hall of Fame Award to the product that has made a lasting impact on the game industry year after year (and which was also not eligible to win in its specific category).

Nominations for this year's Front Line Awards were open to all new software products and new versions of software products related to game development released between September 1, 2007 and August 31, 2008. In determining the winners of the 2008 Front Line Awards we took a different approach than in previous years. After a period of open nominations in October we collected the results and combined them with our own picks to come up with a list that we then narrowed down to five finalists in each category. The result was a mix of focused solutions along with more general applications. We then handed the finalists over to you, the readers of *Game Developer*, via an invitational online survey in November, so that you could have a voice in picking the recipients of the Front Line Awards. We gathered over two thousand responses to the survey and we are proud to present the winners along with commentary by developers from throughout the game industry. A special thanks goes out to everyone who contributed to this year's Front Line Awards and congratulations to all the finalists and winners.

ADDITIONAL ILLUSTRATION BY MOBY FRANCKE

HALL OF FAME

UNREAL ENGINE

Epic Games

www.unrealtechnology.com

EPIC HAS BEEN PRETTY CLEAR ABOUT ITS message—use our engine, and you will save time and money. Not just that, but you'll be able to prototype faster, iterate more, and get more support. Unreal Engine 3 seems to have delivered on that promise for the company, as consumer web site Kotaku was heard to joke "I fully expect to hear that my mom has licensed Unreal Engine 3 any day now."

After three consecutive years of winning the Engine category of the Front Line Awards, Epic Games' Unreal Engine has become the Front Line Awards 2008 Hall of Fame winner. Since the release of Unreal in 1998, Epic's Unreal Engine has helped set the technological bar for the video game industry. With the release of Unreal Engine 3 in 2005, Epic helped a host of developers make the transition to the current generation of consoles.

The Unreal Engine stands out in part due to its robust and thorough toolset. For shooters, real-time strategy games, or even sports titles, teams have been able to create games without authoring substantial core technology. Cutscenes and camera work are created and previewed in-engine by the Matinee cinematics system, and teams are provided with a visual scripting system (Kismet), as well as the UnrealScript programming language. The Cascade particle system editor offers effects artists a huge array of parameters to control. Layered across all these tools is the renderer that made the Unreal Engine famous. The materials, lighting, and shadowing systems are all seamlessly integrated into each of these tools. For example, post-processing effects in Matinee-authored cutscenes are created using the same material editor that lets artists create the surface effects for water. All these tools are bundled into a single application called UnrealEd. Add these tools together, and you've got an engine that would take even the best of teams years to create on their own. Not insignificantly, these tools provide cross-platform game content enabling developers to reach as many customers as possible.

Use of Unreal Engine 3 is now pervasive in the game industry, and the engine has basically become the defacto standard for game development. Art outsourcing partners are increasingly familiar with the technology and are able to test and preview their work directly in-engine, cutting down on iteration time. Furthermore, thanks to Epic's Integrated Partners Program, middleware like Scaleform's GFx and RAD's Bink Video come ready to drop in and use. This standardization also applies to co-development partners. Many teams like ours which have now been working with the technology for years can hop into projects with a full team, or join a project mid-stream thanks to familiar tools.

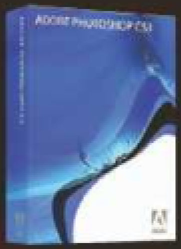
GAME DEVELOPER FRONT LINE AWARDS 08

Epic is also renowned for its hands-on customer support. The Unreal Engine 3 mailing lists get hundreds of posts per week from developers discussing their use of the engine along with the engineers, designers and artists at Epic who are expert users of the technology. When a licensee asks a question about a section of code, the actual engineer responsible for that code will frequently be the one to answer back.

You need only look at the titles that have shipped using Unreal Engine 3 to full appreciate the breadth of impact it has had on the gaming industry. Whether it is ground-breaking work like BIOSHOCK or MIRROR'S EDGE or franchises with rich histories like MEDAL OF HONOR or SPLINTER CELL, Unreal Engine 3 proves time and again that it has everything a team needs to ship a successful AAA title.

We are experiencing a renaissance in our industry that can be partially attributed to being able to focus on design rather than the hurtles of technology. For many developers, that is possible because of Unreal Engine 3.

—Albert Reed, Demiurge Studios



ART TOOL

PHOTOSHOP CS3

Adobe www.adobe.com

ART TOOL FINALISTS
Autodesk 3ds Max 2009
Autodesk Maya 2008
Luxology Modo 302
Softimage XSI 7

IN THE 1999 FILM *THE MATRIX*, WHEN NEO reached for the red pill Morpheus warned, "Remember ... all I'm offering is the truth. Nothing more."

If you've been using Adobe Photoshop CS3, then you've been using the red pill, because that was the code name for the software during its development. And for many people who chose it for the first time or upgraded to CS3 from whatever they were using, it was indeed the truth. While the new version was marketed with the phrase: "Work more productively, Edit with unrivaled power, and Composite with breakthrough tools," (phraseology that would make the Wachowski brothers positively weak in the knees with envy) in reality Adobe's product marketing wasn't Hollywood hooey. In fact, if you collect photo reference or are interested in extending your 3D modeling to 2D applications, you might find it very much the real deal.

For instance, think of CS3's Auto Align as Photomerge on 'roids. While Photomerge can stitch and blend a panorama together by aligning elements within your photos, Auto Align scans through the layers of the current document and calculates which pixels match with the proper layer, and transforms each layer appropriately to achieve astonishing results. If you're taking photo reference for a video game, for instance, gone are the days where you snap scads of freehand photos of a location and align them by hand (or worse yet stitch them together in your head). With Auto Align you can, for example, take multiple photographs walking along a street (sans tripod, of course), and Auto Align does all the heavy lifting to put them together. Use Auto Blend on the results and your tableau becomes a seamless construct.

For anyone using sliders to adjust Raw parameters within Adobe Bridge, the performance in CS3 is quite speedy, and in addition to the basic Contrast, Brightness, Shadows, and Saturation sliders found in CS2, CS3 adds Clarity, Vibrance, Recovery, and Fill Light. Again, video game pros collecting photo reference will appreciate the flexibility.

Last, but certainly not least, Adobe Photoshop CS3 enables 3D objects to be included in special layers and manipulated therein. Once loaded, you can spin, scale, and move the model; the camera can also be moved around the 3D object. All of the model's textures can be edited since they're all found on individual "child layers" beneath the main model layer. Changes are immediately shown.

While it may be carrying the analogy too far, if you stayed with Adobe's Photoshop CS2 (or continued using some other package), and then chose the blue pill, then everything stayed the same for you. While CS3 is far from perfect, it makes a great addition to Photoshop's lineage and provides powerful tools for video game artists and animators.

—Tom Carroll

AUDIO TOOL

FMOD

Firelight Technologies Pty, Ltd www.fmod.org

AUDIO TOOL FINALISTS
Vivox Precision Studio v. 2
Audiokinetic Wwise 2008.3
Rad Game Tools Miles Sound System 7.2c
Di-O-Matic Voice-O-Matic 2.6

IN AUDIO DEVELOPMENT, WE DO OUR BEST TO assess the needs of the project, establish a working methodology, and lay the groundwork before suiting up and heading in to battle. The development war wages on at a breakneck pace, shooting at moving targets until everything is in full swing, and just as we're hitting the beachhead, ammo is running low. The best we can hope for is the minimized stress of well-implemented systems and creative user workflow.

It's times like these when the power of established audio middleware comes in to save the day. One of the leaders of the pack is Firelight Technologies' FMOD Audio Engine and FMOD Designer Tool. Though the Designer Tool is the winner here, the combination is what delivers a total interactive audio solution that addresses the burgeoning specialization of audio integration in today's highly technical world of game development.

After the initial integration with the game engine, most of the work involved with creating dynamic, multi-layered sound is left in the hands of the audio team. FMOD's comprehensive Event system manages everything from volume, pitch, and positioning to more complex functionality such as playback behavior, distance attenuation, and falloff curves. Having this level of control allows the audio team to tailor the way things sound, and make sure that the sound content is played back in the best way possible.

With additional programming support, values and parameters from the game engine can be passed to the FMOD Audio Engine and used within the Designer Tool. This functionality enables a level of dynamic flexibility over an event allowing you the ability to parametrically define what sounds are played back, dynamically effect the sounds through the use of DSP, and trigger multiple sounds over time. Parameters can be leveraged in order to better mirror the reaction of sound in realistic ways, or take them further away from reality and infuse them with a higher level of interactivity driven by gameplay.

Imagine using a parameter for player distance to change the directivity of a fireplace within a 3D space, making it more directional the closer you get to it and alternately distributing the sound of it throughout the available channels when further away. Additionally one could tie a 24 hour clock in the game world to a looping ambient event that would change the types of sounds played based on the time of day. Adjusting the speed of a flange based on distance could add a dynamic effect to the sound of an approaching ghost, making the sound content vibrate at a greater rate when closer to the player. Or you can trigger an event that sets off a time-based chain reaction of sounds, allowing for a type of interactive sound design within the tool.

The FMOD Engine and Designer Tool is a magic box full of functionality that allows the audio team to make creative decisions about how things sound. As interactive audio specialists, we are continually dreaming of new ways to enhance the sound experience for our audience. With FMOD, we can realize some of those dreams.

—Damian Kastbauer, Bay Area Sound

MIDDLEWARE

HAVOK PHYSICS

Havok **www.havok.com**

MIDDLEWARE FINALISTS
Autodesk Kynapse 5
PathEngine 5.16
NaturalMotion Euphoria
GameSpy SDK

LET'S FACE IT: AS THE COMPUTATIONAL POWER of gaming hardware continues to increase, so too do consumer expectations. Demand continues to rise for game experiences presented within the context of believable virtual spaces: living worlds that are huge, interactive, persistent, and "real."

While realism can be a loaded word for game developers, there is no question that the inclusion of advanced real-time physics has become a necessity for many of today's game engines. And while team sizes continue to balloon to meet gamer demands, physics is not necessarily an area that teams want to throw internal resources at, all things considered.

Enter Havok. At its core, Havok offers developers the foundation for creating realistic physics simulation though efficient and optimized calculations. This package allows games to simulate a lot of complexity simultaneously without a serious hit on performance, and provides a framework that's easily adaptable to both online and offline games. This solid codebase continues to make Havok a prime choice for physics middleware across the industry.

If that were all Havok offered, Havok Physics would be just another middleware choice among qualified competitors. What sets Havok apart are the layers of developer support that the company has integrated into all aspects of its product and services. Havok understands what it means to be a middleware company, and what it means for middleware to bring real value to a game's development cycle.

Real value is a company that delivers a solid middleware solution and continues to update it with developer-requested features. It's a mindset of getting to know your clients' products and their product goals, and aligning middleware roadmaps toward providing them with their most-important features wherever possible. It's providing responsive and thorough developer support when bugs do arise, and a dedication to providing solutions quickly and efficiently. It's continued expansion of Havok Physics' integrated product set, with new modules like Havok Behavior and Havok Destruction seamlessly integrating with the core Physics SDK; modules created based on cross-discipline developer feedback, and expanded and refined with an eye towards versatility and ease-of-use.

It is this real value that Havok Physics brings to developers with its bottom-up methodology, an optimized and integrated codebase, and dedication to supporting the creation of more believable virtual worlds. It stands out among some tough competition as winner of this year's Front Line Award for Middleware.

—Jeremy Gordon and Michael Boccieri, Sega Studios

GAME DEVELOPER FRONT LINE AWARDS 08

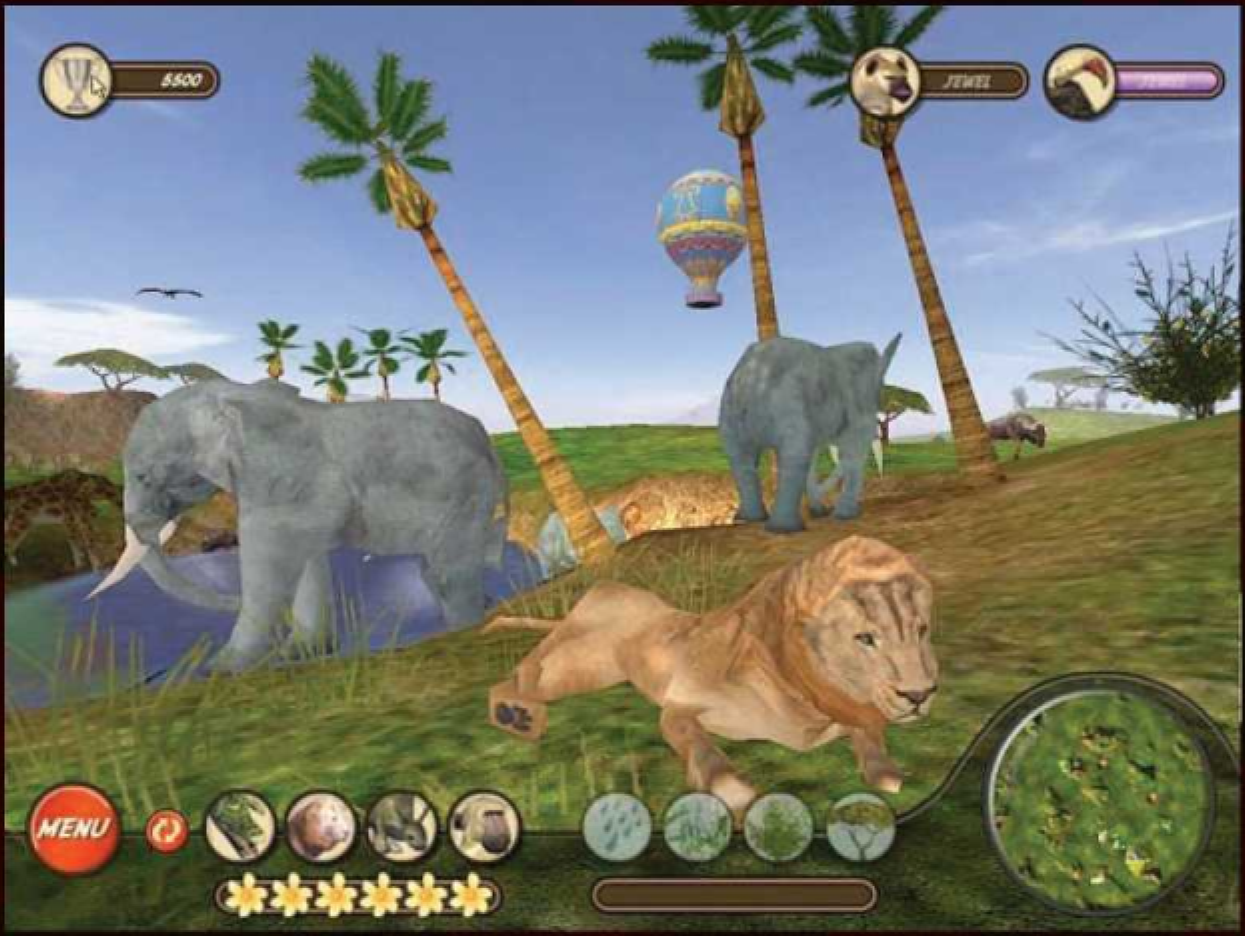
All images and characters are © and TM their respective holders. All rights reserved.

ENGINE

TORQUE GAME ENGINE ADVANCED 1.7.1

GarageGames www.garagegames.com

ENGINE FINALISTS
Unity Technologies Unity 2.1
Crytek CryEngine 2
Emergent Game Technologies Gamebryo 2.5
Valve Corporation Source Protocol 14, Build 3531



I'D LIKE TO CONGRATULATE GARAGEGAMES for winning a Front Line Award for its flagship game engine, Torque Game Engine Advanced. I've personally used the Torque line of products since I quit my mainstream game programming job four years ago and went indie. Over that time, Torque has evolved from a somewhat clunky and underpowered 3D engine to a framework that supports the Xbox 360, Wii, PC, Mac, and iPhone and sports many of the bells and whistles that make Unreal and other high-end engines cost a million dollars. But in stark contrast to those high-end engines, TGEA costs under 300 dollars. And over the past four years the engine has evolved: it's not just for indies anymore.

Originally, the founders of GarageGames had the vision that if they could create a low-cost 3D engine, the world would explode with revolutionary games made by hobbyist game developers. My first indie game, VENTURE AFRICA, sold over 100,000 copies, which made it one of the very few to fulfill this vision. Unfortunately, it turned out that the engine was too feature-rich and obtuse for most hobbyist developers to handle. So GarageGames shifted its focus to building a 2D engine in order to compete in the casual space. Again, the bulk of the engine made it compare unfavorably to other simpler engines out there in the 2D space.

But the rise of casual gaming on the consoles finally created the perfect niche for the GarageGames product line. Rather than focusing on hobbyist developers with their heads in the clouds, GarageGames began to incubate slightly larger studios with visions grounded in reality. The company consolidated and componentized its codebase, offering the features of all of its products in a single engine. The engine was then ported to the Wii, the Xbox 360, and the iPhone. Torque Game Engine Advanced became an incredibly versatile tool with a competitive feature set for small to mid-size developers while remaining dirt-cheap.

GarageGames is now focused on helping developers with a high potential for success, rather than on the myriad dreamers hoping to make their first game into the next WOW. Penny Arcade's PRECIPICE OF DARKNESS series is a good example of what a relatively small team can do with the engine.

These days, TGEA offers complete source code, DirectX and OpenGL support, and Xbox360, Wii, iPhone, PC, and Mac support for only $295 ($1495 for commercial developers). The multiplayer framework is still one of the best in the business, and GarageGames continues to improve the WYSIWYG game and GUI editors. Now that some successful titles are beginning to prove the potential of the engine, it's impossible not to see Torque Game Engine Advanced as an emerging player in the field of major game engines.

Over the past two years, indie games like PORTAL, BRAID, and WORLD OF GOO have made their mark on the mainstream game industry. It's finally come time to recognize a piece of indie technology that has broken through to compete with the major engines. Congratulations GarageGames, Torque has finally realized its massive potential.

— Andy Schatz, Pocketwatch Games

THINK SERVICES
GAME GROUP

2009 EVENTS CALENDAR

MARCH 23–27, 2009

GDC 09

Game Developers Conference®
Moscone Center,
San Francisco, CA
www.gdconf.com

MARCH 27, 2009

GAME CAREERSEMINAR

Game Career Seminar at GDC09
Moscone Center, San Francisco, CA
www.gamecareerseminar.com

MAY 12–13, 2009

GDC Canada

Game Developers Conference® Canada
Vancouver Convention and Exhibition Centre,
British Columbia
www.gdc-canada.com

AUGUST 17–19, 2009

GDC 09 Europe

Game Developers Conference® Europe
Cologne, Germany
www.gdceurope.com

SEPTEMBER 14–18, 2009

Austin GameDevelopers Conference

Austin Game Developers Conference®
Austin Convention Center, Austin, TX
www.austingdc.net

OCTOBER 11–13, 2009

GDC 09 China

Game Developers Conference® China
Shanghai International Convention Center
Shanghai, China
www.china.gdconf.com

FOR UPDATES AND MORE INFORMATION ON OUR EVENTS VISIT:

www.tsgamegroup.com

THINK SERVICES
A DIVISION OF UNITED BUSINESS MEDIA LLC

PROGRAMMING TOOL

VISUAL STUDIO 2008

Microsoft http://msdn.microsoft.com

PROGRAMMING TOOL FINALISTS
Seapine Software TestTrack Pro 2008.2
Perforce 2008.1
Microsoft XNA Game Studio 2.0
CollabNet Subversion 1.5

VISUAL STUDIO IS, WITHOUT A DOUBT, THE most-used programming tool in the games industry. Used for development of Xbox and Windows games, creation of Windows tools in C#, web development, and sometimes even just as a powerful editor, Visual Studio is the go-to tool for most programmers.

But Visual Studio has had a rocky history with game developers. From the solid Visual C++ 6.0, to the best-forgotten Visual Studio .Net (2002), it has had its highs and lows. The recent Visual Studio 2005 was the best of the series and it was an absolute pleasure to work with. The bar was set very high for a follow-up.

Is Visual Studio 2008 a worthy successor? Definitely. Microsoft wisely followed the formula of not fixing what ain't broken, and made 2008 an incremental improvement over 2005, along with a few, big new features.

On the C++ side of things, the improvements are very subtle. It looks so much the same that you might forget you're using a new version. You need to look under the hood to find some of the new features: better multiprocessor build support, managed incremental builds, and improved multithreading debugging support among others. My only wish is that Visual Studio 2008 supported the C99 standard, but alas, that's not to be.

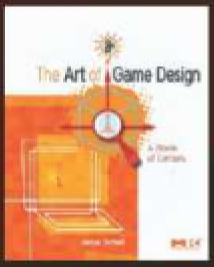
Programmers are in for a real treat with the new features on the C# side. Visual Studio 2008 introduces C# 3.0 with all the goodies: Automatic properties, initializers, anonymous types, and lambda expressions. It also comes with support for Language-Integrated Query (LINQ), the new API to perform SQL-like queries and operations on different sets of data. And if all that is not enough, 2008 also includes a visual design tool for Windows Presentation Foundation GUI tools, which should make most tools programmers feel like Santa just came to town.

