United Business Media

POSTMORTEM TWISTED PIXEL'S THE MAW

DECEMBER 2008

gamedeveloper

THE LEADING GAME INDUSTRY MAGAZINE

INTERVIEW
YUJI NAKA TAPS HIS CREATIVITY

AURAL FIXATION
COMPRESSING SOUND FOR THE IPHONE

GAMESCAPE
A HISTORY OF BOSTON GAME DEVELOPMENT

gamedeveloper FRONTLINE FINALIST '07

"ReplayDIRECTOR rocks. I doubt we'd have found it otherwise. It turned out to be an occasional array overwrite that would cause random memory corruption…"

***Meilin Wong**, Developer, Crystal Dynamics*

BUGS. PETRIFIED.

RECORD. REPLAY. FIXED.

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

DEEP RECORDING. NO SOURCE MODS.

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

REPLAY SOLUTIONS
accelerating you to market

1600 Seaport Blvd., Suite 310, Redwood City, CA, 94063 - Tel: 650-472-2208 Fax: 650-240-0403

EA ELECTRONIC ARTS™ CRYSTAL DYNAMICS eidos XBOX 360 nVIDIA. XBOX

gamedeveloper

CONTENTS

DECEMBER 2008
VOLUME 15, NUMBER 11

16

FEATURES

7

14

POSTMORTEM

DEPARTMENTS

COVER ART: JOE MITCH

COLUMNS

A CHRISTMAS STORY

BY NOW, YOU WILL LIKELY HAVE EITHER FINISHED or just be finishing your holiday crunch. If you're not finished, well, I'm sorry for you. And if you haven't had to crunch at all, maybe you should write us an article.

Our main feature this month is about common things that have gone wrong with game development, and most of them tie in to some degree with crunch and schedule problems. One of these is often the cause or the effect of the rest. It's easy to see how this ties in to the holiday video game retail meat market.

'TIS THE SEASON TO MAKE MONEY

Certainly the holidays are a big spending period, so it's no wonder that publishers, and even developers, want to get their big titles out and in the minds of consumers round about that time. Shareholders, too, want to see or hear things like "holiday blockbuster" in order to keep their investor confidence, and I do think this is a very large part of why we have a holiday glut.

But ultimately, is this really helpful? Consider a game like DEAD SPACE. It's a good one, certainly, and its SYSTEM SHOCK-related pedigree gives it even more street cred. But what it isn't is a holiday blockbuster. I want to be very careful with my phrasing here, because I am absolutely not saying it can't stand toe to toe with the other releases. But when you have a third person game in which the primary action is shooting, it would behoove you not to release it against GEARS OF WAR 2. People are *waiting* for GEARS OF WAR 2, and you know that they are. They are not necessarily waiting for DEAD SPACE.

Had DEAD SPACE come out during a softer period, it might have had a chance to be as much of a financial success as it was a critical success. I don't know the final numbers on it, but I know it wasn't on the charts long, which is related to another major problem. One reason that so many companies feel they have to release during the holiday period is because if they don't, it's felt that consumers won't remember the game when that holiday rolls around. And they're kind of right, because games don't stay in the primary shelves for that long, partially because so many games are released during the holiday rush. It seems to me that if major releases were more evenly spaced, everyone would benefit.

YOU'LL PUT YOUR EYE OUT

Meeting a holiday schedule has its obvious problems. Unless a game is very specifically planned out to release at that time, there's a strong chance you're going to be crunching, rushing, or otherwise cutting corners to get it out the door. Granted this can happen in projects regardless, but rushing a product with the specific intent to compete in the most difficult, most cut-throat period of the year is not very healthy for you or for the game.

There are only so many dollars consumers have to spend, and the problem doesn't just come up during the holidays. DEAD SPACE released against GEARS OF WAR 2, and FAR CRY 2 released against FALLOUT 3—but DARK SECTOR also released against GTA IV, and that's got nothing to do with holidays.

Naturally it's hard to predict a release window, but the holiday season is a known quantity. You know everyone will be bringing their best and brightest.

THE NEW KIDS

I don't want to sound like I'm arguing against original IP. I could see reading this editorial in that light, and it's quite far from what I mean, given my love for gameplay experimentation. Original IP is obviously risky though, there's no denying it. And returning to the DEAD SPACE example, or even to MIRROR'S EDGE, it does speak well of EA's new direction that the company deemed its original IPs important enough to launch them against better-understood market movers.

But I do wonder if it was worth pushing them to meet this three-month holiday window. Even FAR CRY 2, which was a sequel, didn't really have the mindshare required to survive in this holiday climate. You shouldn't have to "survive," but that's often what it amounts to when everyone is scrambling for the same dollars at the same time.

Couldn't the company actually make more money by releasing these games at a different time, all the while increasing quality? I'm far from a financial genius, so it's possible I'm totally off-base here, but it seems like a less ambitious release window for original or lesser known IP would yield better results, so long as doing that doesn't relegate it to lower status within the company. In fact, it seems to me it could even raise its profile and importance. As per usual, I welcome your thoughts and criticisms.

–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

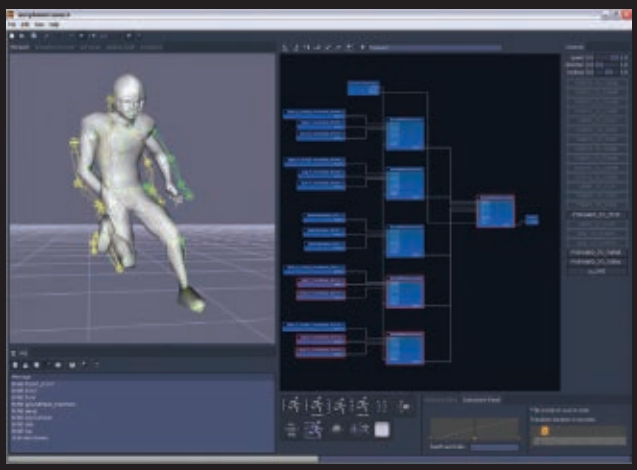
AUDIENCE DEVELOPMENT
GROUP DIRECTOR Kathy Henry e: khenry@techinsights.com
DIRECTOR Kristi Cunningham e: kcunningham@techinsights.com
LIST RENTAL Merit Direct LLC t: 914.368.1000

MARKETING
SERVICES MARKETING DIRECTOR Karen Tom e: ktom@think-services.com
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

morpheme™
advanced animation system



morpheme is the industry's first graphically authorable animation engine. morpheme consists of morpheme:runtime: an advanced runtime animation engine for PLAYSTATION®3, Xbox 360™, Wii™ and PC. morpheme:connect: a highly-customizable 3D authoring application.

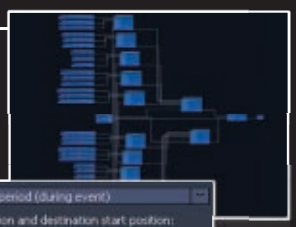
morpheme gives animators and developers unprecedented control over the look and feel of their animations in-game: blends, transitions, compression, etc. can all be previewed and modified graphically in morpheme:connect and live on the target platform.

morpheme:runtime ships with full source code and integrates seamlessly with euphoria, NaturalMotion's Dynamic Motion Synthesis technology.