Microsoft deserves props for continuing in the tradition of previous versions, and offering a set of Express editions that can be downloaded for free. These editions have slightly limited functionality, but they're a great way to allow students and the online community to create games with the Visual Studio.

For professional developers in large teams, a big issue with any new version is how painful the upgrade is going to be, and how much down time it is going to cause. Again, only good news in that front: The upgrade was seamless, and I was up and running in no time. Both 2008 and 2005 can live side by side, and you have the option to import your settings from earlier versions. It even brought over Add-Ins from 2005 and they worked flawlessly under 2008.

Visual Studio 2008 manages to maintain the quality and robustness of 2005, while introducing some significant features for tools programmers. It should be in every programmer's workstation and is a most deserving winner of the 2008 Front Line Awards.

—Noel Llopis



BOOK

THE ART OF GAME DESIGN

by Jesse Schell

Morgan Kaufmann www.elsevierdirect.com

BOOK FINALISTS
Dungeons and Desktops: The History of Computer Role-playing Games
by Matt Barton
Game Production Handbook, 2nd Edition
by Heather Maxwell Chandler
Real-Time Rendering, Third Edition
by Akenine-Moller, Haines, and Hoffman
Game Programming Gems 7
by Scott Jacobs

THERE ARE HUNDREDS OF BOOKS WRITTEN that discuss computer game design. But only a very small handful of those were written by professional game designers. And although many of them are intended as possible textbooks for courses about the subject, very few have been written by successful teachers of game design. So the fact that Jesse Schell is one of the dozen or so full-time professors who have had a successful career in game design would be reason enough to be excited about his book.

But it goes far beyond that. This book was clearly designed, not just written, and is an entire course in how to be a game designer. The very structure of the book serves as an object lesson in interactive design. Its subtitle "a book of lenses" is not just a promotional phrase, but rather says a lot about how Jesse thinks about the meta-process of game design. He uses the term lens metaphorically, as a way of looking at a game and asking questions to help analyze it. He includes 100 lenses he uses in the book, with names like Fun, Surprise, Flow, Chance, Emergence, Puzzle, Beauty, Client, Pitch, Technology, and many more. This philosophical approach captures in concrete form an intangible truth about the essence of game design. A good game designer has to wear many hats—but that's a tired and obsolescent metaphor. Looking at the world—or at the game you are designing—through different lenses is a much fresher, more accurate one.

That perspective frees Jesse to scan the landscape of game design, scrutinizing aspects of game development, creativity, psychology, technology, and many other related fields. He uses skillfully-written, relevant vignettes about what designers must learn in order to practice their art. He also approaches each aspect as a good designer would, providing not just bare facts but also captivating stories to set the mood and provide context, and adds charts, tables, drawings, and cartoons to capture the essence of his subjects and to illustrate them, figuratively and literally, from many different perspectives.

The book is also intensely practical, giving some of the best advice on how to harness your own subconscious I've ever read, as well as short and useful descriptions of probability theory for non-mathematicians, how to diagram interest curves, working with a team, and dozens of other topics. It is simply the best text I've seen that really addresses what a designer should know, and then actually gives practical advice about how to gain that knowledge through life experience. It's a marvelous tour de force, and an essential part of anyone's game design library.

—Noah Falstein

www.seapine.com/gamedev
Satisfy your quality obsession.

Full-Time Quality Assurance Manager–**Immediate Opening**

Don't work yourself to death. Use TestTrack® TCM to manage the testing of your next title.

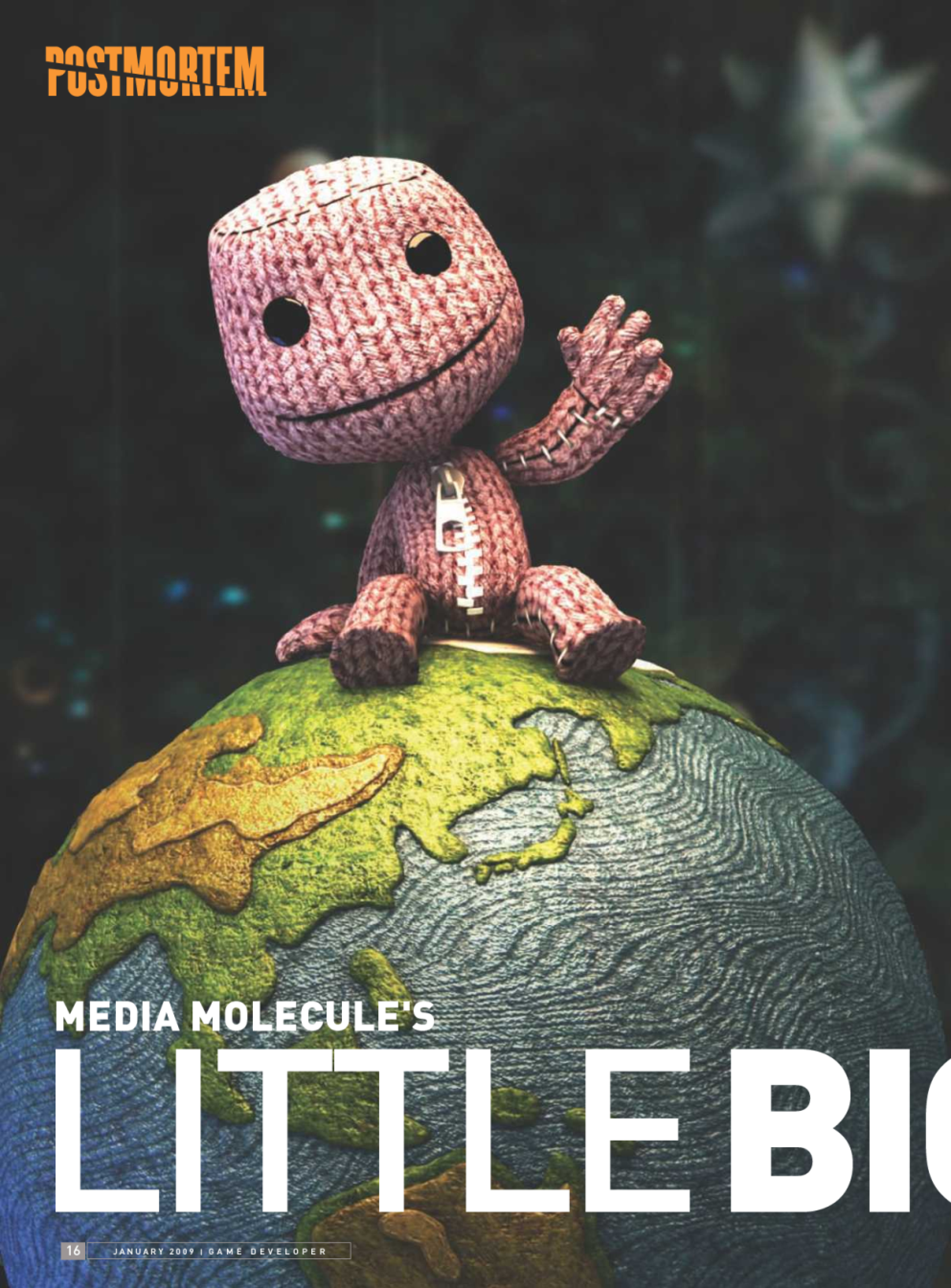
TestTrack TCM puts you in control of test case planning and tracking, providing better visibility over the testing effort and giving you more time to manage your team. With TestTrack TCM your team can write and manage thousands of test cases, select sets of tests to run against builds, and process the pass/fail results using your development workflow.

- Know instantly which test cases have been executed, what your coverage is, and how much testing remains.
- Manage suites of platform-specific compliance tests, functional tests, and performance tests in one central location.
- Assign tests to your QA team, track results, and report on performance and workload.
- Use test variants to target multiple platforms with the same test case for more efficient test case management.
- Streamline the QA > Fix > Re-test cycle by pushing test failures immediately into the defect management workflow.
- Achieve complete traceability between test cases and defects with seamless TestTrack Pro integration.
- Ensure all steps are executed, and in the same order, for more consistent testing.

Seapine Software™

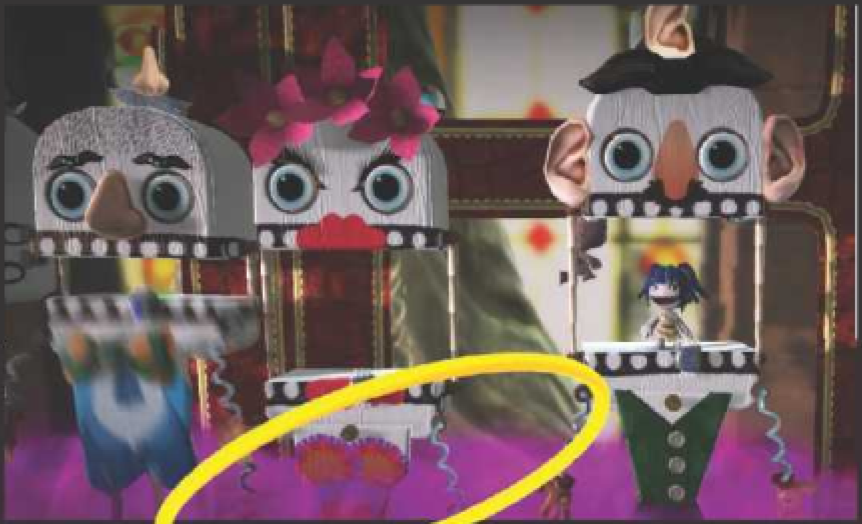
© 2009 Seapine Software, Inc. All rights reserved.

TestTrack® TCM Test Case Management
QA Wizard® Pro Automated Testing
Seapine CM™ Change Management
Surround SCM® Configuration Management
TestTrack® Studio Test Planning & Tracking
TestTrack® TCM Test Case Management
TestTrack® Pro Issue Management



MEDIA MOLECULE'S

LITTLE BI



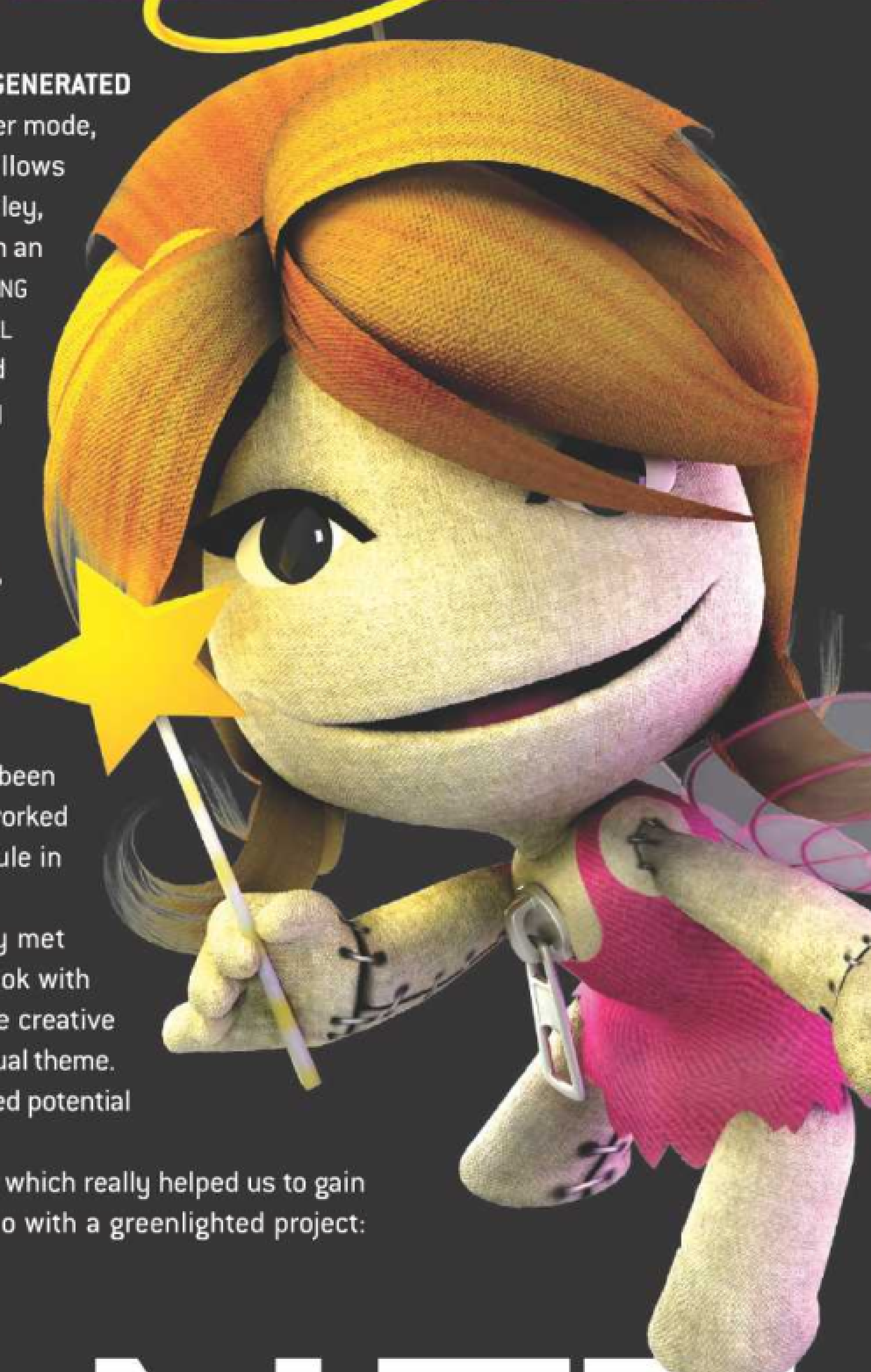
LITTLEBIGPLANET IS MEDIA MOLECULE'S EXPERIMENT IN USER-GENERATED content on consoles, which in addition to offering a single-player mode, takes the skeleton of a multiplayer platforming game, and allows players to add the flesh. Before forming Media Molecule, Mark Healey, Alex Evans, Dave Smith, and Kareem Ettouney worked together on an independent game called RAG DOLL KUNG FU while at Lionhead Studios. RAG DOLL KUNG FU was enthusiastically embraced by the gaming community, especially for its character customization. The team agreed that user-generated content was an exciting area to push forward in a professional project. In January 2006, Healey and several of his colleagues founded Media Molecule.

SIOBHAN REDDY *is the executive producer of LITTLEBIGPLANET. She joined Media Molecule in 2006 after working for seven years at Criterion Games. Email her at* **sreddy@gdmag.com**.

Everyone but Chris Lee, our commercial director, had worked together at Lionhead for many years. Lee, on the other hand, had been vice president of sales and marketing at Criterion Software; I had worked at Criterion Games for seven years before joining Media Molecule in March 2006 to be the executive producer of LITTLEBIGPLANET.

After starting Media Molecule, key members of the company met with Phil Harrison at Sony, as a sort of pitch of talent. They took with them a physics demo, the RAG DOLL story, their ideas for online creative collaboration, as well as a fondness for a craft material-based visual theme. Harrison picked up on the user-generated online ideas and noticed potential in this particular aspect. He pushed the pitch ahead.

Sony and Media Molecule were perfectly aligned at this point, which really helped us to gain momentum. Very quickly, the group turned into a small studio with a greenlighted project: LITTLEBIGPLANET, or as we called it at the time *The Next Big Thing*.



GPLANET

/// WHAT WENT RIGHT

1 SMALL TEAM WITH AREA EXPERTS. One of the first things we did when developing LITTLEBIGPLANET was to set some goals for the studio. Going through this process cemented what we wanted to do: build a small studio of talented and creative people whose focus is to create a genre-defining console game that would be both commercially and critically successful.

LittleBigPlanet was designed as a game that could be made within our means. We weren't making a game that would need enormous art assets like a first-person shooter or RPG driving game. Being able to strike out in a new genre (or an old neglected genre, depending on your point of view) meant we could focus on making the game we wanted to make, and not waste effort by slavishly copying the standard feature set that might be expected of a more typical mass selling game.

Concept art for the "made look."

We decided early on that we wouldn't grow the company larger than 20 people—however, we inflated to 31 once we realized the ambition of the project. We also decided that we would assign a large area of responsibility to each person. For example, one person would be focused on the game engine while another person would focus on physics, or production, or character animation.

Paradoxically, we also wanted everyone on the team to be able to have input on every other aspect of the game. In this way, the different areas of the game would feel more integrated as level designers, artists, and programmers sometimes switched jobs for a few hours a day. The communication overhead needed to support this practice wouldn't have worked with a large team.

2 GAME JAM DESIGN STYLE. During the early days, the process went something like this: Conversations would take place that would then lead to some design work being done. Sometimes the design wouldn't stick, but other times it would, unlocking a whole new host of possibilities. Our natural process, maybe because of the high number of musicians on the team, was for people to riff off each other. We referred to these discussions as "jamming." We didn't start with a design document, but we did lock down areas to try out. The flow that had been discussed was eventually visualized by Healey, and this became the framework for the greenlight and actually the whole game.

Once the framework was set, everyone was free to go wild within his or her area. We gave people freedom to design and prototype in their own way. We tended not to spend a lot of time in design meetings because people were (and are) always itching to get out of them and actually try stuff.

We met regularly to make sure everyone was still going in the same direction or to stir up excitement by sharing at a new direction we wanted to try out. We experimented with, and eventually shelved, many ideas during this early period.

Once we had gotten to the stage (post greenlight) where the concept crystallized, Healey consolidated all the various ideas into a design document to share with the team. This document wasn't kept up to date daily, but a full consolidation was done at key points to make sure everyone was behind it.

In the periods between consolidations, targeted designs would be written up outside of the main document. During the final phase, we moved toward one-on-one discussions and emails, which sometimes led to a lack of cohesion or accountability, though it's perhaps because of the small size of the initial team that we got away with it anyhow.

Ultimately, most of the knotty design decisions at the start of the project were resolved by rapidly trying out our ideas via code or pre-visualization. We went down a fair number of design dead-ends, but even the failures gave us useful information. This tactic wouldn't have been very efficient if we had a large standing army of artists and level designers waiting for the design to be finished.

Now that we're a larger team, we have retained this approach. It's a little hard at times, but with the fundamental idea of a framework being set by the directors, the owners of specific areas of the game are trusted to create great things (and encouraged to try out hard, exciting, or unusual ideas). We regularly review how our projects are coming together. The fact that design documents often come together after you have tried something is a very positive notion in our company culture.

3 POP-IT'S ACCESSIBILITY. Pop-it is the name of the in-game editor. There was a time when pop-it was not a single editor but was a set of tools, a paintbrush for painting, a hair dryer gun to blast away material to sculpt, a decoration placement tool, a paint scrubber, and so on.

We had the project greenlighted with pop-it like this, but afterward, it was one of the first areas to evolve. Initially, we didn't know how versatile the game was going to be or how we would feel about the editor, but as time went on and the concept became more concrete, we became more confident that people would want a more focused editing experience that prioritized functionality over "cuteness."

We quite literally went back to the drawing board. Our new design became a pre-visualization video, which we later tried out in the game.

Making this decision marked a shift in pop-it in that we now expected people to be able to create bigger and more complex things. Between December and May (2007) the pop-it design was worked through until we rested upon the "stamping" concept we finally shipped with. Back then, this was all happening at the

The Media Molecule team at their first Christmas party.

same time that the team was working out what they wanted to make! The switches (basically cause-and-effect nodes that the player can use to construct mechanisms) were the last big feature to be added into creation toolset, and this unlocked the game design enormously.

Creating pop-it was a massive job, and one that had constantly moving goal posts. Amazingly, it was primarily coded by one man, Jonny Hopper.

4 GROWING WHILE FINISHING THE GAME. One of the unique aspects of Media Molecule and LITTLEBIGPLANET is that we were right at the start of building a studio when the game was announced at GDC 2007. At that point, we only had 11 developers on the team. We now have 31 with a healthy combination of talent, humor, ego, work ethic, and loveliness.

The binding factor at Media Molecule was our experience not as professional games makers, but as people who have a shared interest in creating or consuming innovative media and a DIY/home-made hippie-ish approach to life, and very often both.

Talent is obviously crucial, as each individual is expected to develop and deliver an entire area of the game. To this end, we have a probation period, during which the new recruit must demonstrate that he or she can deliver quality work within a given area. The idea is that once you have achieved this, you are trusted to get on with delivering in that area with less management.

5 MOLECULAR STRUCTURE. A key part of our company culture is to acknowledge that the production plan doesn't belong to the production team—it belongs to the creator of the work. Each member must make his or her plan visible to everyone else so that we all know what's going on and can understand how it affects other areas of the game.