For more information, visit www.naturalmotion.com

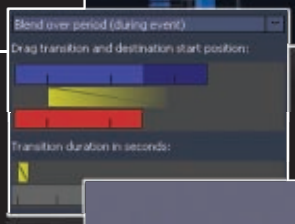
blend tree

Advanced graphical tools for building complex blend trees. Real-time visualization of animation source contribution through node highlighting



blending

Graphical control of transition blending between states in the transition graph



multiple characters

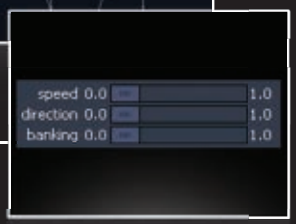
Visualization of multiple runtime characters in morpheme:connect for easy authoring and analysis of character interaction

network

Advanced graphical tools for creating and visualizing transition networks through drag-and-drop

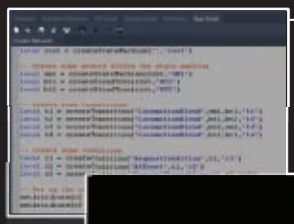
control parameters

Exposure of custom high-level controls for entire animation system. Real-time manipulation through sliders or game pad controller



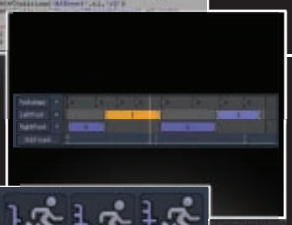
scripting

Full Lua scripting for automating tasks, adding AI logic or polling game pads for real-time input



timeline

Graphical mark up of animation data to add one-shot and duration events, for highlighting footfalls, sound effects, etc.

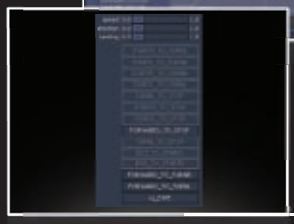


node palette

Advanced blend notes for dragging and dropping into transition network. Fully customizable node types through C++ and scripting

animation browser

Easy browsing and selection (drag & drop) of source animation. Animation list is automatically updated to reflect changed source files



transition requests

Exposure of custom transition messages. In-tool emulation of interaction between morpheme:runtime and game AI system

NATURALMOTION

GOT NEWS? SEND US THE BIG SCOOP AT *EDITORS@GDMAG.COM*

ZEEBO THE DOWNLOAD-ONLY CONSOLE

IN MID-NOVEMBER, A NEW CONSOLE soft-launched in Brazil—a joint venture between Tec Toy and Qualcomm, through a company called Zeebo Inc. (based in San Diego, Calif.).

The console, also called Zeebo, is devised specifically for developing markets. Tec Toy has long been the official distributor of Sega games and consoles in Brazil, but is also known for manufacturing flash RAM-based Sega Master Systems as recently as this year. Qualcomm for its part, which owns 43 percent of Zeebo Inc., has wireless networks and mobile graphics chipsets as its core business.

All Zeebo games are stored on an internal 1 GB flash drive, and downloaded over a 3G wireless network. Apparently, once the drive is full, users will have to delete games to free up space. It's unclear as of press time whether those games will have to be purchased again. The console does support an SD slot, but this is used for pictures and music, not for game storage.

In terms of specs, Zeebo sports Qualcomm's Adreno 130 graphics core, with a 528MHz ARM 11 processor, and a resolution of 640x480. Early support for the platform includes publishers such as Capcom, EA, id Software, Sega, and Namco. Legacy games like NEED FOR SPEED CARBON and QUAKE are already announced for the console.

The interesting thing about Zeebo is that it's designed to thrive in an environment that is dominated by piracy. A download-only console eliminates the possibility of piracy without hacking the network itself (provided hackers don't get too resourceful with that SD card slot). Further, allowing download and UI updates via a wireless mobile network means that broadband penetration isn't an issue. It's quite a clever model, and the cost of the console itself is somewhat reasonable, at $599 Brazilian reais (BRL), with an eventual U.S. cost of $250. The wireless network is free as of press time, airtime fees and other costs are still being assessed. Games cost between 7–30 BRL (U.S. $3–13), but in the U.S. will cost between $5 and $15. Payment methods include not only traditional credit card payments, but also the cash card model, both using what Zeebo calls Z-credits.

The console will hit the U.S. in due course, but indeed Brazil and other developing countries may continue to be its core market. The distribution model is intriguing though, and it has the potential to fit its audience. If the software reaches a decent level of quality, one wonders if it might not work in other piracy-prone markets such as South Korea or China, which consoles have had a tough time penetrating. Time, and other regional launches, will tell.

—Brandon Sheffield

UZEBOX'S DO IT YOURSELF

WHILE SONY, MICROSOFT, AND NINTENDO CONTINUE TO battle it out in a three-way tie for last, a new piece of hardware called the Uzebox is positioned to rocket its way to the top. Now developers with a little soldering gun experience and a DIY spirit can build their own consoles.

Designed by Alec Bourque, the Uzebox is a home-built console that offers generous capabilities with a clean, easy-to-assemble circuit board. The heart of the machine is a single 8-bit RISC microcontroller called the ATmega644, which is part of the AVR family of chips manufactured by Atmel, and is coupled with a DT25 RGB-to-NTSC converter chip manufactured by Analog Devices. It has 4K of RAM and 64K of program memory, runs at 28.61818Mhz (Overclocked), and can display 256 simultaneous colors. It's capable of 4-channel sound and even has the ability to accept MIDI input.

All of the parts can be gathered at a low cost and its C coding tools are free under a GNU General Public License. Check out http://belogic.com/uzebox/index.htm for tools and schematics, as well as videos of a freeware TETRIS-like game called MEGATRIS running on the machine. Also visit www.atmel.com to download the AVR Studio 4 IDE and www.avrfreaks.net to join the AVR programming community.

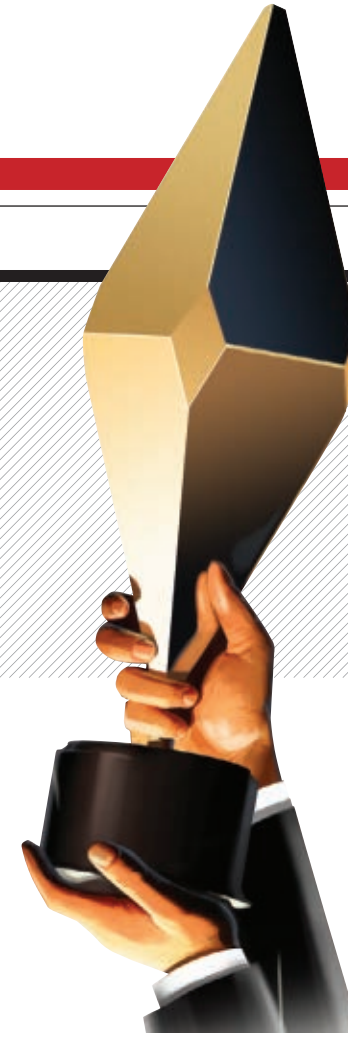
—Jeffrey Fleming

CALENDAR

6th Annual Mobile Games Forum 2009
London, UK
January 21–22
Price: $2,870
www.mobilegamesforum.co.uk/index.asp

Game Design Expo 2009
Vancouver, Canada
February 7–8
Price: Can $75
www.gamedesignexpo.com

2008 FRONT LINE AWARDS NOMINEES



OUR ANNUAL *GAME DEVELOPER* FRONT LINE AWARDS ARE ALMOST UPON US SO HERE IS A QUICK run down of the list of nominees in each category; Art, Audio, Books, Engines, Middleware, and Programming/Production. In determining the winners of the 2008 awards we're doing things a bit different than in years prior. After a period of open nominations in October we collected the results and along side developer suggestions we came up with a list that we narrowed down to five entries in each category. The results were a mix of focused solutions along with more general applications. We then handed the nominees to 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. As this issue goes to press, we're compiling the results—and in our January 2009 issue we'll reveal the winners. We'll also be inducting one special game development tool into the Hall of Fame that has been of enduring importance to the development community. Because the editors of *Game Developer* decide the Hall of Fame winner, it is not eligible in the regular categories. Stay tuned next month for the results.

ART

Photoshop CS3
Adobe

Softimage XSI 7
Softimage

Autodesk 3ds Max 2009
Autodesk

Autodesk Maya 2008
Autodesk

Modo 302
Luxology

AUDIO

Vivox Precision Studio v. 2
Vivox

Wwise 2008.3
Audiokinetic

Fmod Designer 4.17
Firelight Technologies Pty, Ltd.

Miles Sound System 7.2c
Rad Game Tools

Voice-O-Matic 2.6
Di-O-Matic

BOOKS

Dungeons and Desktops: The History of Computer Role-playing Games
by Matt Barton
AK Peters

Game Production Handbook, 2nd Edition
by Heather Maxwell Chandler
Infinity Science Press

Real-Time Rendering, Third Edition
by Tomas Akenine-Moller, Eric Haines, Naty Hoffman
AK Peters

Game Programming Gems 7
by Scott Jacobs
Charles River Media

The Art of Game Design: A Book of Lenses
by Jesse Schell
Morgan Kaufmann

ENGINES

Unity 2.1
Unity Technologies

CryEngine 2
Crytek

Gamebryo 2.5
Emergent Game Technologies

Torque Game Engine 1.5.2
GarageGames

Source Protocol 14, Build 3531
Valve

MIDDLEWARE

Autodesk Kynapse 5
Autodesk

PathEngine 5.16
PathEngine

Havok Physics
Havok

Euphoria
NaturalMotion

GameSpy SDK
GameSpy

PROGRAMMING/ PRODUCTION

TestTrack Pro 2008.2
Seapine Software

Perforce 2008.1
Perforce Software

XNA Game Studio 2.0
Microsoft

Subversion 1.5
CollabNet

Visual Studio 2008 SP1
Microsoft

WHAT WENT WRONG?

LEARNING FROM PAST POSTMORTEMS

POSTMORTEMS HAVE BEEN A STAPLE OF *GAME DEVELOPER* MAGAZINE FOR WELL OVER A DECADE. With so many authors revealing what went right, and of course the car-wreck style appeal of what went wrong with the development process, patterns start to emerge. It becomes clear that a lot of the same mistakes are being made over and over again, and that some companies are even repeating their own mistakes.

With that in mind, we decided to round up every "what went wrong" entry from the last three years, and compiled the most frequently made mistakes (usually over five times each) into this cautionary feature. As the saying goes, those who do not learn from the past are doomed to repeat it. So here you have the top 10 "wrongs," in no particular order, over the lifecycle of the current generation so far. No one is safe!

BRANDON SHEFFIELD *is editor-in-chief of game developer magazine, and has worked on a few games himself. He still hopes to strike it rich through writing. What went wrong? Email him at* ***bsheffield@gdmag.com****.*

WHAT WENT WRONG?

CONTENT ADDED TOO LATE. Getting story and features right is difficult at the best of times, but when that content comes in just under the wire, not only does that content suffer, every element of the game that relies on that content suffers.

TITAN QUEST (Iron Lore, Jeff Goodsill) "We struggled getting 25 detailed design documents to the technical department on time. We were still working on first cut design documentation more than one year after the start of the project, so the technical team had to fly blind in the beginning and some systems had to be redone.

"There was a fairly constant issue of design documents not being detailed enough for technical implementation."

After this trouble, the company devised an approval process, which would have the producer, lead designer, technical director, and the publisher's creative manager sign off on each document before finalizing it. While this helped, it did slow down the company's process a bit, and it's worth noting that for all Iron Lore's good points, the studio is now closed. It stands to reason that a "what went wrong" from a shuttered studio would warrant extra scrutiny.

And to quote Riley Cooper, who had a similar problem with TOMB RAIDER: LEGEND, "This clearly serves to remind us that if you don't want any features in your game to under-perform, you need to do them 100 percent or not at all."

BIOSHOCK (2K Boston, Alyssa Finley) "We had many drafts of the story over the course of development, but the final draft turned out to be an almost complete rewrite.

"Competing demands for time and resources meant that, unfortunately, some of the important narrative details of the game weren't created until the final rewrite, and therefore required quite a bit of work to retrofit into an existing game."

In an RPG-like game, story is the guiding star of the project. While ideally the systems would be fully integrated into the story to the degree that each can affect the other, at the very least the scenario needs to be nailed down before full-on production. In a game like BIOSHOCK, which features loads of spoken dialog, implementation and pipeline problems can crop up at the last moment, if you don't deal with them early on. This is exactly what happened to the 2K Boston team.

COMMUNICATION. When deadlines get fierce, and everyone's chugging away, communication is most likely the second thing to go. The first, of course, is quality of life, leading to crunch, but you can bet that we'll get to that.

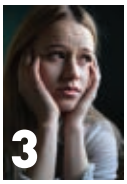
AGE OF BOOTY: (Certain Affinity, Max Hoberman) "Over the course of the project there were numerous disconnects between the perceived state of the game and the actual state of the game.

"The hardest hit were the designers, who continued fine-tuning plans for sophisticated features like matchmaking and party support long after the programmers had already made huge simplifications (and often cuts) to these systems. A combination of lack of attention to the project, poor communication, and wishful thinking led to the design team believing that several features were far more advanced than we were actually able to implement, and they did not find out the reality until very late in the project."

This depressing quote is made moreso by the fact that this was an Xbox Live Arcade title, and thus had a relatively small team. The company was working on multiple projects at once, which widened the communication gap. But that's another element we'll get to in due time.

FINAL FANTASY XII (Square Enix, Taku Murata) "During the development of FINAL FANTASY XII, the pressure to succeed was at such a high point that we were on the brink of losing control during even the slightest misunderstanding. What happened was our team was given the freedom to make changes at various stages of development, but the adverse affect of this freedom was miscommunication, confusion, and disorder. How work was to be distributed was also often ambiguous, which contributed to the problem."