Once the team grew to be more than 10 people, we started using a structure that we call "molecules," which was inspired by Valve's cabal structure and our own experiences. We agreed that our workforce should be organized into small—and ideally cross-pollinated—molecules: pop-it, character, levels, online technology, and player experience, while our audio designer was included in every molecule as well.

Members changed over time, depending on who was working on what. Some people were in two or three molecules at a time, and each molecule works slightly differently, with the only requirement being that their plans need to be visible and shared.

Early on we established something we called Friday features. On Fridays, people show the work they created during the week. This allows the whole team to see what's going on and helps us end each workweek on a high note.

We built a good relationship with Sony, and while producers would come down for milestone presentations, we also delivered videos of our progress via a hand-cam walkthrough of the office, showing work that had taken place for this or that feature. In retrospect, what we created was an amazing record of the start of the studio. The filming stopped when everything we were showing was in the game, but we will be filming again in the future for sure.

At the start of making a game, features very clearly diverge into aspects that look and play well, and aspects that are still in development. It was important to us that our contacts at Sony saw both the sexy work and the unsexy stuff so we could communicate why we were spending time where we were.

We also did well to not build demos for every single tradeshow and conference. Instead, we used shows as a focal point for us to progress the game. For example, E3 (2007) was to be the first time we showed the creative tools live, so we used the show date as a deadline to get pop-it ready to be shown. We added some levels the team made in literally a couple of hours one afternoon to show. It was rough around the edges, but all the areas that introduced people to pop-it and the simplicity of the tools were there without distracting the team on throwaway work. We never got into the trap of having a demo without having a game, and we purposely made the game our demo.

/// WHAT WENT WRONG

1 FINDING GREAT PEOPLE IS HARD! We all had a pretty steep learning curve! This was the first business that the directors had started together, and the first where I was executive producer.

Keeping the studio small meant we had to make sure we had the right team. While all the people we hired were lovely and

GAME DATA

DEVELOPER	Media Molecule
PUBLISHER	Sony Computer Entertainment
RELEASE DATE	November 5, 2008 (U.K.)
PLATFORM	PlayStation 3
NUMBER OF FULL-TIME DEVELOPERS	27 internal, 10 Sony
DEVELOPMENT SOFTWARE USED	3ds Max, Adobe Photoshop, Fmod, SN tools, Visual studio 2005, Ccnet, Subversion, Microsoft office, Microsoft project, Devtrack, Trac, Adobe Premiere, Pixologic ZBrush, SCERT
LINES OF CODE	527,033

talented, we had to make sure we didn't just have a random collection of talented people. We needed a team.

In the beginning, hiring was really difficult, especially because we hadn't announced the game. After GDC, when the game was announced, hiring was still difficult and very frustrating. The areas we were desperate for were people to create tools, gameplay, and online features, and it took us a long time to fill those roles with the right people.

Hiring an experienced games HR manager fairly early on helped. We also signed up with multiple recruiters, though the majority of our team approached us directly. We are currently hiring for some roles that will help spread the load in the future.

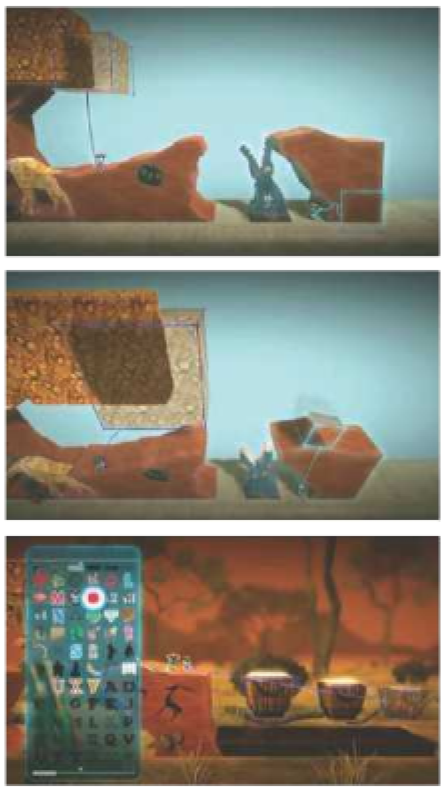
2 BUILDING A STUDIO AND MAKING A GAME AT THE SAME TIME. Our focus was often split between building the game and establishing the studio.

We recently conducted some team surveys and learned that there were a lot of things that worked well enough when we were a very small team, but broke down as we became bigger and busier. These issues ranged from the trivial to the serious: employees not knowing who to get direction from, starting meetings on time, making visible the reasons people failed to come to work, office space not being used efficiently, inconsistency in work-life balance—the list could go on!

There is an expectation from the team, and a good one, that every area of the company be the best it can be. We underestimated the amount of support we needed to ship LITTLEBIGPLANET and run the company to its best simultaneously.

3 SIX-MONTH CRUNCH. We had to crunch for six months in order to get the game out on time. One of our goals was to release the game in 2008. We had always estimated September or October. Shipping on time became a central focus for everyone, and we did our best to not let it affect quality.

When we did the first set of player reviews at alpha, reactions were mixed and pointed out a number of problems. We had work to do on communicating with the player, what people liked/disliked about the play experience, and that we needed tutorials for pop-it. However, it was also evident how much the testers liked the game, especially given that they compared it to great platformers that had actually inspired us.


The Pop-It in-game editor.

It was very motivating, but there just wasn't much time. A tough decision was made to cut a full theme because we knew we didn't have enough time to get it to the quality level we wanted. We also wanted to make sure that the developers on the team were able to finish the work they wanted to do. Overall, the design team did fantastic job making changes between the first review and the final playtest.

One of the challenges of building a game based on user-created content is figuring out how to incorporate teaching and training. Very late in the process, a few weeks before alpha in fact, we decided that we needed to teach players the basics of creation more thoroughly, and so we devised a full set of video tutorials and levels. We hired two companies, Maverick for video production and Side for sound recording, who came in to coordinate 60 videos.

There was a lot of pressure to finish pop-it, including the visuals and obvious bugs, so

that we could write an instruction manual that could then be rewritten by the scriptwriter. The scriptwriter then had to give the manual to the vocal talent, Stephen Fry, to record the voice-over for the videos. And then, we needed to churn out multiple translations of the voice-over. This caused the localization work for all nine languages to balloon overnight.

By this point, we were so focused on simply getting the features written that we lost focus on making sure the performances were completed. We had to call on Sony for backup. Sony's Advanced Technology Group stepped in and helped us get everything running at 30fps.

This was a big job, as we kept iterating the game right up until the very last moment. In a perfect world, we would have had more time. However, the architecture of the PlayStation 3 probably helped, as the SPU programs were perfect targets for optimizations by people with little understanding of the majority of the code base.

4 SETTLING ON SERVERS. We spent a long time talking with Sony about how to proceed with the server technology to support the online components. We had proposed a LITTLEBIGPLANET-specific solution, but it was rejected because, we were told, improving an existing tried-and-tested server would be safer than relying on an entirely new server that would be a perfect match, but would require a lot of testing and not map well to Sony's process.

As a result, a whole section of the game had to be managed and developed collaboratively between Sony and Media Molecule.

The launch has been rocky, but that said, the team working on the online components has worked incredibly hard to support the huge amount of online content that has been generated to date.

5 MANAGING A LARGE Q/A PROCESS AS A SMALL TEAM. Alpha is historically the time when a game development team realizes what it does and doesn't have done. As we approached alpha, the game came into focus and we started to get a clear picture of how much quality assurance we would need. That's also when we realized we would need to come up with a way to teach players how to create content and play.

Sony provided a small test team on-site in Guildford, U.K., and we contracted Testology Inc., which provided us a test team onsite. It was quite big change to be working with these two new and huge groups, and we didn't have any processes in place to properly manage them or their feedback, which led to extra responsibilities for the programmers and production team.

/// PLANETEERS

One of the biggest successes in the history of Media Molecule is that we managed to build a studio and ship a game at the same time, without having done those two things concurrently before. LITTLEBIGPLANET is the manifestation of this experiment, and it deeply reflects the team, both the personalities and the talents within it, and the big gamble we all took. It has proven to all of us, and I hope a lot more people outside of Media Molecule, that it's possible to do your own thing if you rely on and trust in the creative talent you have.

Unreal Technology News

by Mark Rein, Epic Games, Inc.

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 for the past three years, and "Gears of War," the 2006 Game of the Year, sold over 5 million units for Xbox 360 and PC.

Epic recently shipped "Unreal Tournament 3" for PC, PlayStation 3 and Xbox 360.

"Gears of War 2" for Xbox 360 was released in November 2008.

Upcoming Epic Attended Events:

CES
Las Vegas, NV
January 8-11, 2009

D.I.C.E. Summit
Las Vegas, NV
February 18-20, 2009

Game Developers Conference
San Francisco, CA
March 23-27, 2009

Please email:
mrein@epicgames.com
for appointments.

OBSIDIAN LEVERAGES UNREAL ENGINE 3 FOR *ALPHA PROTOCOL*

The following excerpt for www.unrealtechnology.com was written by freelance reporter John Gaudiosi.

Obsidian Entertainment's *Alpha Protocol* is a massive, espionage-focused RPG set to be published by SEGA for PC, Xbox 360 and PlayStation 3 in early 2009. Chris Parker, executive producer and co-owner of Obsidian, recently elaborated on why his team is using Unreal Engine 3 to develop the game.

"There are a number of benefits to using UE3," said Parker. "Much of the heavy lifting has already been taken care of with regards to running on all platforms. The material shader system is powerful and enables you to make great looking stuff. We use all of the engine's editors, from interface to cut scene timeline tools, as well as its scripting system for our gameplay needs."

Obsidian's Chris Parker

One of Parker's goals was to create an action-packed RPG, so Unreal Engine 3 was a great fit for the team from the start. Parker said the aim was to make an RPG that was lighter and used shooting mechanics.

"*Alpha Protocol* is a dream project we've wanted to do for a while. We wanted to take everything Obsidian (and Black Isle) has accumulated with story and role-playing systems and apply it to a modern-day RPG," said Parker.

Another key ingredient that Unreal Engine 3 offers is cross-platform development ease. UE3 gave the team a layer between the different console hardware and the assets they wanted to develop.

"It isn't as simple as making models and out pops a game, but we've been able to get versions up and running on PS3, Xbox 360, and PC with very little trouble," said Parker. "We have to pay close attention to how the engine works to ensure that we properly utilize it for each platform, but as long as we're careful UE3 allows us to develop for all three platforms in a consistent manner."

Since *Alpha Protocol* has been crafted for both PC and console gamers, the gameplay has a blend of classic RPG elements with new modern-day spies, gadgets, weapons and action.

"While seemingly straightforward, it doesn't take long before the game starts taking sharp detours," said Chris Avellone, creative director of the game and co-owner of Obsidian Entertainment. "In *Alpha Protocol*, the storyline isn't linear -- we've added a lot more freedom in how you choose to save (or not save) that world, and tried to provide the player with options on how he chooses to uncover the plotline and the relationships between the key characters in the game."

Alpha Protocol incorporates elements fans of fictional espionage characters like Jack Bauer, Jason Bourne and James Bond are used to today. Avellone promised that there will be plenty of romance, colorful villains, action, bullets, betrayals, and alliances.

Unlike television and film spies, it's the player that's in the center of all this intrigue, and the story progresses based on the decisions the player makes throughout the game.

"Many of our previous titles have relied on a hub structure, but *Alpha Protocol* takes the system a step further by presenting the player with a challenge and then offering many choices and missions for how to approach the problem," said Avellone.

"Options include using espionage, surveillance, running-and-gunning, talking to contacts, or computer infiltration. We also allow the player to pick and choose which missions to tackle in order to experience the story in a way that complements character-building choices."

Obsidian's new take on espionage and the RPG genre is sure to offer gamers something never experienced before on PC or consoles.

For UE3 licensing inquiries email:
licensing@epicgames.com

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

WWW.EPICGAMES.COM

Epic, Epic Games, the Epic Games logo, Unreal, Unreal Engine, Unreal Technology, the Powered by Unreal Technology logo, the Circle-U logo and Gears of War are trademarks or registered trademarks of Epic Games, Inc. in the United States of America and elsewhere. Other brands or product names are the trademarks of their respective owners.

nicholas olsen

iDEVELOP

Unboxing the iPhone SDK

WHEN STEVE JOBS UNVEILED THE IPHONE AT MACWORLD 2007 THE EXCITEMENT WAS CONTAGIOUS. Here was a handheld device that combined a desktop operating system, the iPod's media capabilities, a multi-touch high-resolution display, full-featured web browser, Wi-Fi, camera and accelerometers. When the hype-machine reached its apex for the June 2007 launch, lines snaked from one end of the mall to the other as consumers scrambled to be first to get one. Absent from the initial hype was support for developers. In fact, in the early days of the iPhone announcement the official word from Apple was that there would be no official support for 3rd party applications. This stance was later amended by emphasizing Safari's support for rich internet applications, with Apple suggesting web developers would be able to create applications that felt more or less like native applications. Fast-forward to March 2008 and Apple unveils the iPhone SDK, offering developers access to the same tools Apple has been using to build applications internally, as well as reams of documentation and reels of introductory videos. Although the SDK has effectively been publicly available for months (the only impediment being the need to register for a free iPhone Dev Center account), an NDA prevented discussion of the materials or publishing of articles related to the iPhone tool chain. That restriction was relaxed in October and we can finally bring you an overview of what awaits the would-be iPhone developer.

A BRIEF HISTORY OF THE IPHONE SDK

While the iPhone is Apple's latest product, some of you may be surprised to learn that its development tools go back to the 1980s. After Steve Jobs was forced out of Apple in 1985 he founded NeXT Computer, Inc. with the goal of creating custom 68k-powered workstations that combined the Mach microkernel with a Unix environment topped by an object-oriented application development framework and a display system based on Adobe Postscript. Over the next decade the NEXTSTEP operating system would evolve into OPENSTEP while being ported to x86, SPARC and HP-UX hardware. During the same period the NeXT APIs became the OpenStep APIs, developed as an open standard in cooperation with Sun Microsystems. The development tools for OpenStep would eventually be hosted on Solaris and Windows NT in addition to NeXT's OPENSTEP operating system.

By 1997, after years of failing to develop an operating system to supersede the aging Mac OS with its cooperative multitasking and lack of protected memory and multiuser capabilities, Apple turned to acquisitions. NeXT ultimately became the takeover target, which initiated a reverse-takeover of Apple by former NeXT staff, including the triumphant return of Jobs as Apple CEO. When Mac OSX began shipping as an Apple product, some of the major features touted to developers were the Cocoa API and development tools, primarily Project Manager (the IDE) and Interface Builder (a graphical user interface design tool). These offered the promise of a clean, modern, object-oriented application development framework that would allow developers to build applications quickly and easily. These gradually evolved into the current suite of Xcode, Interface Builder, and Instruments.

So, while the preceding backstory may seem like a significant diversion from the practicalities of what prospective iPhone developers face, it might be helpful to ground the discussion that follows in the history of the NeXT operating system and tools. While the device itself has been on the market for a year and a half and the iPhone SDK is less than a year old, the operating system, development tools, and APIs have a heritage that includes 20 years of shipped software for a number of platforms.

Nicholas Olsen *is an audio programmer at Double Helix Games in Irvine, California. He is a Southern California native and has been in the game industry for seven years. Contact him at* **nolsen@gdmag.com**.

GETTING STARTED

One of the most important considerations for developers evaluating a new platform is the time and money needed to get a development system up and running. In the case of the iPhone SDK, the cost of entry is free ... Sort of. While the SDK is available for download with a free Apple Developer Connection account, the tools are hosted on Mac OSX. In an industry where Windows dominates the development workstation landscape, this probably means you need a Mac. As of this writing the least expensive Mac is the $599 mini, which is more than adequate for iPhone development. While this compares very favorably with development kits for consoles and other proprietary platforms that can easily run into the thousands of dollars, it is not as accessible as other mobile platforms that offer free SDKs that are hosted on Windows as plugins for Eclipse or Visual Studio. For on-device debugging and testing you also need a subscription to the iPhone Developer Program, which starts at $99 and provides you with a certificate necessary to sign code for execution on iPhone and iPod Touch hardware. This subscription also gives you the ability to distribute apps through the iPhone App Store.

Installation of the tools consists of downloading the SDK installer image from the iPhone Dev Center, mounting it, double-clicking the installer package and waiting 10 or 15 minutes. At this point you can run Xcode, create a new project, choose a project template, click Build and Go, and have an application running on the simulator. While it may seem like a small thing, the inability of many vendors to provide working installation procedures has been a pet-peeve on many platforms, so this straightforward installation procedure is very much appreciated.

WHAT NEXT?

With tools installed, the next challenge is figuring out what we can do with this platform and how to do it. Apple provides extensive introductory materials in the form of "getting started" guides and hours of video. The videos provide a helpful grounding in the capabilities of the device and tools, covering topics ranging from getting started with Xcode to user interface design, graphics and media capabilities, using the multi-touch and accelerometer capabilities, and designing web applications that target iPhone. As the iPhone has a small screen that presents one application window at a time, several of the sessions discuss how to accommodate this model using the built-in UI widgets. These are mostly introductions to what is available on the platform, so they don't go too deep into the specifics of creating a complete application, but they do provide examples of how to exploit various features of the phone.

The overview documentation provides introductions to the tools, the iPhone OS and software stack, and the Objective-C language. Developers with no prior Cocoa development experience will probably want to watch the Fundamentals of Cocoa Session video, which provides an overview of Objective-C, and a walkthrough of creating a Cocoa application with Xcode and Interface Builder. Beyond this, you'll want to take a look at the videos for Object-Oriented Programming with Objective-C, Cocoa Fundamentals Guide, and iPhone OS Programming Guide to get the lay of the land. As you dig deeper into specific features of the SDK, there are additional guides for every part of the iPhone OS, as well as extensive online API documentation available through Xcode's documentation browser.

TOOLS

The heart of the iPhone SDK, the Xcode IDE, provides project and resource management, integration with the GDB debugger and iPhone simulator, and code editing capabilities. The editor itself provides the usual syntax-highlighting, auto-completion, integrated documentation, and source control integration (CVS, Subversion and Perforce) that one would expect. As editors tend to be a matter of extreme personal preference I won't go into detail about Xcode beyond saying that it does what I expect it to do and does it pretty well. In particular I thought the All-In-One layout did a nice job of making my relatively small laptop display usable.

Xcode's companion tool is Interface Builder, which allows developers to create user interfaces graphically and to establish connections between controls and program objects. The basic workflow in Interface Builder involves dragging and dropping controls onto a representation of your application's screen layout and then wiring up their Outlets and Actions. In Cocoa parlance, Outlets and Actions are class methods and members that are exposed to Interface Builder. A button, for example, can be associated with an Action in one of your classes so that button presses invoke the associated method. Outlets bind to member variables of an object, so for example, a text label in your application code would be exposed as an Outlet and connected to a text field in your Interface Builder project. When the Interface Builder file is loaded at runtime, the Outlets in your application code will be automatically bound to the controls in your user interface, so when your code sets the label Outlet's text property, the user interface updates accordingly.

WHAT KIND OF C?

Perhaps the biggest adjustment for newcomers to OSX development is discovering that development is not via a C or C++ API, but an object-oriented C dialect that is almost exclusive to Apple platforms. Objective-C was the brainchild of Brad Cox and Tom Love, who were inspired by Smalltalk when they set out to create an object-oriented variant of C that was later licensed by NeXT. It is a strict superset of C, adding a small number of keywords along with a messaging syntax borrowed from Smalltalk and the id type to represent objects. The language also differs from C++ in that dynamic typing and binding are the rule, and Cocoa makes use of this behavior extensively, particularly through the use of delegates and object composition.

```
// Method invocation in C++
object.method;
// Method invocation in Objective-C
[object method];
```

Memory management is provided by the familiar `malloc` and `free`, or in the case of objects the `alloc` and `dealloc` methods. Reference counting is baked into the Objective-C runtime, as all objects derive from a root NSObject that provides `release` and `retain` methods for managing an object's reference count and allowing the runtime to reclaim un-referenced memory.

Apple's low-cost Mac Mini.

```
// Object life-cycle in an Objective-C application
// This allocates and initializes an object,
// which starts life with a reference count of 1
[[object alloc] init]
// Decrements object´s reference count, bringing it
// to zero and causing the runtime to automatically
// call object´s dealloc method to reclaim its memory
[object release]
```

CONTINUED ON PG 24

CONTINUED FROM PG 23

Coming to grips with the conventions for the proper use of `release` and `retain` was perhaps the most confusing aspect of Cocoa programming, so I suggest reading Apple's *Memory Management Programming Guide for Cocoa* http://tinyurl.com/6853k5) for the details.

APPLICATION DEVELOPMENT TECHNOLOGIES

IPhone OS shares its underpinnings with Mac OSX, including the Core Foundation, Core Graphics, Core Audio, and Foundation Frameworks. Core Foundation is a C API that provides low-level data types and system services including collections, I/O and networking while Foundation provides an Objective-C API for the same functionality. Because functionality is shared between Core Foundation and Foundation (they are essentially C vs. Objective-C interfaces to the same low-level functionality), many data types can be toll-free bridged between the two APIs, meaning they can be used interchangeably with or typecast to their companion API's equivalent data type with no performance overhead.

The major difference in the application stack between iPhone OS and Mac OSX is the introduction of the Cocoa Touch user interface library in place of the Cocoa library used by Mac OSX applications. Cocoa Touch classes are provided by the UIKit Framework (in place of OSX's AppKit), and provide a reduced set of UI controls targeted towards the iPhone's display and user interface. UIKit's drawing is integrated with Core Animation, meaning many UI elements can be animated implicitly with very little work on the programmer's part. Cocoa Touch also provides support for handling multi-touch events, and while supporting gestures is left as an exercise for the developer, documentation and code samples provide examples of handling common cases.

GRAPHICS

Graphics are provided by the Core Graphics and Core Animation Frameworks, or for more advanced applications OpenGL ES. While UIKit itself provides extensive support for 2D drawing and implicit animation, if you need to do more complex drawing, the Core Graphics API provides finer control of drawing vectors, bitmaps, gradients, and PDFs. It is also possible to use the Core Animation APIs explicitly for fine-grained control of animation using a layer-based animation system that integrates content from OpenGL, QuickTime and the media rendering libraries.

The iPhone's GPU is a PowerVR MBX, which game developers may recognize as a descendant of the GPU used in the Sega Dreamcast and NAOMI arcade boards. Specifications can be found at http://www.imgtec.com/powervr/mbx.asp, and a detailed programming guide is available at http://tinyurl.com/5nngcx.

OpenGL applications are limited to 24MB of memory for textures and surfaces, so developers may want to pay attention to the support for the PVRTC texture format exposed by the GL_IMG_texture_compression_pvrtc extension and the texturetool provided with the SDK for compressing textures.

SOUND

Core Audio provides playback, recording, mixing, steaming, and vibration. Apple recommends using System Sound Services for one-shot uncompressed sound effects shorter than 30 seconds while Audio Queue Services is available for compressed audio, long sounds, and situations where continued control over playback is required. An important limitation of the iPhone hardware is that only one compressed audio stream can be played at a time, so for simultaneous playback effects need to be PCM.

For games and applications that require positional audio, the iPhone also supports OpenAL 1.1, support for which is built on top of the Core Audio framework. Apple recommends OpenAL for games and applications that require low-latency, while also suggesting it is appropriate for general audio needs. For a primer on iPhone compression, see *Squashed*, December 2008, pg 43.

INSTRUMENTS

Instruments is the turnkey performance analysis tool for OSX and iPhone OS, giving developers the ability to monitor memory allocations, memory leaks, CPU utilization, and graphics performance. The stars of the show here are CPU Sampler and Leaks. CPU Sampler is a hierarchical profiler, meaning it runs your program and tells you how much of the CPU is being used by which function. This allows you to quickly pinpoint performance problems in your code, as the offending function will be clearly shown to consume too much CPU time.

Leaks, as the name implies, is a tool that finds memory leaks in your code. I found this to be quite helpful, especially in learning the correct use of `release` and `retain` to keep the Obective-C runtime happy. I was able to run code under Leaks and immediately see where I had introduced a memory leak, then go directly to the offending function and repair it. Apple takes great pains to emphasize the importance of respecting the system's limited RAM and lack of a pagefile; if your application is using too much memory it receives a notification from the system and must free memory or be force-quit. Leaks does a good job of reigning in the developer's carelessness.

THE SIMULATOR

The iPhone SDK provides a simulator of the iPhone hardware and software so that developers can get started with or iterate on development without the need for iPhone hardware. An important point to note is that it is a simulator rather than an emulator, so it does not provide a cycle-for-cycle equivalent of device performance and there are a several notable omissions. As of this writing, the iPhone OS 2.1 simulator supports OpenGL ES, OpenAL, and the 2D user interface functionality provided by UIKit. Notable omissions are accelerometer and GPS data, a limitation that could be rectified by providing developers with an interface to feed the simulator sample data. It is also impossible to prototype complex multitouch gestures on the simulator, although it understands the common cases of swipe, flick, drag and pinch.

DEPLOYMENT

Distributing applications on the iPhone requires an iPhone Developer Program account. With a $99 standard account you receive a certificate that enables on-device debugging and Instrument and the ability to distribute your applications through the iPhone App Store. When distributing through the App Store, the developer sets the price and receives 70% of the revenue (there are no distribution fees for free applications). There are no additional hosting, processing or marketing fees and revenue checks are sent monthly. Apple also offers a $299 enterprise program that allows companies to develop and deploy in-house applications.

Light It Up! *Quake Wars** Gets Ray Traced

BY DANIEL POHL

A SCENE UNFOLDS in the computer room of a major university. Those watching sense electricity in the air, the kind of tension that builds before a thunderstorm, as a cluster of 20 networked PCs, each equipped with spanking new dual-socket technology and dual processors, warm to the task assigned to them: distributed ray tracing of the game *Quake* 3* (www.q3rt.de). Though the modest display resolution (512x512) and a frame rate of 20 frames per second (fps) aren't overwhelming by the standards of the day, this doesn't diminish the accomplishment in the least. Special effects never before seen flimmer across the display screen. The viewers watch with rapt attention and a feeling of satisfaction as the intricately rendered images move about the screen. Amazingly, this happened in 2004, a time when most people rejected the concept of real-time ray tracing.

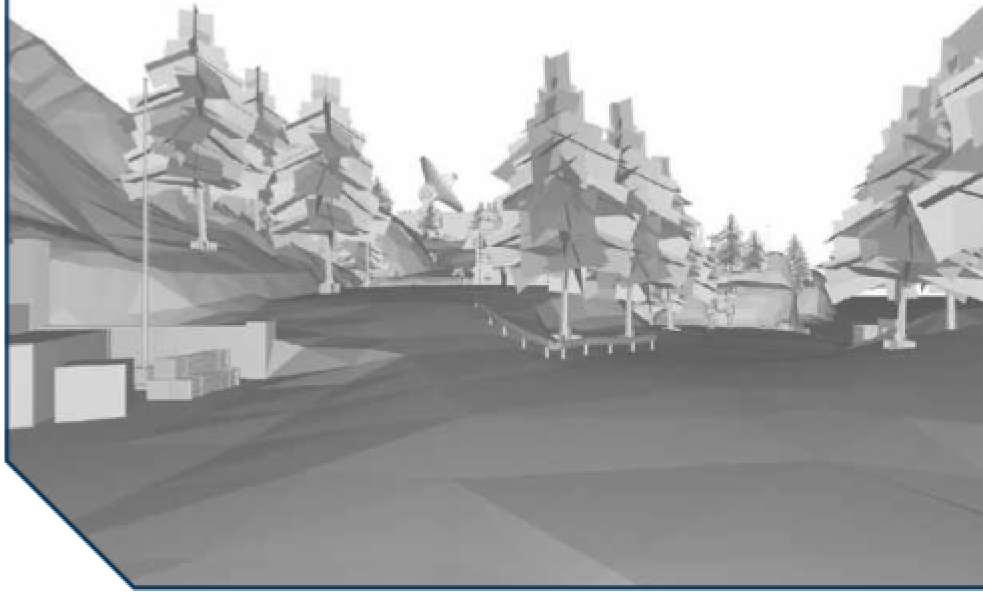
BACK TO THE FUTURE (2008, THAT IS)

A new research project from the ray-tracing team at Intel advances beyond the 2004 achievements, this time converting the game *Enemy Territory: Quake Wars**, which was created by id Software and Splash Damage, to use ray tracing. Read on to learn about the development process that followed, the challenges we had to overcome, and the benefits we ultimately achieved—all of which provide valuable insights into the future of ray tracing. To pump up your visual adrenaline level, we've also included numerous images to show the process in action. By the time you finish this article, you'll have a better idea of the ways in which ray tracing can quickly and easily render light and shadow.

STARTING FROM SCRATCH

For this project, we started rewriting the renderer from ground zero. Because of this, the very first images from the renderer were not of typical ray-tracing caliber, but displayed only the basic parts of the geometry, without any shaders or textures (Figure 1). Typically, games load their geometry from a variety of different model formats—either created over the in-game map editor or through external

Figure 1. Quake Wars*: Ray traced without textures.



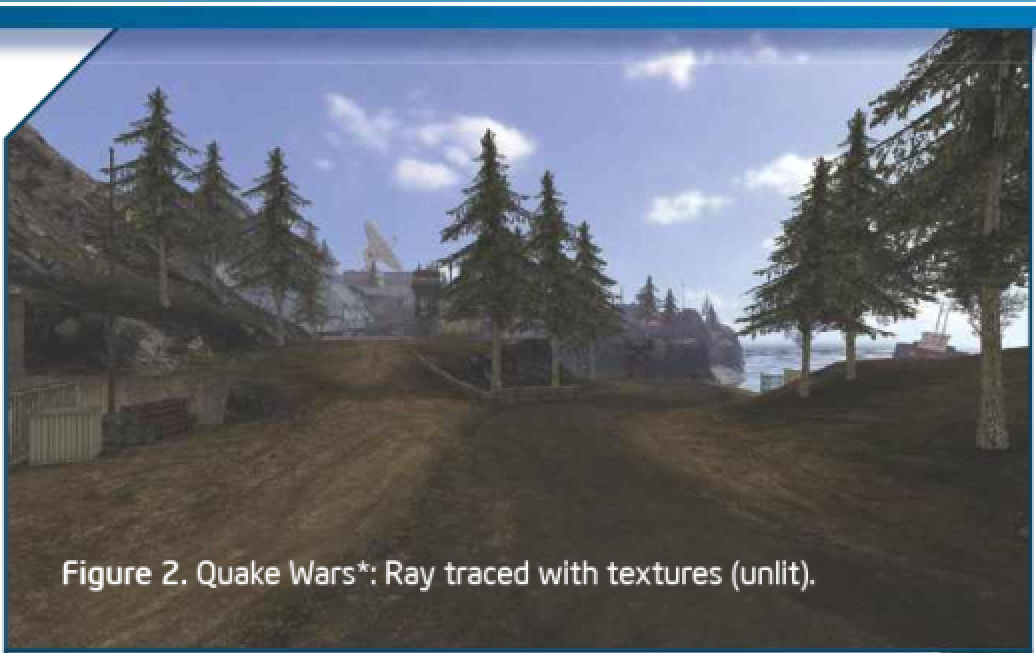


Figure 2. Quake Wars*: Ray traced with textures (unlit).

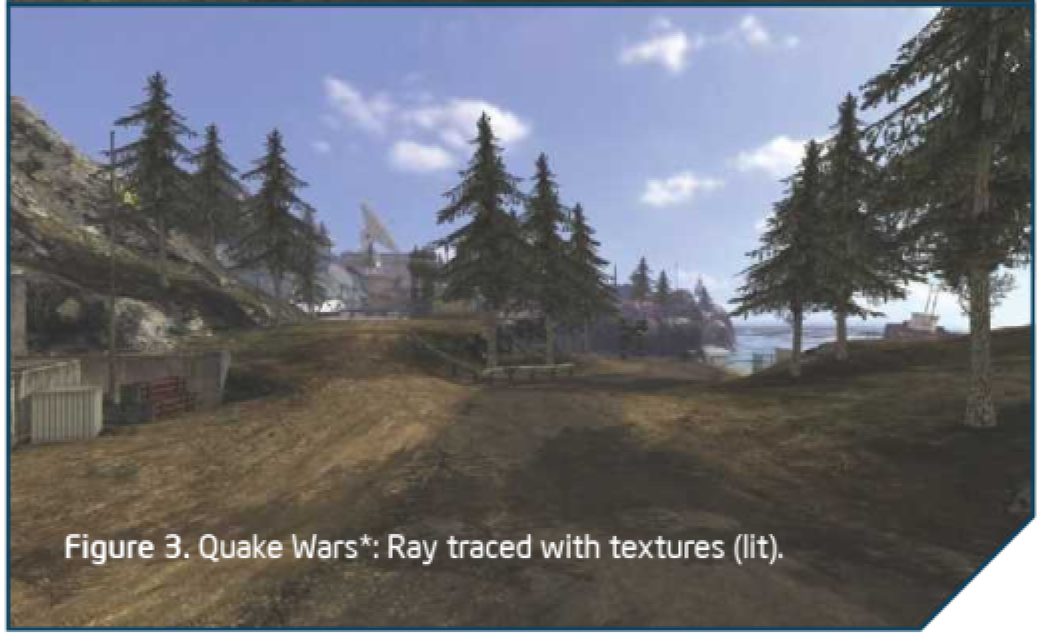


Figure 3. Quake Wars*: Ray traced with textures (lit).

modeling tools. Once it is verified that there are no missing objects, the loading of textures can begin. Modern games have their own material description language that allows designers to easily modify texture parameters, blend textures, use bump and specular maps, and write small shader programs. For example, compare the untextured image in Figure 1 with the unlit (Figure 2) and lit (Figure 3) textured images of the same scene.

Today's games all use a rendering technique called rasterization. Rasterization requires difficult programming work as many special effects (such as shadows or reflections) need to be calculated as approximations over multiple rendering passes and are often stored in resolution-limited textures in between. These approximations fail in certain cases. Let's look closer at shadows. With ray tracing, you only need to check if the path from the light to the surface is blocked or not. This can be easily determined with just a ray (the so-called "shadow ray"). If the ray from the light source can reach the surface, the point on the surface is lit. Otherwise, it is in shadow. The gameplay in *Quake Wars: Ray Traced* takes place primarily outdoors, where the most important light source is sunlight. We were able to apply this form of lighting to the scenes quite easily, and the appearance of the shadows is what one would expect.

TRANSPARENCIES

Instead of employing real 3D geometry, game developers sometimes approximate 3D properties with a 2D quad surface (or two triangles, as shown in Figure 4) and a texture on which transparency values have been applied.

Creating correct shadows from partially transparent quads is not an easy task for a rasterizer. The most commonly used algorithms for calculating shadows in rasterization (called "shadow mapping"—see http://en.wikipedia.org/wiki/Shadow_mapping) does not deliver additional information that might help in the case of shadows from transparencies. For that reason, shadows are sometimes baked into textures, and, because of this, they don't change when the light position changes

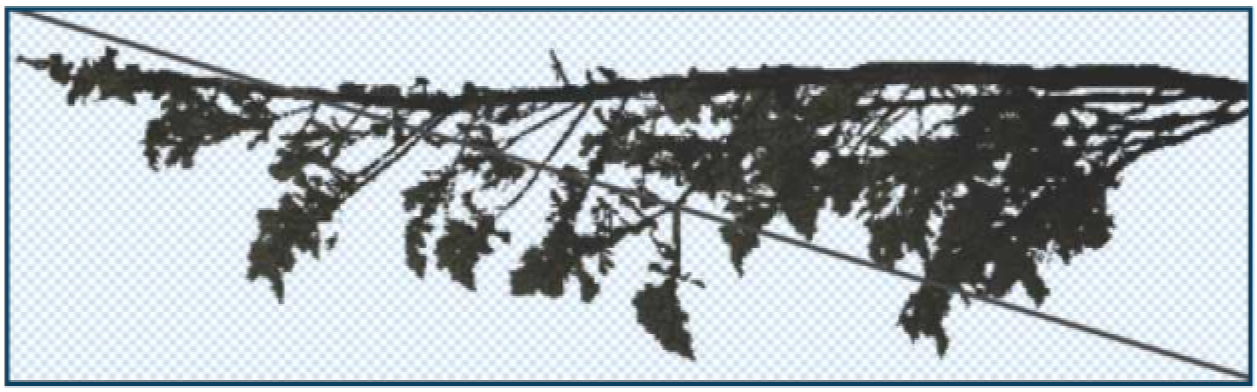


Figure 4. Example of a partially transparent leaf texture applied to a two-triangle surface.

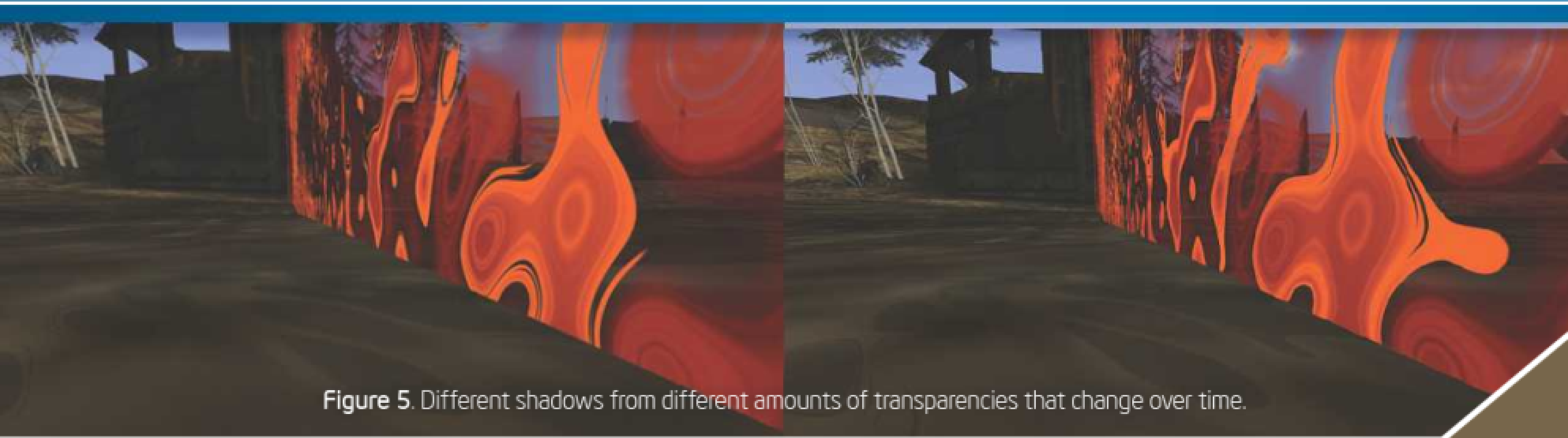


Figure 5. Different shadows from different amounts of transparencies that change over time.

(such as when a scene changes from sunrise to sundown).

When using ray tracing, however, the algorithmic solution is simple. If the shadow ray hits an object, the program can read the transparency value of the texture and continue tracing that ray, when the texture sample is transparent. This offers interesting special effects, but also creates challenges. The images in Figure 5 show an animated force-field shader effect that casts a different intense shadow depending on the transparency values of the orange force field.

Another advantage of using a ray tracer for partially transparent objects is that they don't need to be sorted by their depth. This makes it easier for the developer, but there is a downside: increased rendering costs. Whenever a ray hits such a surface, another ray needs to be shot from that point in the same direction. If this happens once, the impact is small. But what happens if you have ten or more of these surfaces in a row? This can happen if a tree, for example, consisting of a mix of partially transparent quads, is rendered [Figure 6(a) and (b)].

Rendering a large number of those trees in the outdoor world quickly became our biggest performance bottleneck. During several optimization cycles, we came up with many improvements. The following improvements had the greatest impact on performance.

- Avoid shooting a new ray each time after reading the transparency value; instead, we reused the same ray now originating from the hit position and continued in the same direction afterwards.
- Signify whether a texture uses transparencies with a single flag. If not, there is no need to shoot additional rays through potential transparencies. The bark inside a tree is an example of an opaque texture mixed in between many partially transparent textures.

Decrease the number of rays that are bundled together. In many cases, bundling rays with almost the same path can lead to substantial speedups. However, if one part of the bundle hits another surface then the other one, it produces some reorganization overhead to split those bundles. In rendering the trees, this

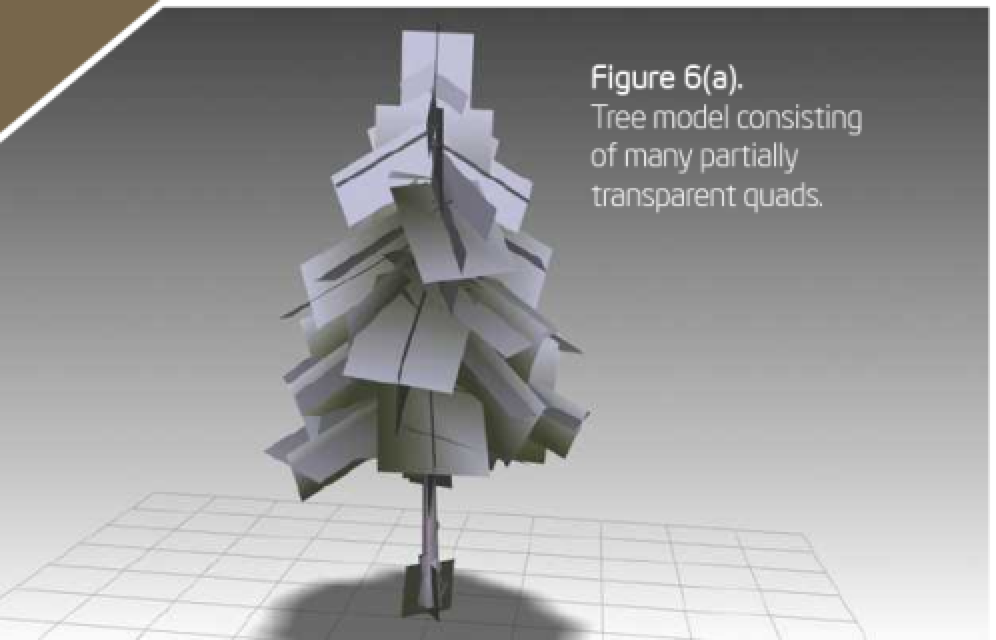


Figure 6(a). Tree model consisting of many partially transparent quads.