Management overhead is a particularly large problem in Japan, but that's not to say it can't happen at home, too. Often there are too many managers, but not enough actual management going on. Sound familiar?



SCOPE AND SCALE. Ambition is a double-edged sword. On the one hand, as necessity's sister, it is the aunt of invention. On the other, biting off more than you can chew means your project will bite back. Schedules aren't always determined by developers, but they are agreed upon by them. Keeping the schedule and the scope of your game within reasonable limits while still doing the best you can is not easy. But it's absolutely critical.

STRANGLEHOLD (Midway Chicago, Brian Eddy) "Believe it or not, STRANGLEHOLD began its life as a free-roaming GTA-style game. But after about six months of development we convinced management that this would take far too long to develop, and was too risky to be Midway's first next-gen game on top of everything else we were tackling. We were then allowed to cut the game back to a more linear sandbox experience. But it still had an extensive feature list: the single player game, world interactions, four special moves, massive destruction, drivable vehicles with combat, and extensive multiplayer. We aggressively cut down the scope of each of these features as we progressed, but we kept them all in the game. After about a year of development it was clear we could not complete all these features at AAA quality, but we still kept trying. At the two year mark management set the hard end date. This let us cut drivable vehicles so we could concentrate on getting the rest of the features done at a high quality level. Ultimately cutting vehicles helped get the game shipped, but if we had made that decision a year earlier we could have put more time into mission variety and overall polish."

There's not much you can do when you're fighting against a parent that's beholden to stockholders. This just goes to show the importance of upper management pushing for realistic goals and cuts when necessary—or in fact making sure the scope is manageable in the first place.

GUITAR HERO (Harmonix, Greg LoPiccolo and Daniel Sussman) "We were really excited about including a freestyle mode so players could assemble their own crazy solos with divebombs, feedback, finger-tapping, and all the other adolescent guitar showboating moves that we so dearly love. We poured a lot of precious development time and resources into this feature, and sadly, had to cut it. It was very ambitious and we simply didn't have the

3vis

www.3vis.com

Film / TV

Games

Architecture

Autodesk
Premier Solutions Provider
Media & Entertainment

CREATIVE INDUSTRIES, CREATIVE TOOLS.

WHAT WENT WRONG?

time we needed to both make it sound good and integrate it into gameplay. Some of us feel that it was a gamble worth taking. Others aren't so sure."

Proper managing of time and scale can prevent some, but not all cases like this. It's one thing to cut a feature when it's not working, but another when time is all it will take to improve it. Cutting partially-developed features primarily because of time doesn't make anyone happy.

HIRING. Humans are the most expensive aspect of game development, and also the most difficult to find. It was incredibly common for past postmortems to mention not only the reluctance to hire, but also the difficulty of integrating new hires into the existing culture and methodology.

ROCK BAND (Harmonix, Rob Kay) "The only way to complete the title on time was to grow the team. We put most of our staff onto the game, and expanded the organization to new levels. Our offices were bursting at the seams. With staff spread across three floors, and no room left for the new hires, we needed to finish the game, we bit the bullet and moved the entire company to a larger space mid way through Alpha.

Despite all of this, we still didn't hire aggressively enough. Many years making small, tightly focused games had ingrained an efficiency bias and 'smaller is better' mentality that was hard to shake. We were afraid of the additional management required to hire more people, and it resulted in a longer harder crunch for all of us."

The ROCK BAND team had further trouble later, and "fooled ourselves into thinking it was too late to integrate any new hires." But ultimately, they had to do so during Beta—not the easiest time to be training new employees.

CRACKDOWN (Realtime Worlds, Phil Wilson) "Another shocking reality was that the number of coders yet to be hired was expressed in the order of tens. In a pragmatic development environment there would have been three choices: 1) increase the budget, 2) lower our ambitions, or 3) pull the plug. Unfortunately, the stakeholders were far from pragmatic, and rather than moving to jettison anything that wasn't in direct support of the clearly defined 'project pillars,' they used the scope discussions as a forum to add yet more ideas."

Keeping the team to measured goals is difficult when re-evaluating a project. Hiring more people and then increasing the scope kind of defeats the purpose! But as a counterpoint, TOMB RAIDER LEGEND had the problem of too many cooks initially. With too large of a team in the pre-production phase, attempting to get 45 people to agree on the direction for the title was a challenge.

JUGGLING PROJECTS, LACK OF LEADS. These two problem areas have been lumped together because they yield very similar outcomes—people are stretched too thin, and hold too many responsibilities. This means people often can't see past their bottom line to check the larger trajectory of the game's progress or feel.

AGE OF BOOTY (Certain Affinity, Max Hoberman) "Despite our strong start, we made some serious mistakes. Much of this was due to the fact that we split our focus across too many projects and shifted staff onto and off of the game on numerous occasions.

"On the programming side, for instance, we pulled the original two programmers off of the project while we negotiated the deal, which took several months to finally get signed. We eventually got these two back on it by hiring two replacements to handle LEFT 4 DEAD, our other big project at the time. We hired a third programmer to assist these two while also contracting out some programming work. We then hired another full-timer and ended our work with the contract programmer. We handed the work the contractor was doing over to one of our LEFT 4 DEAD programmers, who volunteered to complete it in his spare time."

And it goes on from there. Certain Affinity then took on a third project, pulling the volunteer back off, and eventually putting the original two coders back on. This coder confusion, even on a smaller project, can make a game's development a bit schizophrenic. While Certain Affinity felt as a young company it couldn't say no to projects, AGE OF BOOTY would've been better off without distractions.

CATAN (Big Huge Games, Brian Reynolds) "Our biggest mistake ... was failing to assign a full-time lead programmer or lead artist to the project.

"Not having a solid management structure meant that things tended to fall through the cracks. There was no one to set goals for the programming team or art group. There was no one to assert what needed to be done day-to-day, or week-to-week, or month-to-month. The employees sometimes drifted, unsure what they should work on next, spending too much time on assets that were unimportant, neglecting elements of the game that were actually critical."

As much as developers rail against management on occasion, leads are important for setting structure and scope. Left unchecked, an artist (just as an example!) will fiddle with an asset until doomsday.

LACK OF TECHNICAL DOCUMENTATION. This problem was more common than I imagined it would be. But it does stand to reason that determining the scale of your tools, with appropriate code review, is incredibly important to building out your game in the early stages.

ROCK BAND (Harmonix, Rob Kay) "Expanding our code department for ROCK BAND meant bringing in new programmers who were very talented but weren't all designer-programmer hybrids, or if they were didn't know they were allowed to be. Too often we jumped headlong into implementation of a new system without taking the time to properly examine the implications or test the edge cases of the design. This bit us in a few areas, notably online matchmaking, which had to be redesigned multiple times.

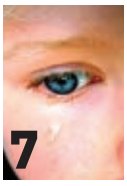
"No doubt about it, jumping into development of complex new systems without a technical plan upfront was a flat-out mistake."

Harmonix's solution going forward is to include code review and a technical design document before implementing new systems. It seems obvious in some ways, but the frequency of this problem in postmortems indicates it's not yet at the top of everyone's list.