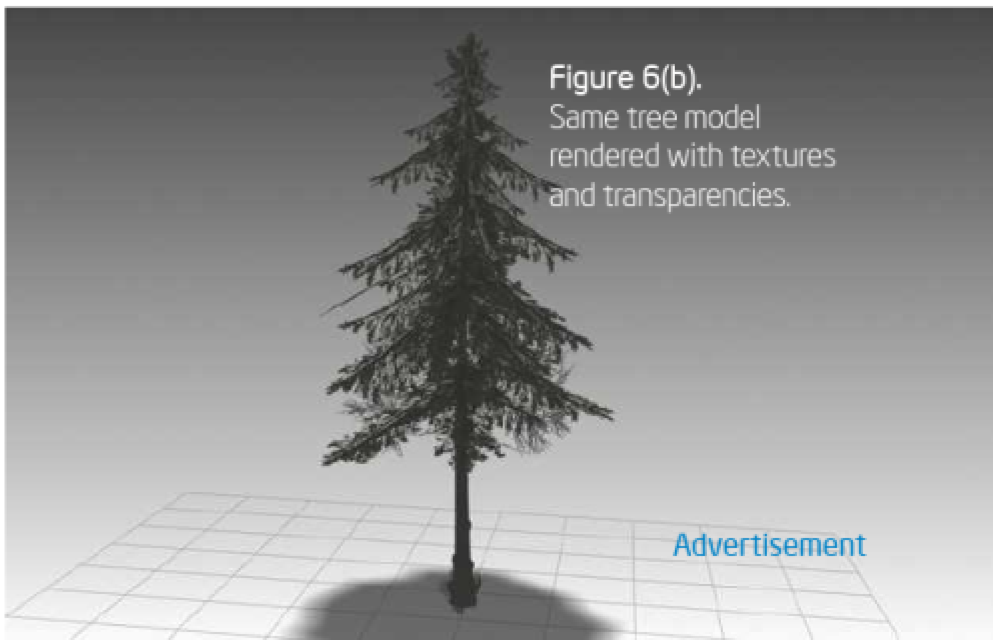


Figure 6(b). Same tree model rendered with textures and transparencies.

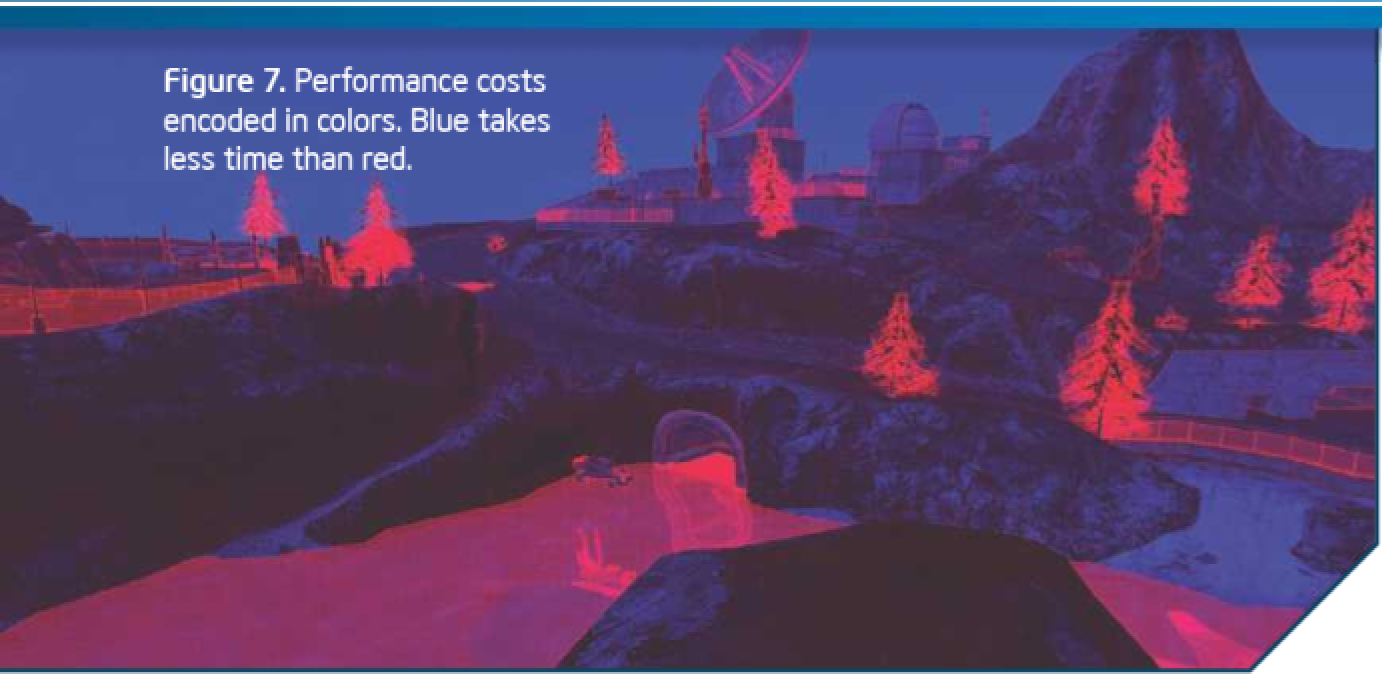


Figure 7. Performance costs encoded in colors. Blue takes less time than red.

overhead can become significant, slowing everything down.

Even after a great deal of tweaking, rendering the trees is still very time consuming. We visualized the costs of rendering a single pixel in a color scheme where a blue pixel represents a quickly calculated pixel and a red pixel an expensive one (Figure 7). An intense red is also more costly than a light red tone. As can be seen in the figure, rendering the trees is still more expensive than rendering a reflecting water surface or other parts of the scene. More research needs to be done on rendering these trees to discover if further improvements are possible.

ADDING MORE SPECIAL EFFECTS

In our ray-tracing conversion, once we reached the same quality as the original game, we began adding enhancements and more special effects. Ray tracing does a very good job with reflections and refractions. The most common everyday objects in the world that exhibit this behavior are glass and water.

GLASS

A large dome exists in the original game. We changed the surface properties of the dome so that it would appear to be made out of glass (Figure 8). Using the refraction index for glass (which you can find in your favorite physics books), we wrote a shader to accurately depict the reflections and refractions. The code is about 15 lines long in our HLSL-like ray-tracing shading language and generates very pleasing results.

WATER

Rendering water can be accomplished different ways. We investigated two approaches: water on a 2D surface and water with genuine 3D properties [1]. To render the water in 2D, we used a bump map to simulate waves [Figure 9(a)]. The 3D water image uses a mesh with around 100,000 triangles in several subgrids [Figure 9(b)]. Those subgrids are updated every frame, depending on their visibility. (During rendering, subgrids that are not visible are ignored.) The visibility test is performed over rays.

THE PERFORMANCE ISSUE

Performance is the main reason why ray tracing is not yet used in mainstream games. Compared to special-purpose rasterization graphics hardware—such as current-generation GPUs—ray tracing is fairly slow. Also, a lack of texture units for our CPU-based approach to ray tracing causes significant slowdowns when trilinear filtering

[1]Source: Implementation by Jacco Bikker

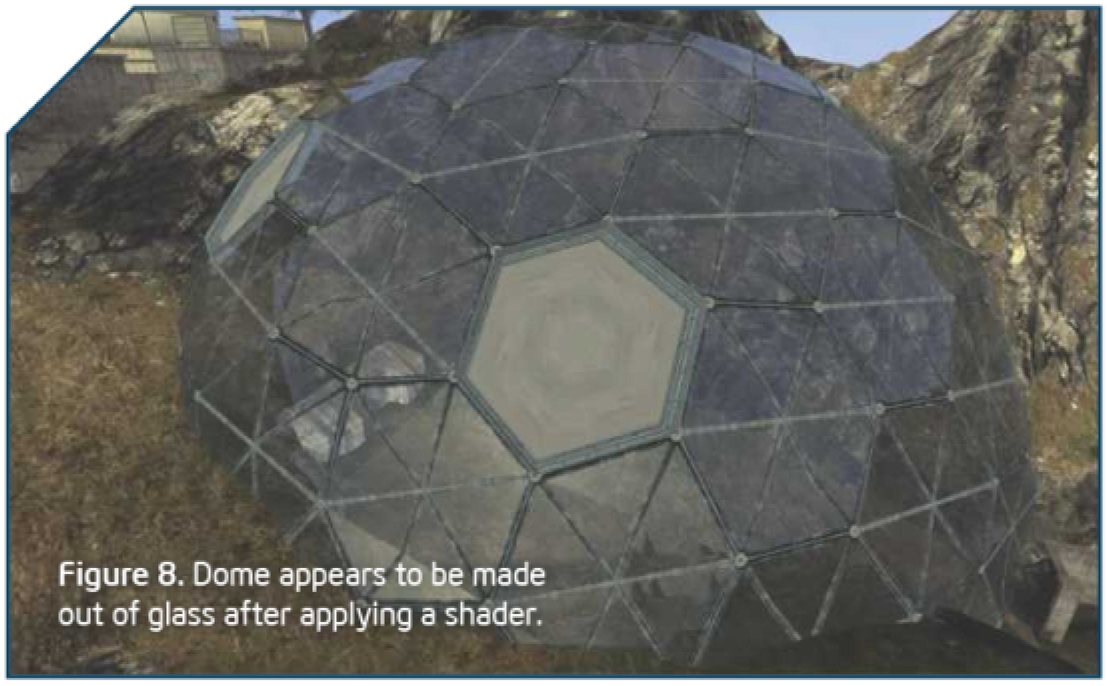


Figure 8. Dome appears to be made out of glass after applying a shader.

is used for all texture samples. With Intel's latest quad-socket systems—equipped with a 2.66 GHz Dunnington processor in each socket—we can achieve approximately 20 to 35 fps at a resolution of 1280x720. Nonetheless, this represents a significant improvement over the experiments in 2004 that required 20 machines to render a simpler game more slowly and at a lower resolution. The greatest performance gains result from research efforts around the world that improve efficiency and the new, many-core hardware platforms that use parallelism to accelerate graphics operations.

THE FUTURE OF RAY TRACING

As mentioned earlier, creating very realistic shadows in games is not an easy task. Given the current state of our demo work, only hard-edged shadows are produced. Modern games tend toward soft shadows, which usually require many more rays. This important topic deserves more study; smarter approaches to this task need to be developed. Also, to obtain higher quality images, better anti-aliasing methods are needed. Adaptive super-sampling is a smart way of refining the rendering of the scene at those exact places where it will deliver the greatest benefit. There are experimental implementations, but they need to be tested and tuned for the best results. With the industry moving from multi-core to many-core (that is, greater than ten cores), improving the algorithms so they can fully use the newly acquired power will be interesting.

Even though Intel's upcoming many-core graphic architecture, code named Larrabee, has been primarily developed as a rasterizer card, it will also be freely programmable. This opens up some extremely interesting opportunities to perform ray tracing with the Larrabee architecture.

Stay tuned for more information about our upcoming ray-tracing projects! ▪

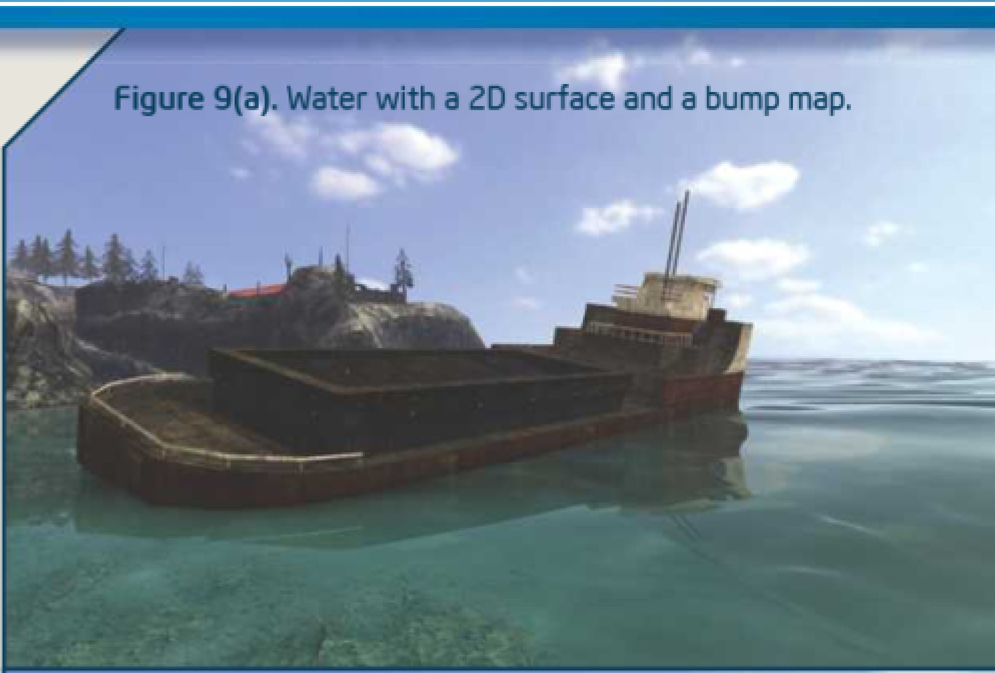


Figure 9(a). Water with a 2D surface and a bump map.

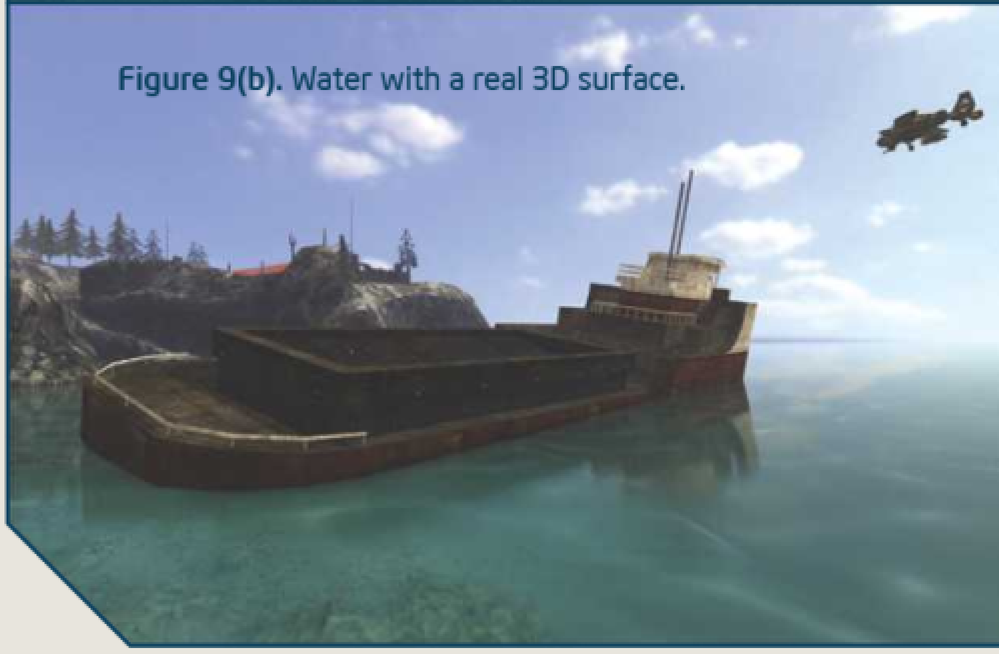


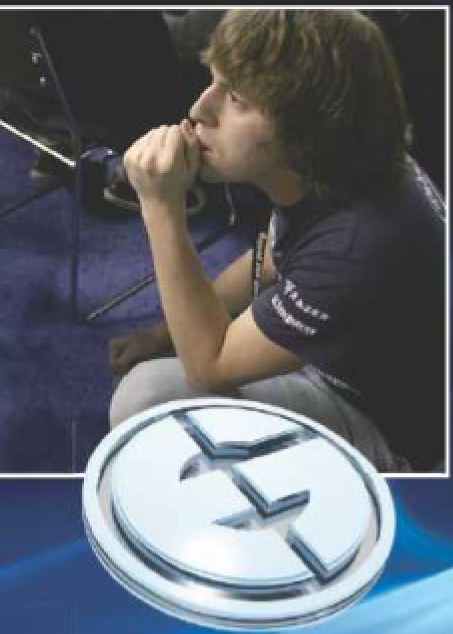
Figure 9(b). Water with a real 3D surface.

ABOUT THE AUTHOR
Daniel Pohl started researching real-time ray tracing for games in 2004 during his study of computer science at the Erlangen-Nuernberg University in Germany. As his master's thesis, he developed a ray-traced version of *Quake 4*. In 2007, he joined Intel's ray-tracing group. In 2008, he moved from Germany to sunny California where he continues to research game-related ray tracing.

To get more great articles like this one, subscribe today to Intel Software Dispatch for Visual Computing at: www.intelsoftwaregraphics.com.

VISUAL ADRENALINE intel Software

Intel does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of third-party vendors and their devices. All products, dates, and plans are based on current expectations and subject to change without notice.
Intel and Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
*Other names and brands may be claimed as the property of others. | Copyright © 2008. Intel Corporation. All rights reserved.



"If I were going to give advice to a gamer today I would just say, 'Don't compromise. Go for a processor like the Intel® Core™ i7; which is easily the best processor I've ever used."

—Alex Garfield, Executive Director, Team Evil Geniuses

Playing games for a living . . .
It's good to be evil.

Alex Garfield, Evil Genius, put the Intel® Core™ i7 Processor Extreme Edition to the ultimate test. The Challenge: Could it perform right out of the box *without turning anything off . . .* and not drop a frame while playing Crytek's *Crysis**?

The Result: Oh yes, it could.

Read the full story today:
www.intel.com/go/visualadrenaline

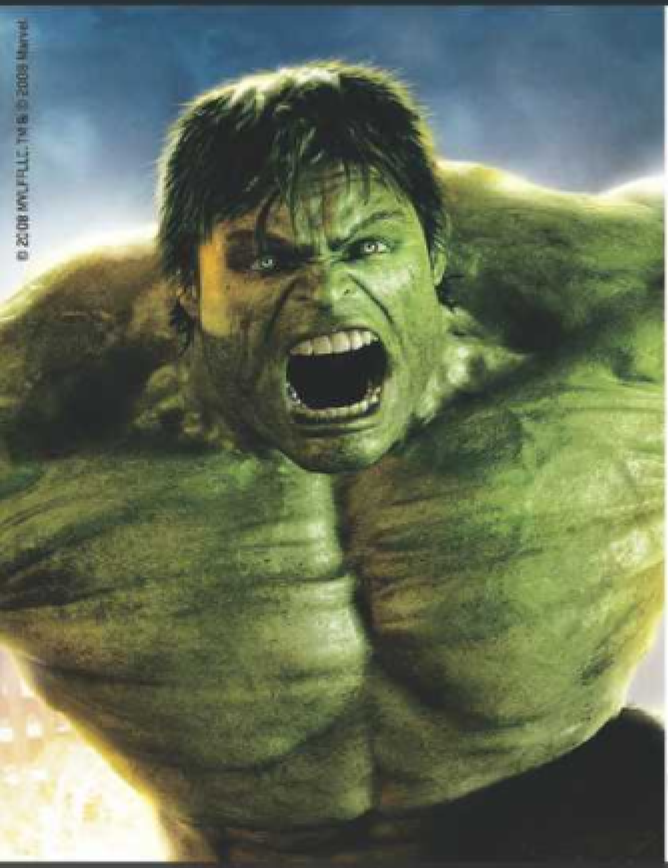
And find out more about other game innovators, technologies, and trends, by subscribing to Intel® Software Dispatch for Visual Adrenaline. It's FREE! **www.intelsoftwaregraphics.com**

VISUAL ADRENALINE intel Software

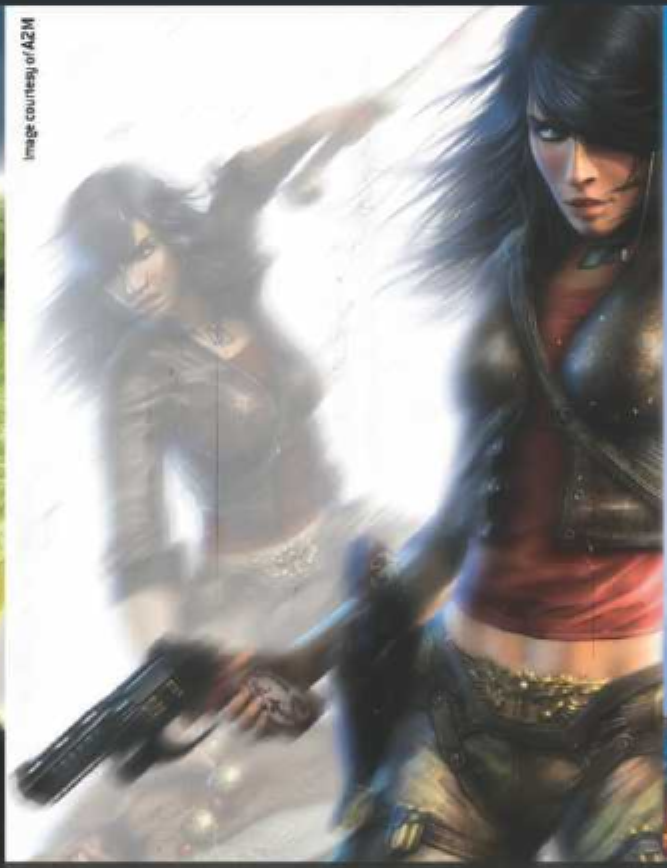
©2008, Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.
*Other names and brands are the property of their respective owners.

3vis

www.3vis.com



Film / TV



Games

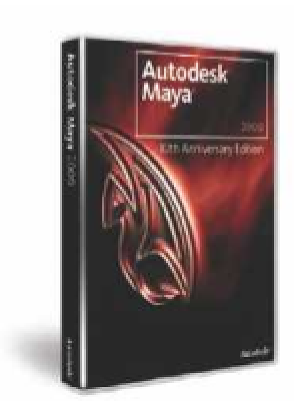
Architecture

Autodesk®
Premier Solutions Provider
Media & Entertainment

CREATIVE INDUSTRIES, CREATIVE TOOLS.

AUTODESK MAYA 2009

By Bijan Forutanpour



AUTODESK

★★★★½

STATS

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

PRICE

$1,995 Maya Complete

PROS

1 Powerful new animation layering feature.
2 Good polish on polygon component selection tools.
3 Improved Maya Asset system.

CONS

1 Face loop selection does not work on triangles.
2 Multi Select mode should exist for NURBS and subdivisions as well.
3 Sculpt Geometry tool needs to be developed further.

I MISS THE GOOD OLD DAYS WHEN software had version numbers, and when even the fractions in the version numbers had meaning. Today software seems to be marketed like automobiles, with year, make, and model names. The lines become blurred when RX7 is a sports car and CS4 is software. Without further pondering of the wisdom of marketing professionals, lets begin our test drive of the 2009 model of Maya, which I would like to nostalgically refer to as Maya 10.0, or Maya X for Mac users.

THE BIG PICTURE

True to its history, Maya continues on its development path with new features and functionality. Some are geared towards efficiency, some focus on managing scene complexity, and others offer completely new functionality. The new developments in the 2009 version display a global view at Autodesk, with a crossover of features and ideas from Autodesk's other products into Maya. With regard to their recent acquisition of Softimage, I'm sure the new baby is in good hands.

IMPROVED MODELING WORKFLOW

Maya 2009 has introduced a set of workflow improvements intended for speeding up modeling, by focusing on one of the most common tasks, namely model and component selection. The first of these is the new multi-component mode, which allows for preselection highlighting of faces, vertices, or edges. As the cursor is moved along the model, the underlying edge, vertex, or face is highlighted before the actual selection is made. This saves the step of using a hotkey or menus for switching between component types. I found it to be useful, but noticed that NURBS or Subdivision surface tools didn't have a Multi-Component mode feature as well. For those of us working in the games industry this may not matter, but for CAD users I'm certain NURBS tool enhancements would be a welcome addition as well.

The next improvement in the Selection tool is the introduction of the soft selection feature. This comes in handy mostly in organic modeling, in changing proportions, or creating bulges using the usual transformation tools, with a smooth blending back to the rest of the model. Soft selection is based on a radius, which can describe either a volume or a surface. There is also a Global mode, in which the vertex selection is not limited to vertices on the currently selected object, but can also include vertices on other objects as well. Choosing a radius setting is easy with a press of the B hotkey, which displays a resizable circular radius cursor and provides live feedback of the effects by color coding the model's vertices.

Another setting is a falloff curve that specifies the "weights" assigned to the vertices selected. It can be edited, and includes curve profile presets and the ability to create new profiles. It was immediately reminiscent of Mudbox's brush controls. This made me ponder Maya's other organic sculpting tools, namely the Sculpt Geometry Tool. The Sculpt Geometry Tool has remained mostly unchanged for many versions, and it could really stand to benefit from the user interface and display features added to the soft selection tool, as well as adding a few new features like pinch and bulge.

Another enhanced feature to the Select tool called Drag Select gives users the ability to paint a selection of faces. This is a raycast version of the existing Paint Selection Tool, with an option for culling backfaces. It does in fact work better than the Paint Selection Tool, which requires an appropriate radius to easily overlap with a face's centroid and work correctly. However, the Paint Select tool also works with vertices and edges, which the Drag Select does not. It would be nice to see the Drag Select and Paint Selection tools be merged into a single, more robust tool.

The Select Tool has added some edge and face loop selection workflow enhancements. Double clicking on an edge now selects an edge loop without having to explicitly invoke the Edge Loop Tool. Faces loops and partial face loops are also now easier to select. Double clicking a face, then hitting shift and double clicking another face will select the faces in between them. However, I noticed this seems only to work on quadrilateral meshes, and not triangulated meshes, or on a mix of triangles and quadrilaterals. Still, selection is more efficient, and once a partial face loop is selected, using the arrow keys to select the full face loop allows for pick-walking the face loop along the model.

One "new" feature to the Move Tool is the Preserve UVs option. This option used to be part of the Maya Bonus Tools, but has made its way into the standard release. Preserve UVs allows for vertices, edges, and faces to be translated while maintaining the overall look of the texture mapping on the model. Usually modifying model geometry after UV texture coordinates and textures have been assigned results in the final result looking distorted. The Preserve UV feature attempts to avoid this distortion by automatically recalculating new UV values for the newly transformed vertices. For the most part the feature works relatively well, but it cannot really account for all scenarios. For instance in the case of discontinuous UVs, where one vertex has multiple UV coordinates, there is only so much that can be done, short of automatically changing the texture's pixels to compensate. Another small detail to note is that even though in the desirable case of continuous UVs, the end results aren't pixel perfect either, and there is still a small amount of distortion visible. It can be a huge time saver, but may require some paintwork to clean up some details.

IMPROVED UV CREATION WORKFLOW

One interesting new feature in Maya is in the area of UV creation. Selected UVs can now be interactively unfolded or relaxed for more control of the final result. The amount of unfolding or relaxing is controlled by how much the mouse has moved, like a scrollbar. Unfolding can also be constrained to a single direction by holding down a modifier key. This gives more control to the user so that when the UV layout is good enough it is possible to stop iterating new UV values.

There are other UV workflow enhancements as well, such as prescaling UVs for multiple meshes such that their world space size is taken into consideration. This helps texture assignment so that there is some consistency in texture pixels per world space unit.

Yet another option in creating UV layouts is Placement Settings, which can help when a single texture is being used for multiple objects. And finally, some general UV editing improvements have been made for when individual or sets of UVs need tweaking.

ANIMATION LAYERS

As Mel Brooks once said, "Its good to be the king." Given that MotionBuilder also lives under the Autodesk umbrella, it was inevitable and much needed that technology from MotionBuilder would make its way into Maya. Any attribute can now be layered, and animation layers can be reordered, merged, grouped, and blended. The animation-layering engine allows for import, export, and reuse of animations, allowing different animations from different characters to be combined. Different types of animation can be combined as well, such as cycle animation, keyframed, constraint driven, or simulated animation. There is the ability to turn off or mute animation layers, or isolate a layer and mute the remaining layers. This allows for experimentation with different takes, or different versions within a single scene. Animation layers may also override conflicting animations that are below them in the animation stack.

Maya has had its current Trax Editor for quite some time, which is also used for editing and compositing animation tracks. Again, I would encourage a unified system that provides the best solution. I believe the new animation layering technology meets the needs of most animators, although time will tell.

MAYA ASSETS

As 3D scenes become more complex, efficiency in managing all the scene data can become an issue. To help manage complex data, Maya introduces the notion of Assets. An Asset is a package that contains other nodes and provides a custom interface that exposes only the necessary controls. This helps with organization, sharing, and using multinode objects in an easier way. Typically the interface exposed would be animation controls, but is not limited to animation. Shading and rendering controls, visibility controls, blend shapes, physics simulation controls, or any other type of attribute can be published, thereby being exposed in the Attribute Editor for that Asset.

Although this feature existed before Maya 2009, in this edition of Maya it has been greatly expanded to include better creation and publishing options, hypergraph improvements, and is said to have better performance. For instance, in 2009 there are user-editable XML template files that can be customized. Different attributes can be presented to different users of the same Asset, based on their individual needs. For example animation controls for animators, rendering controls for technical directors, and so forth. These container nodes can be exported into an asset library, which encapsulates information about the original author, original asset and file name, creation dates, and notes. Asset management systems can now provide a finer granularity of project and production management.

RENDERING

For the film and visual effects industry Maya has been updated to provide all the intermediate rendering passes involved in the final image creation. This allows for precise control and fast iteration in the production pipeline by allowing 2D compositing packages to combine the render shading elements. There are over 50 production level passes available, such as refraction, reflection, translucency, motion vectors, coverage, and custom render passes.

Another new feature in Maya 2009 is the ability to create stereoscopic renders. The 3D viewer now supports a three-camera setup for left eye, right eye, and center, and renders the final stereoscopic image. This allows for quick previewing without the need for final renders. When it comes to creating the final renders, the render layers needed are automatically generated to produce the separate images for the left and right eye. Finally, in the interest of rendering speed, stereoscopic multi-camera renders are optimized in Mental Ray so that computational results are cached and reused. These include render time tessellation, final gathering, global illumination, light maps, and shadow maps.

MORE MORE MORE

There are actually many more features and enhancements to Maya 2009 than those mentioned here. For the purpose of brevity, the enhancements and features that relate to game production more directly have been reviewed in greater detail, leaving less time to discuss the less relevant.

WHAT'S LEFT?

Perhaps the biggest feature not yet discussed is the continued development of Maya Nucleus, the new advanced unified simulation framework. Available only in Maya Unlimited, Maya Nucleus is at the heart of Maya nParticles, nCloth, and nRigids. Cloth, particles, and rigid bodies can all interact under Maya Nucleus, from collisions and stickiness, to cloth tearing and liquid simulation. One fact to note is Classic Cloth is not available in 2009.

Another area of development is the improved IPR (Interactive Photorealistic Rendering) for Mental Ray. Objects and lights can now be moved, duplicated, and instanced. More light types are now supported, including shadow maps and area lights.

Last but not least is Maya Muscle, an expanded muscle and skin deformation toolset that allows animators to create muscle and skin motion on their characters. The muscle system now offers NURBS-based muscles in addition to their previous muscle types. Other controls added include jiggle, wrinkles, sliding, stickiness, and collisions.

MILEAGE MAY VARY

If Maya 2009 were a sports car I would say that it has gained some good performance enhancements, and a few new engine upgrades. From the new MotionBuilder animation layering technology, to Maya Nucleus, it also handles complex models nicely with Maya Assets. The 5 speed manual transmission gives you better control over UV creation and editing, and the new Select Tool features with Soft Selection and Multi Component mode provide a smooth, fast ride. Sometimes there are too many controls, and some of them seem to do the same thing, but no car is perfect. With so many cars in the Autodesk garage, we know the mechanics have a lot of work ahead of them. But then again, these cars are fun to drive.

BIJAN FORUTANPOUR *is a senior graphics programmer at Sony Online Entertainment in San Diego with 16 years experience in the visual effects and games industries. He is also the author of Enzo 3D paint for Photoshop (www.enzo3d.com). Email him at* **bforutanpour@gdmag.com**.

SEATTLE, WASHINGTON

THE MICROSOFT CORPORATION STRETCHES far across the game development landscape and the root of its $230 billion mountain lies in Seattle, the childhood home of its founders Bill Gates and Paul Allen. Although formed in New Mexico in 1975, Microsoft relocated to the Seattle area in 1979. Two years later Microsoft introduced MS-DOS for the 8086 family of processors and the software was such a success that it became the default operating system for the vast majority of PC games released over the next ten years. Late in 1995 Microsoft released Windows 95 with a revamped interface and soon after the company made a serious commitment to game support with the introduction of the DirectX APIs.

Although Microsoft had previously funded game development with titles like FLIGHT SIMULATOR and AGE OF EMPIRES, the company knew that it would need to invest heavily in first-party development talent when it decided to enter the hardware business with the Xbox console. Known for making thoughtful and visually ambitious games for the Macintosh, Chicago-based Bungie seemed like an odd fit for Microsoft's new console. Joining the company in 2000, the studio packed up and moved to Microsoft's campus. As work progressed on what would become the genre-defining HALO, the studio chaffed under the Microsoft corporate structure and eventually moved to its own location in nearby Kirkland. Days after the release of HALO 3 Bungie became an independent studio again although Microsoft retains a minority stake and the two companies continue to have a close publishing relationship.

Over the years, many ex-Microsoft employees have started their own companies in the Seattle area. The founders of Valve Corporation, Gabe Newell and Mike Harrington, cut their teeth on OS development at Microsoft before forming a game development studio in 1996. Influenced by ULTIMA UNDERWORLD and the new wave of first-person shooters that were hitting computer screens in the 90s, Valve created a sophisticated mix of action and narrative in HALF-LIFE. Since then the company has become a major presence on the game development landscape, producing critically acclaimed games and developing its Steam digital distribution platform.

LEAVE LUCK TO HEAVEN

Like Microsoft, Nintendo of America (NOA) did not pick Seattle as its initial choice for a headquarters. Founded in 1980, the fledgling arcade machine company first set up shop in New York City but soon found itself falling behind the curve as it waited weeks for cabinets manufactured in Japan to be shipped across the Pacific, only to be delayed even longer as they made their way across the continent to the East coast. Seattle was quickly decided on as a new head office because it had the advantage of being a port city, had a lower cost of living, and was home to a large pool of skilled workers which to hire from.

Unfortunately, NOA was saddled with a huge inventory of unsold RADARSCOPE cabinets. While the game was a hit in Japan, it had flopped badly in America and NOA was stuck with a warehouse full of unsellable machines. Desperate to recoup its mounting losses, NOA begged Nintendo Japan to send them a new game on circuit boards that could be retrofitted into the RADARSCOPE cabinets. It was sent an oddly titled game called DONKEY KONG that had been developed by an apprentice designer named Shigeru Miyamoto. A test machine was set up in a local bar and within days people were lined up to play the new game. DONKEY KONG had saved the company. By 1982 Nintendo had outgrown its Seattle warehouse and a new a headquarters was built in nearby Redmond.

In the decades since, Nintendo of America has largely acted as a publisher and distributor, although the Nintendo Software Technology development group (METROID PRIME: HUNTERS) makes its home at Nintendo's Redmond campus, as does the DigiPen Institute of Technology. Originally founded in Vancouver, Canada, the institute partnered with Nintendo in 1998 to open the DigiPen Institute of Technology in Nintendo's office building. Its graduates have gone on to work on a long list of games throughout the industry, perhaps the most famous example is the NARBACULAR DROP team which after being noticed at a job fair, was tapped by Valve to develop PORTAL.

COFFEE HOUSE

When Pop Cap Games' BEJEWELED was released in 2001 the game's phenomenal success ushered in a new era of casual games, not as a genre but as a bonafide industry. Since then Seattle has become a major hub for casual and mobile game development. RealArcade, Big Fish Games, Amaze Entertainment, Big Top Games, Sandlot Games, I-Play, Mobliss, and WildTangent all make their home in the Seattle area. UIEvolution, previously supporting Square Enix's mobile games division and now independent, is based in Bellevue. The Casual Games Association, which is also headquartered in Seattle, helps keep the industry linked and informed through its *Casual Connect Magazine* and yearly Casual Connect Seattle conference.

Of course, the Seattle area is home to core developers as well. Arena.net, the creators of GUILD WARS is based in Seattle and SUPREME COMMANDER developer and Square Enix partner Gas Powered Games is located in Redmond. Working out of Kirkland, Monolith Productions has produced a diverse catalog of shooters including SHOGO, F.E.A.R., and NO ONE LIVES FOREVER. The company's LithTech engine has been rebranded as the Jupiter EX game engine and is being developed by its subsidiary Touchdown Entertainment. Surreal Software (THE SUFFERING) works out of the area as does 5th Cell Media, creator of DRAWN TO LIFE and the recently released LOCK'S QUEST. The Seattle area is also home to several military sim developers including Zipper Interactive (SOCOM), and Zombie Studios (SPEC OPS, AMERICA'S ARMY).

JEFFREY FLEMING *likes coffee, but only on the weekends. Email him at* **jfleming@gdmag.com**.

Join the developer of F.E.A.R.™, Condemned 2: Bloodshot™, Tron® 2.0, No One Lives Forever™2, Aliens vs. Predator 2™, and Shogo: Mobile Armor Division.

NOW HIRING

LITH.COM/JOBS

MONOLITH™

Monolith is an Equal Opportunity Employer.
™ and ® indicate trademarks or registered trademarks of their respective owners. All rights reserved.

NOEL LLOPIS

DATA ALIGNMENT

Part 2: Objects on The Heap and The Stack

LAST MONTH WE LOOKED AT DATA alignment, how to align static variables, and how to allocate aligned memory on the heap through a simple custom allocator. It's a good start, but we need more than that to be able to use alignment effectively in a game.

This month, I'll wrap up this topic by looking at how to allocate any object on the heap with a given alignment and explore allocations on the stack.

Be warned that this is going to take some digging into the dark, dusty corners of the C++ language. So sit comfortably by the fireplace, grab your trusty copy of the C++ Standard, and let's get started.

ALIGNING OBJECTS ON THE HEAP

The main problem with `aligned_malloc` is that it simply allocates a block of data, and we often want to allocate full instances of classes or structs with a particular alignment.

In a pinch, we can turn to the underused and unloved placement new. Placement new works just like the regular `new` you're used to, but it takes an extra parameter with the memory address where the object will be instantiated. Placement new won't allocate the memory for you—it will just create an object of the right type where you tell it to and call its constructor.

Combining placement new and `aligned_malloc`, you could create an aligned waypoint object on the heap. See Listing 1.

It works, but that's some ugly code, not the kind of thing you want to see all over your codebase. Not only are we responsible for keeping around the aligned pointer so we can free it ourselves, but we have to call the object's destructor by hand.

A cleaner way to do it would be to create custom `operator new` and `operator delete` functions for the class. Those operators would be responsible for doing all the bookkeeping behind the scenes: hanging onto the pointer, calling the destructor, and freeing the memory with the correct function. If we implement this approach with our Waypoint class, Listing 1 would be reduced to:

```
Waypoint* waypoint = new (aligned_
malloc(sizeof(Waypoint), 16)))
Waypoint();
//...
delete waypoint;
```

That code is better, but we can simplify it further. Instead of passing the pre-allocated memory, we can simply pass the desired alignment, and let the custom `operator new` call `aligned_malloc`.

To free memory correctly, the corresponding `operator delete` also needs to call `aligned_free`. The allocation code is now:

```
Waypoint* waypoint = new (16)
Waypoint();
//...
delete waypoint;
```

That's much cleaner!

As we saw last month, memory allocated with `aligned_malloc` needs to be freed with `aligned_free`. Attempting to use the wrong `free` function on a block of memory will result in a heap corruption (and possibly some really hard to track down bugs). That means that `operator delete` needs to know how the object was allocated and decide which free function to call. Since `operator delete` can't easily access the contents of the object it's creating, we would have to allocate extra space and set some flags around the memory

LISTING.01
Aligned Waypoint Object Created on the Heap

```
void* buffer = aligned_malloc(sizeof(Waypoint), 16);
Waypoint* waypoint = new (buffer) Waypoint();
//...
waypoint->~Waypoint();
aligned_free(buffer);
```

NOEL LLOPIS *has been making games for just about every major platform in the last ten years. He's now going retro and spends his days doing iPhone development from local coffee shops. Email him at* nllopis@gdmag.com.

that was just allocated, indicating which function was used.

Alternatively, a simple solution is to overwrite the default `operator new` and call `aligned_malloc` from it as well, but without any alignment. That way all memory is allocated through `aligned_malloc` and `operator delete` can safely call `aligned_free` for all objects.