INDIGO PROPHECY (Quantic Dream, David Cage) "Another classic error we committed was trying to develop generic tools with a view to possible future productions, rather than tools dedicated to the experience of the game we wanted to create. The initial scripting tool was supposed to enable us to script both an FPS and a tennis game. The reality quickly proved to be different from the theory.

"A generic tool enables management of a great variety of cases, but none of them very effectively. The prospect of reusing a tool as-is for future productions is usually a pipe dream that costs time and money in the short term, with no guarantee of profitability in the long term."

Quantic Dream managed to realize this mid-project, and adapted the tool—but not without some lost work. Naturally, what would help here is some proper technical documentation of the project's specific needs.



OUTSOURCING. Outsourcing is becoming increasingly common in game development, with art being the most common target. It can save money, but it's tough to get right, and requires a lot of senior bandwidth.

SAM & MAX (Telltale Games, Telltale Team) "Once we settle on which contractors we're going to use, even more time goes into explaining exactly what we need. Since the SAM & MAX schedule is very tight, we don't have a lot of time to account for corrections or mistakes due to inadequate contractor talent or miscommunication. For example, we felt a lot of pain when a contract studio sold us an A-team, then once the contract was signed, gave us a B-team whose work fell short of the standards we agreed to.

"Communication became even more difficult when the contractors weren't local (we had some on the other side of the planet). When the contractor's workday takes place during the middle of the night for us, what would normally be a 10-minute conversation turns into a two-day exchange."

Due diligence is incredibly important in the case of contractors, but it's tough to foresee the A/B-team problem, which incidentally was also mentioned by the STUBBS THE ZOMBIE team. Speaking of which ...

STUBBS THE ZOMBIE (Wideload Games, Alexander Seropian) "We underestimated just how much time was required to manage contractor submissions. We knew it would be time consuming, but even with that expectation, the combination of art directing and art production was more work than we had time to do. We were short on producers and our artists were scheduled to produce content on their own. We didn't have enough bandwidth available for reviewing submissions in a timely manner. We realized too late that our production phase requires an intense focus on the work coming in from the contractors. Focusing our internal efforts on the contractor feedback loop should have been a higher priority for our art direction team."

Now we get to the senior management overhead. Without a team dedicated to managing contractors, a lot of work is going to go unused. Wideload now has a section of the team devoted to this, and subsequent production has reportedly been smoother.

CONTINUED ON PG 12

WHAT WENT WRONG?

CONTINUED FROM PG 11

POLISH The devil is in the details, but when we talk about "immersion," that's often the result of that particular Beelzebub. Running out of time in the polish phase often means a game that's rough around the edges, or which doesn't fall in line with player expectation.

TITAN QUEST (Iron Lore, Jeff Goodsill) "We always told ourselves that six months would be the right amount of time to balance and polish the game. With more than 30 hours of gameplay at each of three difficulty levels and entirely new technology, we needed all of that scheduled time and then some. Just balancing the skill system, with all the combinations of masteries and skills, not to mention the thousands of pieces of equipment with hundreds of thousands of variants, was going to be an epic task.

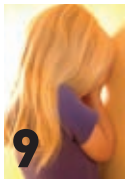
"In the end, we only had three months. Luckily, we were able to whip the game into shape, but we also had pages of polish feature suggestions that we simply ran out of time to implement."

The team had to compromise, focusing on the first half of the game, and letting polish of the higher difficult levels go until near the project's end. Not necessarily the best option for player retention.

GUN (Neversoft, Scott Pease, Chad Findley) "What's worse than a game that's too short? One that's too short and too easy. Due to a tight schedule and inexperience with the genre, we took a very simplistic approach to game difficulty, putting the standard 'Easy, Normal, Hard, and Insane' selection at the front of the game. Then we sat back and depended on the players to select the appropriate challenge level for their particular tastes. What the hell were we smoking?

"In the video game market, asking players to set their skill level before they've even played your game is a freaking naive way to go about it. ... our gut reaction to the reviews was that too many people played at the wrong difficulty level—and it was our fault."

Though the Neversoft team focus tested the difficulty levels, choosing difficulty before playing is a daunting task for many players. This is another problem that more time can fix! But how to get that time is a much deeper problem.



POOR TOOL IMPLEMENTATION. Building tools for your game is horrifyingly difficult, and the problem becomes larger when the use of these tools is not obvious, or when they become too big for their britches. This is the cousin of the lack of technical documentation problem, and no less common.

RESISTANCE: THE FALL OF MAN (Insomniac, Marcus Smith) "The flipside to homegrown tools and technology is that our tools changed quickly and our ability to properly train people on all the changes proved impossible. Building assets while simultaneously building the tools needed to create them is akin to trying to build a house on quicksand. Artists would literally open their tools one day and discover new interface buttons and have no idea what they were or how to use them. Many assets needed to be rebuilt, re-lighted and/or re-animated because of changes to our builder tools.

"Adding to the confusion, only a small number of programmers had the knowledge required to debug the problems, and these people were overwhelmed with requests for help. If it weren't for their inhuman effort and long hours hunched over their keyboards, we would have never hit the launch date."

Insomniac had a related problem, wherein the company couldn't maintain stable builds without a great deal of effort. Making your tools accessible is absolutely essential.

UNCHARTED: DRAKE'S FORTUNE (Naughty Dog, Richard Lemarchand) "We realized that we'd bitten off too much with our tools approach—we'd tried to be too clever, coming up with convoluted approaches that were intended to solve every last problem that anyone had ever had with each kind of tool. To make it worse, we'd gotten distracted by our lofty aspirations, and had left it very late to implement a build pipeline that let people actually run and play levels.

"The moral that we took away was that even though it's good to aim high with your tools, you should choose your battles, and shouldn't try to solve every last tools issue that your team faces. In some cases, simple work-arounds are better and free up more time to work on the game itself."

The goals were noble, as these problems were originated in a desire to share technology. The lesson remains the same—reign in scope until the project's goals are completely clear.

CRUNCH. Well here it is, the inevitable crunch discussion. Everyone knows it's a problem, but it keeps on keeping on. Crunch is usually a side effect of most of the other problems on this list, and then exacerbates those problems in turn. Some studios go well out of their way to avoid it—others can't seem to help themselves, for reasons of ambition, money, or any number of things. It's never good.

DRAWN TO LIFE (5th Cell, Joseph Tringali) "Consistent overtime is not fun, nor is it something that should be assumed when developing a game. We ended up working late weekdays and Saturdays for the entire second half of the project.

"Our studio had the perfect storm of factors that contribute to crunch—lack of experience on the platform, an ambitious game, and a schedule that was too short. Combined with so little pre-production, we were playing catch up from day one, taking far too many shortcuts to implement the required game features."

Playing catch up from day one is the key phrase here, and proper schedule and project management is the solution. Easier said than done!

STUBBS THE ZOMBIE (Wideload Games, Alexander Seropian) "(In spite of using contractors), we crunched for a solid three months, which isn't too bad relative to past experiences, but is way worse than zero. Because we let some major components slip into post-production and were four months behind schedule, and we let ourselves get behind on contractor approvals, we ended up with more than post-production tasks during our post-production phase."