Since creating all those operators is just a bunch of busy work, we can wrap them in a macro called `DYNAMIC_ALIGNMENT`. The advantage of the macro over a base class implementing those functions is that we can use the macro in any class or structure without requiring any inheritance or changing the data layout in any way. Since the classes we want to align are often fairly small, low-level ones, that's a particularly important consideration. (See the code on www.gdmag.com for the full implementation).

FINDING THE ALIGNMENT

You still need to be very careful with the alignment of member variables. If you remember from last month, the C runtime will align a statically allocated structure based on the alignment properties of its member variables. However, if we allocate it on the heap, it's up to us to ensure its correct alignment.

When you dynamically allocate an object of the Waypoint class without specifying an alignment, the matrix member variable is not guaranteed to be on a 16-byte boundary. To make sure the matrix is aligned correctly, you need to remember to allocate the whole Waypoint object on a 16-byte boundary.

Having duplicate information in multiple places in the code is a common source of errors, so if we've already tagged a structure with a particular alignment for static allocation, it would be ideal to use it as the default alignment on the heap.

We can do that by querying the required static alignment for the data type we're about to allocate. Visual Studio provides the `__alignof()` operator and gcc has the equivalent `__alignof__()`, which returns the static alignment for any data type. We use this alignment as the default value when an object tagged with the macro `DYNAMIC_ALIGNMENT` is created and no other alignment has been specified. (See the source code on www.gdmag.com.)

THE ARRAY PROBLEM

At first glance, it seems that dealing with aligned arrays is just a matter of implementing `operator new[]` to return an aligned block of memory. That's almost true, but with a few nasty details thrown in.

In the abstract, an array is a data structure holding a group of elements that can be accessed by its index. There are many different ways in which such a data structure could be implemented, from very complex and abstract, to very simple and barebones ones. Some programming languages take the high route and implement arrays as a full-blow data structure, hiding the implementation details and providing extra checks to make sure that no element outside the current bounds is accessed. As you can imagine, that's not how they're implemented in C.

That's not a bad thing, though. By keeping to a minimalistic implementation, C provides top performance at the expense of some flexibility and safety checks. That mentality is a result of the hardware available when C was created, back when 10MHz was a state-of-the-art mainframe. It also means that even in this age of machines with gigaflops to spare, we can choose to make maximum use of the hardware resources.

C's philosophy has always been that you only pay for what you use, and if you want something more complex, you can always build it on top yourself (or grab someone else's implementation like std::vector).

In C, an array is a memory block that contains all elements stored contiguously in sequential order. That means you can access a particular element through its index by adding the index times the size of the element to the beginning of the array. What's more, if the index you're accessing is a constant, the calculation can even be done at compile time so there's virtually no overhead at runtime.

That's all very clever and efficient, until we have to deal with alignment. Array elements are laid out sequentially, without any empty space in between. If the struct in the array is a size that is a multiple of the alignment we want, then we're in luck. We can just align the beginning of the array and everything will work fine. But if that struct has a different size, then no matter how we

www.rtpatch.com

RTPatch and Pocket Soft are registered trademarks of Pocket Soft, Inc.

align the array, some of the elements are going to have the wrong alignment.

There are a variety of ways to deal with array alignments, some more complex than others. However, there is one method that is easy, fast, and has no overhead: Set the desired static alignment for the structure. That will force the compiler to pad the structure to a multiple of the alignment you requested. All elements in the array will still be sequentially laid out, but they will start at the correct alignment as long as we align the beginning of the array correctly with `aligned_malloc`.

The only downside of this approach is that it can lead to wasted space, because it will align and pad all structures of that type in the game, not just the ones that need to be aligned in the array. Similarly, if the same structure needs to be aligned on one boundary for one array, and on a different boundary in another array, you need to pad it to the least common multiple of the two alignments, meaning even more wasted space. To be fair, that's a very uncommon situation and the wasted space is probably insignificant. And if space really is a problem, you could always create a new struct that contains the original struct plus some padding and use that in the array. Problem solved.

The last "gotcha" with aligned dynamically-allocated arrays is that sometimes aligning the memory in `operator new[]` might not work at all. Yes, you heard that right. For our approach to work, the C runtime must place the array at the beginning of the block we took so many pains to align correctly. Unfortunately, the C++ standard allows the runtime to pass a larger size parameter to `operator new[]` and rearrange things a bit. We're back in the realm of platform-dependent behavior, but fortunately it's something that only seems to happen when creating arrays of objects with virtual functions. It's probably a bad idea to try to align data when vtables are involved anyway, so stick to simple structures and everything will be fine.

STACKING THINGS UP

The last allocation type we need to deal with is the stack. Even though the stack is one of the easiest memory allocation types to understand and work with, it's the one where ensuring alignment can be the hardest.

Since we have no control over where exactly in the stack our variables are allocated, you would think that the C runtime could help with alignment issues. The good news is that some recent versions of compilers do (like Visual Studio 2005 and later versions) by honoring the same alignment attributes we saw in the static allocation section. The bad news is that a lot of compilers won't respect those rules (like Visual Studio 2003). The even worse news is that they not only won't follow the alignment rules, but also will remain totally quiet and not print a single warning.

But wait! Things get even more complicated. In addition to local variables, the stack may also hold some of the parameters passed to functions and the value returned from functions as well (unless they can be optimized by using registers instead). How the registers are used, and how exactly the stack is laid out is called the application binary interface (ABI) and is described in a standard for each platform. Using the ABI, different programs and libraries can interoperate correctly and call into each other's functions.

Some platforms, like Windows x64 and PowerPCs, allow for stack parameters to be aligned correctly, but the ABI used in Win32 does not support parameter alignment. The only sure way to know how something will be aligned is to wade through the arcane documents describing the ABI for your specific platform.

If you can't rely on your target platforms to correctly align data and parameters on the stack, you can avoid the stack completely when you need to use aligned data and dynamically allocate data on the heap instead. It seems like a step backward to trade stack allocation for heap allocation, since heap allocation is much more complex, potentially expensive, and can lead to many problems of leaks and memory fragmentation.

A good compromise is to use a heap-based, fixed-size memory pool that implements correct alignment but behaves like a stack—data gets allocated and removed from the top. That way you get the benefits of a stack in fast performance, no fragmentation, and fixed bounds, but you can also enforce the correct alignment on all the variables.

To be on the safe side, you might want to prevent objects that require a particular alignment from being created on the stack at all. A simple way to accomplish that is to declare a private destructor and require programmers to call a `Destroy` method instead of using the destructor directly. If anybody attempts to create one of those objects on the stack, the compiler will catch it right away.

Another solution is to assert in the constructor that the following pointer is aligned correctly: `assert(IsAligned(this, 16));`. But it's not an ideal solution because it's a runtime check instead of a compile-time check. To make matters worse, it might not be triggered for a while because the object was accidentally allocated on the correct boundary, but it's much better than getting memory corruptions or exceptions when trying to use the misaligned data.

With these techniques in hand, you should be able to control where your data is placed. Maybe it will instill in you a healthy fear for the intricate details of C++, but it will also allow you to take full advantage of that hardware with those alignment restrictions, and hopefully bump your game into a solid 60fps. ⁙

The author thanks Jim Tilander for his feedback on some of the technical details.

STEVE THEODORE

WHO AMONG US SHALL BUILD THIS SHADER?

HAVE ARTIST-FRIENDLY SHADER TOOLS finally arrived?

Shaders can drive you crazy. Every new generation of hardware or slick new rendering technology tantalizes us with untold possibilities. Yet just when we fall in love with some new look, it's snatched away by engineers who tell us it's killing the frame rate or hogging memory.

Working as an artist in a shader-driven medium can be like supporting the Chicago Cubs. It's a slow-motion torture in which hope and excitement inevitably decay into disappointment and frustration.

We're entering a new round in this endless tango of technological enticement and frustration. The creative promise of shader tech has hovered just out of reach for artists for quite some time. We know many tactics for managing a complex visual appearance. It's not as if we've never heard of texture maps or compositing or multiplying pixel colors together. But actually putting together a working shader has always required unpleasantness like syntax-highlighting text editors and compilers. Many artists who are fully capable of imagining and even designing great shaders have been put off from actually building them by all these programmer-ish accoutrements.

HOW MUCH DO WE REALLY USE?

It looks as though things are changing for the better. The latest crop of modern shader creation systems tries to lure artists with UI and workflows that seem to come straight from the familiar Maya HyperShade playbook.

The material editor that ships with the Unreal Engine toolset, the Max plugin ShaderFX from Lumonix, and the combination of NVidia's FX composer and Mental Mill all provide graphic interfaces for creating shaders using well understood, artist-approved node graphs instead of the horrors of typing code for yourself. If you know your way around Maya HyperShade or DarkTree, for example, you can now find a shader authoring tool that looks pretty similar and will let you spit out fiendishly complex effects files that you would have never had the fortitude to type in manually. (Sigh.)

Unfortunately, despite this new convergence in tools, offline and online rendering remain very, very different. To achieve the millions of computations they must do every second, graphics chips evolved into the idiot savants of computing: amazingly powerful, but weird and off-putting.

It is absolutely true that the new generation of shader tools lets you create sophisticated effects with (relative) ease. What has not changed, though, is the awful truth that achieving reliable, sustained performance in the harsh world of online rendering still demands specialists. Despite the cool new GUIs, the intricacies of graphics hardware and the esoteric, special-purpose shading languages that drive the hardware are still the province of engineers.

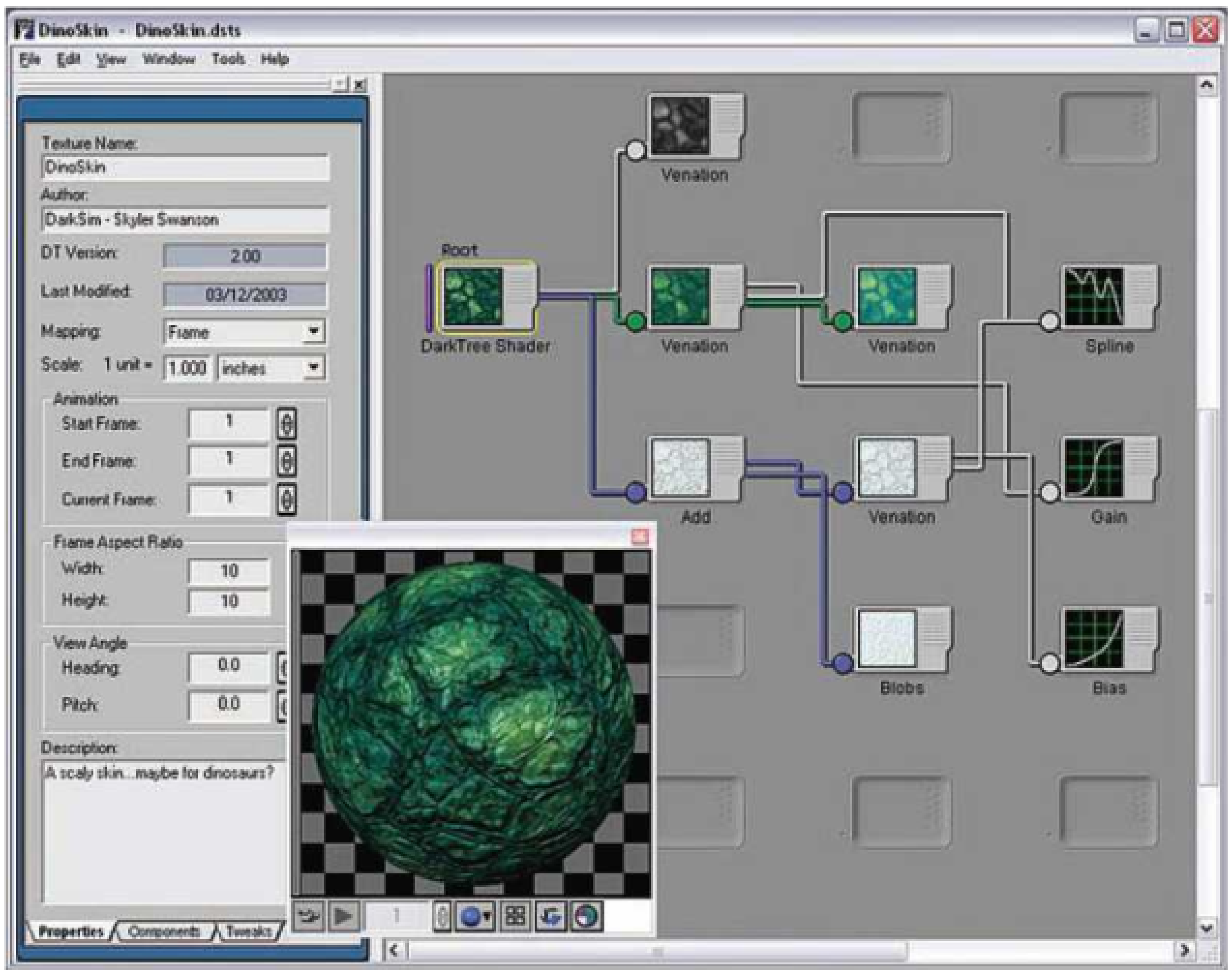
Turning shaders into just another form of "content," like a bitmap or a model, is an appealing idea. Unfortunately, that dream remains a long way off. Many teams have learned the hard way that giving artists the whole responsibility for shader creation is dangerous for at least three reasons.

First, even with the "friendly" face of a modern shader editor, many artists are still intimidated by shaders. Open-ended systems are very powerful in the hands of technically-inclined artists with a lot of patience and the will to learn. For many production line folks with deadlines to worry about, though, they seem like a waste of time. If all you want really is a standard bumpy-shiny-Phong clone, creating it for yourself out of nuts and bolts isn't a plus—it's a drag. If you're going to reuse the same pieces over and over, wouldn't it be better if they were custom tailored and optimized by a professional graphics engineer instead of cobbled together by a harried artist?

Second, the artists who do embrace the tech and run with it are still going to run head-on into performance issues. It's torture to give an artist a tool that invites—even demands!—artsy experimentation and neat little extra touches, only to swoop in at the end of the project and rip them all out because some combination of graphics calls is causing the game to stutter like a 1978 AMC Pacer. Even if all the shader artists are fairly good at performance programming (a big "if"), a library of hand-built shaders is still very difficult to rationalize when the project nears its ship date, so last-minute performance tweaks will be haphazard and risky.

Finally, and most damning, letting every artist build his or her own shaders

STEVE THEODORE *has been pushing pixels for more than a dozen years. His credits include MECH COMMANDER, HALF-LIFE, TEAM FORTRESS, and COUNTER-STRIKE. He's been a modeler, animator, and technical artist, as well as a frequent speaker at industry conferences. He's currently content-side technical director at Bungie Studios. Email him at* ***stheodore@gdmag.com****.*



DinoSkin - DinoSkin.dsts

Texture Name: DinoSkin

Author: DarkSim - Skyler Swanson

DT Version: 2.00

Last Modified: 03/12/2003

Mapping: Frame

Scale: 1 unit = 1.000 inches

Animation

Start Frame: 1

End Frame: 1

Current Frame: 1

Frame Aspect Ratio

Width: 10

Height: 10

View Angle

Heading: 0.0

Pitch: 0.0

Description: A scaly skin...maybe for dinosaurs?

The Texture Tree Editor in Darkling Simulations' DarkTree procedural shader authoring tool is shown.

can mess with your art direction. If, by some miracle, your artists manage to traverse the minefield of performance without blowing up the game, they'll reach the end of production with hundreds of materials created in very personal ways. It's possible that my shiny metal and your shiny metal will look similar in the shader editor and yet behave altogether differently when dynamic lights, moving characters, or HDR lighting kick in.

Artist-built shaders offer no common way to herd these divergent shaders into the same dynamic range. If artist A's shaders are always much hotter than artist B's under the same lighting, how can you make them look like they come from the same physical universe without editing dozens or hundreds of complex node graphs?

COOPERATION STATION

All of this should not be taken as a slam on GUI shader tools. Graph-based shader authoring is a huge step in the long, slow process of educating ourselves about how our medium works. By providing artists with tools that let them sketch out the way they'd like their shaders to work, graphics shader editors make a huge contribution to the thorny artistic problem of imagining and then realizing complex visual appearance. If you're setting out to prototype a new effect or custom material for your next game, these tools are ideal places to start.

The important caveat, though, is that you can't assume that shaders are now a "done" topic like bitmaps that can be safely left to the art department. There are too many complex interactions between shaders and the rest of the game, and too many difficulties (as we've seen) involved in managing a shader library that was created by non-engineers.

WELL, WHAT THEN?

If we artists aren't going to write our own shaders, what can we do to make sure the shaders we get are going to do what we want and tell the stories we want to tell? Assuming the engineers can produces a good shader framework, what features should artists be lobbying for to make it a creative and reliable tool?

Here are a few suggestions to keep in mind the next time you have engineers asking you for input on a shader system.

DON'T OVERWHELM ME

Many non-technical artists are turned off by the sheer number of choices they have in their shader tools. A good system should make it easy for line artists to concentrate on the jobs they know best and do most—modeling and texturing. Don't make them hack through graphics options they don't understand.

The first tool for managing this complexity is to offer a smaller number of high-level choices. A library of several basic shaders with contrasting material qualities is better than a single "uber-shader" with dozens or hundreds of sliders. Yes, it is more limiting, but for most artists, that limitation is a stress-reducing bonus, not a problem.

Non-technical artists will be happier with, say, dedicated shaders for metal, plastic, and wood than with a text doc detailing the dozens of parameter settings needed to get such different effects out of a single shader. A good library system with strictly limited tweakability also helps junior artists produce materials that behave in predictable ways, and makes it easier for engineers to do performance adjustments late in the game, since the shaders are already grouped by function.

While designing the library system, it's also a good idea to build in a rough measure of shader cost to each of the library shaders. Engineers will tell you over and over that they can't give you the "real" performance cost of a given shader, because there are so many factors involved. This is partially why having artists build shaders for themselves is impractical. However, relative costs are fairly easy to estimate. A three-pass shader is probably a lot slower than a single-pass one. A shader with anisotropic textures will be a lot slower than one with bilinears, and so on.

If your library shaders prominently display a relative cost factor (even if it's nothing more than fast, medium, or slow) it will help educate less-technical artists about the performance issues they have to deal with and hopefully head off last-minute cutbacks as production winds up.

EXPERT MODE

Of course, some artists will look at a library system like a straightjacket. They're a minority, but an important one.

If you have high-powered technical artists, you might allow them to create shaders directly for some limited purposes using tools like those above, or you could simply expose more of the inner workings of your underlying shader system to them. In either case, it's important to make sure that "expert mode" really is reserved for special cases.

Whether the choice to use custom shaders should be made by art leads, or handed off to tech artists, or enforced by the honor system alone is a question that depends more on company culture. The important thing is to make sure that the "expert mode" doesn't morph into "everybody" mode. If a lot of artists are clamoring for a particular look or behavior, it ought to be packaged up (with necessary performance tweaks and budget constraints) into a library shader instead of passed around the office by copying and pasting settings nobody understands!

CONSISTENCY

The most difficult goal to achieve in designing a shader system is consistency. Even in a streamlined, fill-in-the-blanks library system, there are plenty of ways to make objects that look quite similar in some lighting conditions and wildly different in others. Unfortunately, this is one of those problems where sociology trumps technology. While you can build a system that encourages consistent approaches to lighting and texturing, it's impossible to make one that will enforce consistency 100 percent of the time.

One important thing you can do to foster consistency is to build the lighting preview environment right into your tools. If artists drop a copy of their shader into the game to test it in the "real" environments, you can bet they'll find lighting backdrops that suit their tastes rather than ones that are really representative of typical in-game lighting. If your shader editor comes with well-maintained lighting stages that accurately reflect standard lighting, you'll have a huge leg up on making shaders that look like they all belong in the same game.

Of course, maintaining consistent lighting across the whole game is itself a dicey experiment in art management, but we'll leave that for another discussion.

Another very helpful step you can take is to name your shader controls in ordinary English. Shader technology over-stimulates that part of the programmer brain that delights in obscurity. For example, many shiny shaders control their highlights with two values. The first is typically named something like "specular power" and runs from 2 to 32 (that's how many times you multiply the diffuse lighting falloff by itself to get the falloff of the specular lighting). The second, typically named something like "specular coefficient," is multiplied against the result of the first, but runs from 0 to 1 for natural surfaces or maybe higher than 1 for unusual effects. Not only are these names pretty incomprehensible to your average artists, but the value ranges don't mean anything intuitive. You can get similar looking results from many combinations of these two values. A coefficient greater than 1 combined with a high power often looks much like a lower power value with a smaller coefficient. The result is confusion for newbie artists and headaches for the lead who has to figure out why some shiny shaders white-out under HDR lights and others don't.

UNDERSTANDING LEADS TO EXPERIMENTATION

Obviously, designing a shader system from the ground up is a daunting task. The engineering is tricky and has high demands, but the social side—the education and training and development of a house style—is also very difficult.