This example should be a reminder to us all that contractors don't necessarily eliminate crunch. A healthy post-production timeframe is very useful, but shouldn't be used as a cushion when other phases fall behind.

WHAT WENT WRONG?

In the end, a lot of the same problems will keep cropping up in development. People fall into their old working habits and let best practices slide. The jury remains out on methodologies like Scrum, but certainly a very conscious management of projects can help identify problems before they become critical. Let's keep sharing those best practices, and dare I suggest they be shared in *Game Developer* magazine? Do drop us a line!

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 on Nov. 7, 2008.

Upcoming Epic Attended Events:

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.

THROUGH THE LOOKING GLASS: UNREAL ENGINE 3 POWERS *MIRROR'S EDGE*

This story was written for www.unrealtechnology.com by freelance reporter John Gaudiosi.

Electronic Arts-owned Digital Illusions (DICE) is redefining the first-person shooter with *Mirror's Edge*, the sci-fi action game that puts players in control of a runner named Faith who must deliver messages to and from people who live on the edge of a society that has become controlled by a strict government regime.

Every move in the game, whether it's running, jumping, climbing or shooting, comes to life through a first-person perspective courtesy of Unreal Engine 3.

"Epic had been developing *Unreal Tournament 3*, so I think we came along at the right time," said Owen O'Brien, senior producer. "A lot of the PS3 support came in when we needed it. From that respect, Unreal Engine 3 gave us a good cross-platform base."

According to O'Brien, DICE chose Unreal Engine 3 two years ago because the team was going to be innovating more on gameplay than on technology, so they needed a stable engine that could handle fast iterations. DICE blended Unreal Engine 3's tools with its own animation and lighting system, which was created in tandem with Sweden's Illuminate Labs.

"Illuminate Labs adapted their Turtle lighting system for Unreal and we call that 'the beast,'" explained O'Brien. "The engine allows us to do soft shadows and light bouncing, and it gives *Mirror's Edge* a really unique look, especially since the game is based so much on white and the primary colors. The shadows and lighting are very important to the game."

The unique look of *Mirror's Edge*, which has received critical buzz from gaming media, is the result of years of experimentation. By choosing Unreal Engine 3, DICE was able to hire level designers from the Unreal development community. Thanks to the ease of use, the engine opened up new freedoms for level designers, who were able to achieve things once relegated to programmers.

"We'd been prototyping for a long time trying out things with the field of view and the movement and the animation rig, and we just needed something stable that we could keep hammering on," said O'Brien. "Using UE3 made it easier for us to hire people because a lot of people know Unreal and have worked with the engine. Recruiting talent quickly was attractive to us."

According to Tom Farrer, producer of *Mirror's Edge*, the impetus behind the game was to create a first-person game set in an urban environment that could replicate third-person perspective action like that seen in Ubisoft's *Prince of Persia* games.

"We focused on the through-the-character experience, rather than through-the-gun," said Farrer. "We wanted to give a sense of physicality and let people know they can use their legs and arms to reach up and grab something or climb up something."

Mirror's Edge by EA DICE

The game employs a momentum-based system for the more action-oriented chase sequences, which allows players to better navigate the city environment at a quick pace. The skill element of the game comes from the timing of Faith's many moves.

"*Mirror's Edge* is an action-adventure game, but it's from a perspective that you haven't seen before," added O'Brien. "I think people think of Unreal as a good first-person shooter engine because up until now all first-person games have been shooters. We're going to change that perception."

With multiple teams across EA developing games on Unreal Engine 3, O'Brien said "the beast" is already being employed by several EA LA teams.

"Unreal Engine 3 is used quite a lot at EA, and we always share knowledge across teams," said O'Brien. "We take things from other studios, whether it's optimizations for PS3 or other technology. That's one of the strengths and benefits of being at a company like EA."

POWERED BY UNREAL TECHNOLOGY

EPIC GAMES

For UE3 licensing inquiries email:
licensing@epicgames.com

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

WWW.EPICGAMES.COM

INTERVIEW: YUJI NAKA

YUJI NAKA IS BEST KNOWN FOR HIS ROLE IN THE CREATION OF SONIC THE HEDGEHOG. His programming skill allowed for Sonic's iconic speed, as well as the multitude of ring sprites he got the Genesis console to push in its early days, alongside the impressive-for-the-time parallax scrolling. More recently he left Sega to form his own studio called Prope, and is now developing two games, LET'S TAP and LET'S CATCH, for the Wii. Sega remains publisher and partial owner of his company. LET'S TAP is unique in that it's played without touching the controller—the Wii remote sits on top of the provided box, and you actually tap the box to make characters move, jump, and run, or complete other actions.

Lately, I feel that Naka has been somewhat marginalized by press and developers alike. He is often described as unfriendly and reclusive, and overly controlling of projects with which he's associated.

I would like to propose a counterpoint here. The times I've met him, he was walking around the show floor of the Tokyo Game Show, seeing how people liked his games, or at E3 social events. When spoken to directly, without an intermediary, he's approachable, candid, and well spoken.

There is an unconfirmed anecdote that circulated around the Western press which states that when he left to build Prope, he offered any Sonic Team member the opportunity to go with him, and almost no-one did. This story was meant to indicate his lack of relevance in the current industry. I could see this anecdote being true—but I would approach it from another perspective. He left to do his own thing, in a regional industry that is very reluctant to change, or to challenge the status-quo. Of course nobody moved. Further, Naka's reasons for trying to control projects related to SONIC become clear once you read to the second page of this interview.

I find Yuji Naka to be thoughtful and driven, not arrogant, and he left Sega because he still wanted to make games, not just manage them. Though the article is brief, it's my hope that it will increase understanding both of Naka, and the constraints of the Japanese industry.

BRANDON SHEFFIELD *is secretly hunting down every member of Sonic Team. He definitely does not go to sleep every night holding an Amy doll. Email him at* **bsheffield@gdmag.com**.

BRANDON SHEFFIELD: ***Where did the idea for LET'S TAP come from?***

YUJI NAKA: It was something I came up with while we were working on another action game. I had noticed around that time that the Wii controller was a remarkably sensitive device, capable of detecting even very small, faint vibrations. We did a bit of a test where we placed the controller on a desk and started tapping on the desktop around it—not only did the Wii remote detect that, but it also detected when we tapped on a desk placed adjacent to the one it was lying on. Looking at that, I thought that it was pretty neat, that maybe we could do something with this. From there, I thought about how up to now, rhythm games have been largely digital in nature—made up of 0s and 1s representing "off" and "on"—but this controller could measure more gradual levels of input in between those two extremes. So it was a process of discovery that ultimately led to the idea, the idea to take a digital game and make it analog and able to accept a wider range of input.

BS: ***Was it difficult to find the "fun" of such a simple interaction as this?***

YN: Well, think about it—sometimes, do you find yourself just idly tapping on something during the day? I know I do it pretty much all the time, and I think everybody else does too. So I thought about making that into a game somehow. That's what makes this game fun, I think—the fact that it's something everybody does now and again. Now, of course, there are other music and rhythm games, titles where you're matching some kind of rhythm onscreen. But even in the case of GUITAR HERO, it's still a matter of "on" or "off," 0 or 1. There isn't any measure

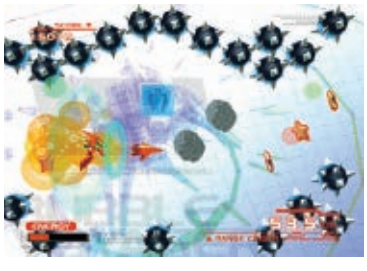
of strength. Meanwhile, with this there's a whole spectrum of strength to the tapping. It's something that's really pretty innovative even within the field of rhythm games.

BS: ***With a lot of Wii games, my wrists end up hurting from all the flailing of the controller I'm doing.***
YN: Yeah, they sure do.

BS: ***Is this game different? Any concerns about finger fatigue?***
YN: Well, your fingers are going to be tired here either way. It's a music game, after all! It's all a matter of moderation. I'm used to it so my fingers don't bother me at all, but when you begin, you might deal with that a bit. I think it's a pretty easy process.

BS: ***What made you decide to quit Sega, or at least no longer work under the name Sega?***
YN: Well, I could have stayed under Sega itself, but I already had a very high position there. The game industry has a very short history behind it, and as a result, the more games you make, the further you work your way up the company ladder, until you become one of the heads of the whole outfit. Once that happens, you start running out of time to actually make games. It's better to keep yourself directly involved with the game—actual process, you know? Directors are pretty high up the job ladder in the movie industry, but they're still involved with every aspect of the film they're working on; they're still making movies their entire careers. The game industry isn't quite like that, and I think that's a lost opportunity for a lot of people.

Before I left Sega, I was high enough up that I was looking at every game the company was developing. Once I was in that position, though, I found that I wanted to get into the nitty-gritty details instead with the games, including SONIC—the whole "it'd be better if this bit were like this instead of that" type of thing. There was a lot I wanted to do that I couldn't gauge until someone actually tried making it. So, at the age of 40, I convinced Sega to let me build a company—since it's Sega that's behind the company, that's the company publishing the games. Really, if you're a game creator, no matter how high a position you have in the industry, you need to keep creating. It's better for the industry, and it's more fun for everybody involved.



BS: ***How many people are in Prope right now?***
YN: Right now it's about 40 staff members.

BS: ***Will the "LET'S" lineup become a series of games?***
YN: If it sells well enough, sure. Of course, you can't really say how well it'll sell at this point, but if it does great—if we can get a lot of people to play it and enjoy it—I'd love to make another one.

Shown here is the SilentBlocks minigame from LET'S TAP.

BS: ***Is Prope focused on these types of simpler games?***
YN: We're actually planning on making a game like SONIC right now. We want to keep trying to make various kinds of games.

BS: ***Character games and so on?***
YN: Yes, that's what we're making now.

BS: ***I noticed the penguin on the promotional page [one of the slogans for this game is "The world's first game that even a penguin can play!"]—you see them on Suica [mass transit] cards too. What is with Japanese people and their fascination with penguins?***
YN: You're right! Well, I like them! I always have. And, you know, it's true that even a penguin can play this. The (TGS booth) over there, showing the visualizer—all you do in that game is tap away, and there really isn't anything more to it. Anyone from a one-year-old to an 80-year-old man can enjoy that mode; it's the sort of thing you can see for yourself when you try it out. In fact, the controller's so good at detecting the tapping that you can play it with your feet, if you actually wanted to try that.

BS: ***I'd like to see some penguin playtesting.***
YN: I sure would too!

BS: ***For that matter, this is a game that people who are missing limbs could potentially play. Did you think about that as well?***
YN: I can't say I was thinking about that in particular, but it always makes me happy to see a large variety of people enjoying our work.

BS: ***There's a site called ablegamers, which is written for disabled video game fans. This is certainly a project they would find interesting.***
YN: I can definitely see that, because they wouldn't have any problem playing this, certainly. You can play the game with a single finger, even.

BS: ***Have you ever considered giving a talk about your ideas at GDC?***
YN: I'm bad at speeches, but I'd certainly like to go again as an attendee. If I do, I definitely want to get the First Penguin Award. That's part of the reason why I'm making new games like this one. It's really a shame that they changed the name of it to the Pioneer Award. If I had the choice, I'd much rather have the "First Penguin" one someday! Even if it takes me another ten or twenty years!

TWISTED PIXEL'S THE MAW

MOST GAME DEVELOPERS HAVE SOME DESIRE TO make an original game based on their own ideas for characters, story, gameplay, technology, and whatever else they think would make a good game. It is this desire that drove Michael Wilford, Josh Bear, and I to create Twisted Pixel Games. We felt that having our own company was the best way for us to realize our ideas and get them out there for the world to see.

We believe that digital distribution on game consoles has really opened up a lot of doors for new small developers like us.

The downloadable console game market is flooded with lots of very casual games that are relatively small scale—but we want to bring a more retail-like experience to the downloadable console game market.

/// WHAT WENT RIGHT

1 COMMUNICATION. The majority of the team members have worked together in some fashion before joining Twisted Pixel, so everyone was already familiar with how to work with one another. This meant that communication was never an issue for us.

Because the whole team worked together so well, there was rarely, if ever, a time when someone didn't know something they needed to in order to accomplish their tasks. We never had any need to have formal scrums or any other contrived methods to aid with communication.

Normally if a publisher is involved with a game, developers have milestones roughly every month during the game's development in order to show the game's progress to the publisher. When developing a game for XBLA, this is not the case. Having had the experience of required milestones before, we saw the value in it. From the very beginning of the project we set up internal monthly

milestones that we would adhere to in order to ensure that we would keep on track with the game's schedule. Although it was not required or even recommended, we sent these milestones to Microsoft every month so they could see the game's progress. This was an incredible help in terms of keeping the game on track, and helped build a better relationship with Microsoft. Because of our retail background, the team was used to working in this fashion and the milestones went very smoothly for us.

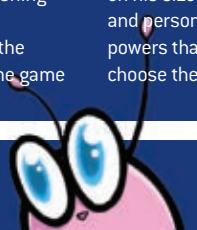
2 FOCUSING ON OUR GOALS. When we started work on THE MAW, we came up with a few vision points for the game. We made a conscious effort to always adhere to these vision points throughout development. The game's director, Josh Bear, ensured that everyone remained dedicated to accomplishing the vision we had for the game.

The first and most important vision point was to give the game personality. We wanted people to have fun with the game whether they were playing it themselves or watching others play it. We accomplished this through the game's animations, characters, cinemas, color scheme, and general art style. The biggest contributors to the game's personality are the animations. Our art director, Dave Leung, did a phenomenal job of giving loads of personality to a giant purple blob with one eye.

The second vision point was to give the game "pick up and have fun quickly" gameplay. We accomplished this by giving the game a small set of clear overall goals: Maw needs to eat things to solve puzzles and Frank needs to help Maw eat things.