It's the human element that really counts. Very few artists are really pushing the limits of current shader tech because so few of us really know what the medium can do. While we are not at the point where shading is the personal responsibility of an ordinary artist in the same way texturing is today, we do need to keep enlarging our understanding of the underlying tech so we can tell the stories we want to tell.

The new generation of shader tools is not going to have your graphics engineers scanning the want ads any time soon, but it can do a lot to encourage artists to learn more about the possibilities and limits of our ever-changing medium. Even if all these new tools do is eliminate some of the aura of mystery around shaders, they will have earned a valuable place in the artist's toolbox. ⁙

GAME CAREER GUIDE

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

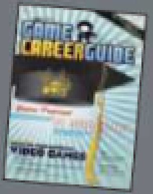
NEW!

DIGITAL COUNSELOR

I'll match your interest and goals with the right game related programs and schools from around the world.

Sponsored by
The Art Institute of Pittsburgh
Online Division

Download your FREE digital edition of the 2008 gamedeveloper Game Career Guide online at:
www.gamecareerguide.com



SOREN JOHNSON

SID'S RULES

MOST GAME DEVELOPERS ARE FAMILIAR with Sid Meier's dictum that "a good game is a series of interesting choices." In fact, my co-columnist Damion Schubert started his recent article on player choice (*Designing Choice*, October 2008) by referencing this famous quote.

However, over the course of his career, Meier has developed a few other general rules of game design, which I heard him discuss many times during my seven years (2000–2007) at his studio, Firaxis Games. As these insights are quite practical lessons for designers, they are also worth discussing.

DOUBLE IT OR CUT IT BY HALF

Good games can rarely be created in a vacuum, which is why many designers advocate using an iterative design process—build a simple prototype of the game very early and iterate it repeatedly until the game becomes a shippable product.

Meier calls this process "finding the fun," and the probability of success is often directly related to the number of times a team can turn the crank on the loop of developing an idea, play-testing the results, and then adjusting it based on feedback. Because the number of times a team can go through this cycle is finite, developers should not waste time with small changes. Instead, when making gameplay adjustments, developers should aim for significant changes that will provoke a tangible response.

If a unit seems too weak, don't lower its cost by 5 percent; instead, double its strength. If players feel overwhelmed by too many upgrades, try removing half of them. In the original CIVILIZATION, the gameplay kept slowing down to a painful crawl, which Meier solved by shrinking the map in half. The point is not that the new values are likely to be correct; the goal is to stake out more design territory with each successive iteration.

Imagine the design space of a new game to be an undiscovered world. The designers may have a vague notion of what exists beyond the horizon, but without experimentation and testing, these assumptions remain purely theoretically. Thus, each radical change opens up a new piece of land for the team to consider before settling down on the final product.

ONE GOOD GAME IS BETTER THAN TWO GREAT ONES

Meier likes to call this design principle the COVERT ACTION Rule, a reference to a not altogether successful spy game he made in the early 1990s. Meier was once quoted as saying:

"The mistake I made was actually having two games competing with each other. There was an action game, where you break into a building and do all sorts of picking up clues and things like that, and then there was the story, which involved a plot where you had to figure out who the mastermind was and what cities they were in, and it was an involved mystery-type plot.

Individually, each part could have been a good game. Together, they fought with each other. You would have this mystery that you were trying to solve, then you would be facing this action sequence, and you'd do this cool action thing, and you'd get out of the building, and you'd say, 'What was the mystery I was trying to solve?' COVERT ACTION integrated a story and action poorly because the action was actually too intense—you'd spend 10 minutes or so of real time in a mission, and by the time you got out, you had no idea what was going on in the world."

In other words, even though both sections of the game were fun on their own, their coexistence ruined the experience because the player could not focus her attention on one or the other.

This rule points to a larger issue. All design choices only have value in relation to one another, each coming with its own set of cost-benefit trade-offs. Choosing to make a strategic game also means choosing not to make a tactical one. Thus, an idea may be "fun" on its own, but it still might not make the game better if it distracts the player from the target experience. Indeed, this rule is clearly the reason why the CIVILIZATION franchise has never dabbled with in-depth tactical battles every time combat occurs.

However, sometimes multiple games can coexist in harmony. SID MEIER'S PIRATES! is an example of a successful game built out of a collection of fighting, sailing, and dancing mini-games. However, these experiences were always very short, a few minutes at the most, leaving the primary focus on the meta-game of role-playing a pirate. Each short challenge was a tiny step along a more important and larger path of plundering all Spanish cities or rescuing long-lost relatives.

Another example of a successful mix of separate sub-games is X-COM, which combined a tactical, turn-based, squad-level combat game with a strategic, real-time, resource-management game. As with PIRATES!, X-COM works because the game design has one focus, in this case, the compelling tactical battles between the Marines and the invading aliens. The high-level, strategic meta-game exists only to provide a loose framework in which these battles—which could take as long as a half hour each— actually matter. The player doesn't fight the aliens to get to manage resources later; instead, he manages resources to get to perform better (and have more fun) in future battles.

DO YOUR RESEARCH AFTER THE GAME IS DONE

Many of the most successful games of all time—SIMCITY, GRAND THEFT AUTO,

SOREN JOHNSON is a designer/programmer at EA Maxis, working on an unannounced project. 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. Email him at sjohnson@gdmag.com.

CIVILIZATION, ROLLERCOASTER TYCOON, THE SIMS—have real-world themes, which broaden their potential audience by building the gameplay around concepts familiar to everyone. However, creating a game about a real topic can lead to a natural but dangerous tendency to cram the product full of bits of trivia and obscure knowledge to show off the amount of research the designer has done.

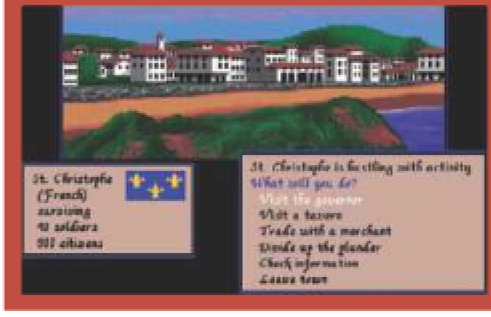
This tendency to showcase knowledge spoils the very reason real-world themes are so valuable: Players come to the game with all the knowledge they already need. Everybody knows that gunpowder is good for a strong military, that police stations reduce crime, and that carjacking is illegal. "The player shouldn't have to read the same books the designer has read in order to be able to play," Meier says.

Games still have great potential to educate, just not in the ways of standard pedagogical practices. While designers should still be careful not to include anything factually incorrect, the value of an interactive experience is the interplay of simple concepts, not the inclusion of numerous facts and figures.

Many remember that the world's earliest civilizations sprang up along river valleys—the Nile, the Tigris/Euphrates, the Indus—but nothing gets that concept across as effectively as a few simple rules in CIVILIZATION governing which tiles produce the most food during the early stages of agriculture. Furthermore, once the core work is done, research can be a very valuable way to flesh out a game's depth, perhaps with historical scenarios, flavor text, or graphical details. Just remember that learning a new game is an intimidating experience for players, so don't throw away the advantages of an approachable topic by expecting the audience to already know all the details when the game starts.

THE PLAYER SHOULD HAVE THE FUN, NOT THE DESIGNER OR COMPUTER

Creating story-based games can be an intoxicating experience for designers, many of whom go overboard with turgid backstories full of proper nouns, rarely-used consonants, and apostrophes.



TOP: Civilization IV (2K Games, 2005)
FAR LEFT: Sid Meier's Pirates! (MicroProse Software, 1987).
LEFT: Civilization (MicroProse Software, 1991).

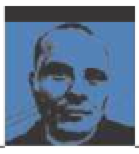
Furthermore, games based on complex, detailed simulations can be especially opaque if the mysterious inner workings of the algorithmic model remain hidden from view.

As Meier likes to say, with these games, either the designer or the computer is the one having the fun, not the player.

For example, during the development of CIVILIZATION IV, we experimented with government types that gave significant productivity bonuses but also took away the player's ability to pick which technologies were researched, what buildings were constructed, and which units were trained, relying instead on a hidden, internal model to simulate what the county's people would choose on their own. The algorithms were, of course, very fun to construct and interesting to discuss outside the game. The players, unfortunately, felt left behind. The computer was having all the fun. So we cut the feature. Games require not just meaningful choices, but also meaningful communication to feel right. Giving players decisions that have consequence but which they cannot understand is no fun.

Role-playing games commonly fail at making this connection, such as when players are required to choose classes or skills when "rolling" a character before experiencing even a few seconds of genuine gameplay. How are players supposed to decide between being a barbarian, a fighter, or a paladin before understanding how combat actually works and how each attribute performs in practice? Choice is only interesting when it is both consequential and informed.

Thus, in Meier's words, the player must "always be the star." As designers, we need to be the player's greatest advocates during a game's development, always considering carefully how design decisions affect both the player's agency in the world and his understanding of the underlying mechanics.



JESSE HARLIN

WORKING IN CONCERT

Designing Your Audio Documents Deliberately

GAME COMPANIES ARE NOT JAM BANDS. They're orchestras. While still a creative enterprise, games are a business driven by billions of dollars, and billions of dollars are never efficiently spent via "make it up as we go along" approaches to business. When it comes right down to it, uncontrolled creativity costs companies money. As such, AAA game design requires deliberate boundaries and goals established early in the pre-production process and communicated throughout the design team to a variety of different audiences with different needs.

Just like all of the other disciplines, the audio lead will be required to create a slew of design documents—from spreadsheets to bar graphs, from wikis to Word docs—which define all the nuanced needs of the game's audio components. Not all of this information is helpful to all of those who will read it, however. Knowing in advance who needs what information will help you to more clearly communicate your needs and give you a better chance of succeeding without being tied down by a deluge of email clarification requests.

KNOW YOUR AUDIENCE

The sum total of the audio documentation is like a conductor's score. Every last detail is noted and all intentions specifically spelled out at the beginning of the process. For the audio lead, all these separate elements will combine to help them lead diverse groups through different parts of the audio process toward a single finished goal.

However, harp pedal notation doesn't mean much to piccolo players. Similarly, what might matter to an audio engineer isn't necessarily going to be helpful for the game's composer. When creating pre-production audio documentation, the audio lead must keep in mind who will be reading which sections or documents and write them accordingly.

Arguably, the most important audience at the earliest design phase is producers and other production staff. These will most likely be people with absolutely no background in audio, but who control the checkbook. A producer's primary concern with all audio documentation is going to be cost, while time and schedules are a related secondary concern. Producers will want to see staffing plans and a breakdown of man-hours over the life of the project. They'll want to see budget documents detailing SFX or sample library costs, potential equipment or field recording costs, and estimates for outside contractors.

Additionally, producers will want to see recommendations regarding the game's audio tech. They'll want to know if the audio lead is planning on using existing technology as-is or if there are engineering changes needed to suit the game's design. If no tech exists, they'll want to know the audio lead's suggestions regarding the purchase and implementation of middleware versus building proprietary tools from scratch, and how many engineering resources will be required to accomplish these goals.

Engineers will also make extensive use of the early audio documentation. As expected, though, an audio engineer's concerns are very different than that of a producer. For engineers, it all comes down to a single question: What am I expected to build? While a producer might be interested in an outlined technical overview, engineers are going to need extensive descriptions of very specific software requirements. The more detailed the documentation is, the less chance there is for technical ambiguity. Elaborate on plans for improved distance limiting, instance culling, or dynamic bank loading. Discuss file formats or DSP architecture. Clearly define the ins and outs of the game's interactive music system or adaptive voice engine. The more technical you are, the better starting point you'll find yourself in for getting what you ultimately need from the engineering staff.

The final segment of your audience is the sound designers, voice editors, and composers who make up your audio staff. These people are looking at your documentation to learn your aesthetic vision of the game. Producers will also be interested in this material as they ensure that your creative goals fall in line with those of the rest of the team, the previous games of a given franchise, or the source license the game reflects.

Audio content creators will want a style guide explaining how realistic the audio for the game must sound, and citing any reference materials that might serve as signposts for their creative work. Some amount of technical explanation regarding the in-game audio tech may be needed, but detailed tech breakdowns will serve mostly as a curiosity for the audio content staff rather than necessary information. Make sure that the description of the interactive music system you hand to your external composer isn't the same one you hand to your audio engineer. While both may need to know about sample-accurate beat syncing, your composer won't need extensive documentation on stream management any more than your audio engineer needs suggestions on orchestration goals or key signature planning.

CONDUCTING BUSINESS

Always keep in mind that while deliberate documentation is always necessary, it should never be an inflexible hindrance. The game's design will inevitably evolve through the iterative process of creative exploration. As such, let your audience know up front that they're reading a "living document;" one that will most likely see revisions and expansions as development of the game continues. ⁙

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

EDGE OF REALITY

NOW HIRING

Producer
Project Manager
Lead Game Designer
Lead Level Designer
Project Art Director/Lead Artist
Studio Art Director
Special Effects Artist
Senior Engine Programmer

More information can be found at:
www.edgeofreality.com

mary-margaret
network

Find. Grow. Work. Play.

The destination for recruitment, staffing and career services in the global interactive entertainment industry.

It all begins here...

www.mary-margaret.com

Create the Game

Gaming Degree Programs for the Next Generation

Game Art
Bachelor's Degree Program

Game Development
Bachelor's Degree Program

Game Design
Master's Degree Program

ANIMATION | DESIGN | ENTERTAINMENT BUSINESS | FILM | RECORDING ARTS | SHOW PRODUCTION | VIDEO GAMES | WEB

Master's | Bachelor's | Associate's Degrees

800.226.7625 • 3300 University Boulevard • Winter Park, FL 32792

Financial aid available to those who qualify • Career development assistance • Accredited University, ACCSCT

fullsail.edu

Online Programs Available

FULL SAIL UNIVERSITY.

VFS

GAME DESIGN EXPO

FEBRUARY 7-8, 2009 VANCOUVER

gamedesignexpo.com

G4 TECH TV · EA · UBISOFT · BIG FISH Games · ANNEX PRO · OBSIDIAN · RADICAL ENTERTAINMENT

straight · Vicarious Visions · EPIC · BIGPARK · METROPOLITAN HOTELS · NEXON Humanature · LUCASARTS · INSOMNIAC · VANCOUVER · relic · PROPAGANDA GAMES

CENTER FOR DIGITAL IMAGING ARTS AT BOSTON UNIVERSITY

WALTHAM, MA | WASHINGTON, DC

CREATE your WORLD

cdia BOSTON UNIVERSITY

CERTIFICATE PROGRAMS | 3D ANIMATION+INTERACTIVE MEDIA
GAME ART+CHARACTER ANIMATION

Intensive full- and part-time programs for the skills and tools you need to turn your ideas into reality. Financial assistance and career services available. ***Apply today!***

CALL **800-808-CDIA** EMAIL **INFO@CDIABU.COM** WEB **WWW.CDIABU.COM**

GET EDUCATED

[GEEKED AT BIRTH.]

University of Advancing Technology UAT Learn. Experience. Innovate.

You can talk the talk.
Can you walk the walk?
Here's a chance to prove it.
Please geek responsibly.

LEARN:

DIGITAL ANIMATION	**GAME PROGRAMMING**
DIGITAL ART AND DESIGN	NETWORK ENGINEERING
DIGITAL VIDEO	NETWORK SECURITY
GAME DESIGN	SOFTWARE ENGINEERING
ARTIFICIAL LIFE PROGRAMMING	WEB ARCHITECTURE
COMPUTER FORENSICS	ROBOTICS

www.uat.edu > 877.UAT.GEEK
877.828.4335

ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
3Vis	31	Perforce Software	6
Center for Digital Imaging	47	Pocket Soft	37
Edge of Reality	45	Rad Game Tools	CV4
Epic Games	21	Seapine Software	15
Full Sail Real World Education	46	Trinigy	CV3
Havok	CV2	Unity Technologies	3
Intel	25–30	University of Advancing Technology	47
Mary-Margaret.com	45	Vancouver Film School	46
Monolith Productions	35	Vicious Cycle Software	11

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. For back issues write to *Game Developer*, 4601 W. 6th St. Suite B, Lawrence, KS 66049. 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.

MATTHEW WASTELAND

ARRESTED DEVELOPMENT

DUBIOUS PITCHES

THE FRIENDLY MANDATORY CRUNCH

Starting today, we're going to expand our core hour definition to mean 24 hours a day, 7 days a week. And since it's not fair to everyone if only certain people stay, we need the entire team to be here during that time. I know that this is probably a disappointment to some of you, but it is an unfortunately necessary measure to make our game the best it can possibly be and hit our shelf date. It's not all doom and gloom, though. To make the next six months a little more tolerable and fun, we will be ordering *pizza*!

THE ULTIMATE MONOLITHIC ENGINE

The Extraneous Engine is the most powerful, most technologically advanced and easiest to use video game engine ever conceived of by a sentient mind—human or otherwise. With this, your game will practically make itself! All you have to do is make art and put it in the "assets" directory. The Extraneous Engine does the rest, automatically arranging these pieces into a gamelike experience. Your artists and designers will fall in love with it, and you can save overhead by strategically eliminating all your engineers! Who wouldn't want to put the overwhelming potency of the Extraneous Engine to work for them?

THE SURE BET STARTUP OPPORTUNITY

This is your chance to get in on the ground floor of the next big thing! You will have the opportunity to work long hours doing what you love, and will be paid competitively (to third world countries) and then just as soon as our WORLD OF WARCRAFT killer and gamer-centric social network comes out, we'll all be rich!

THE FLAWLESS PROJECT MANAGEMENT METHODOLOGY

From out of the depths of chaos and darkness, the silver bullet appears—the one that will finally kill the werewolf that is modern video game development! This heaven-sent savior, called Scrimmage, is now here to fix everything that was bad about making games in those bygone days of madness and despair. Behold now, the radiant beams of knowledge and illumination that pour forth from this book on Scrimmage—available to you for a special discount if you sign up for our seminar at the same time! Act now while supplies last!

THE PUBLISHER THAT REALLY CARES ABOUT YOU

Hey, we both know that great games are good for business. Mediocre games for cheap are even better for business—but let's forget that for a moment. The point is that we care about you! You, the creative guys, the makers of all of this stuff we put in a box and sell. And of course we are all about letting you go do your creative thing, as long as we can make money from it. And as long as you stay inside this here "creative box" we've specially constructed for you. That ought to be enough creative freedom for anybody! Sure, we may occasionally need to make some creative calls for you here and there, but don't worry. We'll only do that if you make the wrong decision. We trust you, really!

THE PERFECT OUTSOURCEE

Our people are more than happy to work twenty times as hard as your people, and for twenty times less money! We have no overhead costs because we make everyone buy their own workstations and hand-crank their own electricity! We have also hired local psychics, who will work tirelessly to read your mind so we can give you exactly what you want every single time! We are a rock-solid stable company with tons of cashflow! Sorry I can't write more, but the Internet café is closing! Talk to you soon!

THE PAINLESS SUBSTITUTION

We should just scrap what we have and start over. I'm pretty sure I could re-write the particle system in, oh, say, two weeks.

THE COMPLETELY SEAMLESS MIDDLEWARE

Our middleware will slide in to your existing tools and pipeline so easily, you'll have trouble remembering that you integrated it at all. One day you'll be working on the game and just say, "hey, where did all of this awesome functionality come from?" And our APIs are so transparent and well-documented that you will never, ever wonder what our code is doing. And you will never, ever need to stay up until two in the morning on a Sunday to debug some weird memory leak that originates from the non-obvious way you implemented our product.

THE SMOOTH OPERATOR PRODUCER

Stop worrying. I've seen this kind of thing before and I know exactly how to handle it. I guarantee it will not be a problem.

MATTHEW WASTELAND is a pseudonymous game developer who has a fairly common first name. Email him at mwasteland@gdmag.com.

TRINIGY
www.trinigy.net
visit us at GDC 09
"A perfect fit for us."
Dietmar Meschede, Head of Programming at UBISOFT® Studio BlueByte

Vision Engine 7 - leading technology for leading platforms

XBOX 360 · XBOX LIVE arcade · PLAYSTATION®Network · Tools & Middleware PLAYSTATION®3 · Wii · Games for Windows

www.trinigy.net

RAD GAME TOOLS PRESENTS

MILES

Sound System

COMPLETELY HANDLES ALL ASPECTS OF SOUND

WWW.RADGAMETOOLS.COM/
MILES

MULTI-CHANNEL DIGITAL AUDIO, INTERACTIVE MIDI,
RED BOOK CD AUDIO, 3D AUDIO, INTERNET
VOICE CHAT, DECOMPRESSION, EAX, AND MORE!

TUNE IN AT:
425-893-4300

MP3,
OGG, ADPCM
DECOMPRESSION

RAD GAME TOOLS

ALL PLATFORMS:
PS3, XBOX 360,
Wii, PC, MAC

GRANNY 3D　BINK VIDEO　PIXOMATIC　MILES