The third vision point was to make eating and getting powers fun. We implemented several different ways for Maw to eat throughout the game and changed the way that Maw eats based on his size and on some of his powers in order to add variety and personality. We also spent lots of time designing the five powers that Maw would be able to acquire. Taking the time to choose the right vision points for our game up front and making

FRANK WILSON *is CTO and co-founder of Twisted Pixel Games and is the primary engine programmer on THE MAW. He's been working in games since 2001 and has loved every minute of it. He also has a beard on his face. Email him at* ***fwilson@gdmag.com****.*



an effort to always stick to these points has definitely made our game better. It's something that I think that all of our future games will benefit from.

3 GETTING A NICE OFFICE SPACE. While it may be a given to most that any company should have a nice office space, sometimes as a very small startup there are costs that just aren't worth it quite yet.

When we started the company, there were only 3 of us, and we started out using half of a warehouse as a temporary office space. This building had a concrete floor, cinder block walls, and a minimal amount of heating and air conditioning. The other half of the building was occupied by craftsmen who were working on the space so most days we worked to the tune of circular saws. There were many days where we had to pray that the saws didn't start up during some of our most important phone calls. We had to wade through piles of sawdust to get to the rest room which was inconveniently located in the other half of the warehouse.

We started looking around for office space, and it turned out that our lawyer had some space available in the basement of his building that was formerly occupied by a urologist's office. Due to the office's history we certainly had some modifications to make like removing sinks from every room, but it also had some things that worked out very well for us as we started hiring people over the next several months. The former patient waiting area was perfectly suited for a conference room and lounging area. After removing the glass window and built in desk, the receptionist area was perfect for fitting the gameplay team together in one room

We learned very early on that we shouldn't underestimate the positive effects of working in a nice environment and having multiple bathrooms.

4 DEVELOPING OUR OWN ENGINE WITH THE HELP OF MIDDLEWARE. Though our team could be considered large for a downloadable console title, it is actually quite small considering we made a 3D action/adventure/platformer from an engine that was developed concurrently with the game.

With a team size of 8 people as its high point, we had some serious decisions to make about what we would use for our technology. After taking a few months to evaluate the various commercial engines and middleware that were available, we determined that the best solution for us was to create our own engine but use various pieces of middleware where we thought it was appropriate.

We chose to use middleware for our animation exporting and playback, physics, scripting language, and visual effects. These were areas that we knew would take significant effort to develop but were separate enough that we could use middleware to do the job for us.

The rest of our engine was designed and implemented internally. One of our focuses was to ensure that all of the design tools were not cumbersome, but very easy to use while providing a large amount of flexibility to the design team. We accomplished this by putting a lot of time into our terrain and level editor early on in the development cycle. The other major contributor to this was our choice to use Lua as our scripting language. We've designed our engine so that our Lua scripts control everything from data loading to actor manipulation to the user interface. This means that we could make a completely different game without changing a single line of C++ code, but by only changing the Lua code. The flexibility that Lua gave our engine improved iteration time and allowed our lead designer, Sean Riley, to implement many of the game's features without help from the team's lead gameplay programmer, Mike Henry.

Our entire engine was built as a library so that we can easily link it in with other applications. This has proven to be invaluable with the development of our command line tools and game editor. It promotes code reuse, makes it incredibly easy to ensure that all our tools are using the latest engine code, and allowed the game editor to render the levels exactly as you would see in the actual game.

5 FITTING THE GAME INTO THE REQUIRED SPACE. One of our biggest concerns with developing a game of this magnitude for Xbox Live Arcade was being able to fit all our content into the required 150 MB. Coming from our background in retail development, we typically didn't need to worry about reducing the amount of disk space used by the game. It was more often the case that we would have so much disk space available that we would duplicate the same data multiple times on a disk in order to decrease load times.

The final game ended up having over 850 animations, 25 music tracks, over 1000 sound effects, over 200 visual effects, and over 150 character and object models as well as lots of other data that contributes to the disk space used. So, our efforts for reducing disk space usage were certainly needed and definitely paid off.

The biggest boon to accomplishing this was Granny 3D, our middleware choice for animation and mesh compression, and animation playback. Granny uses some very clever methods (that I won't dive into here) to reduce the amount of disk space and memory used by animations. The extensive set of export settings that Granny provides allowed us to set the compression level on a per animation basis or even on a per bone basis.

From the beginning of our engine, we built all of our data driven components so that they could load data from both an easy to read XML format and from a binary format that would allow superfast load times. With the help of the Granny

GAME DATA

DEVELOPER:
Twisted Pixel Games

PUBLISHER:
Twisted Pixel Games

NUMBER OF DEVELOPERS:
7 full time, 2 part time

LENGTH OF DEVELOPMENT:
9 months

RELEASE DATE:
January 2009

PLATFORMS:
Xbox Live Arcade, Windows

DEVELOPMENT SOFTWARE USED:
Visual Studio, Photoshop, 3ds Max

COMPRESSION RATIO OF MODELS, ANIMATIONS, AND OTHER CUSTOM DATA:
About 85 to 1

NUMBER OF DUNGEONS WORKED IN:
1

preprocessor toolkit, we were easily able to convert this data from XML to binary with minimal effort and get the benefit of using compressed Granny files.

/// WHAT WENT WRONG

1 GETTING A PROJECT GREENLIT. When Mike, Josh, and I started Twisted Pixel, we had no idea how long it would take to get a project approved for development. At the time, the Xbox 360 was the only current generation console on the market and thus the only console with a definite plan for how they would be digitally distributing games. Because Microsoft's plan seemed clear, we thought they would be a good choice for our first game. We also saw this as an opportunity to get in early with Sony and Nintendo on whatever their plans would be for the digital distribution of original titles.

We came up with several ideas for all of these platforms, created overview design documents, created some prototypes, and had Mike and Josh fly out and pitch some of the ideas directly to the proper people on each platform. Other ideas were merely discussed over the phone or emailed. We learned a lot from these experiences.

You have to be careful with a prototype. Sometimes people don't quite get past the fact that something that you show them is just a prototype and not a finished product. For example, one of the prototypes that we made was at about the quality level of a typical downloadable console game at the time. Because it was just a prototype, we planned to put a lot more polish and many more features into it if the title was approved. However, it was apparent that many of the people that we showed the demo to were stuck on the idea that what we showed them was basically the finished product.

Getting in too early before the console manufacturers had figured out for themselves what their plans were resulted in a lot of wasted time. Luckily, we were able to get meetings and face time with the right people early on, but because it was too early, it was hard to get approval to start on anything because there was no official process in place. This may have been different had we been a more established studio at the time. At one point, we actually did get approval for development of one of our ideas from one of the console manufacturers only to have their business model change, rendering the approval somewhat useless to us.

2 TOO LITTLE TIME. This is a strange one for me. It's kind of a "what went wrong" even though it was something we had prepared for from the very beginning. We knew that in order to make THE MAW be the game that we wanted it to be in the budget that we could afford, we would have to work some pretty crazy hours to make that happen. We were very aware of this from the very beginning.

We basically just had too much work to do in too little time. We were developing our own engine and game editor while also trying to develop the game, so there were many dependencies on the order in which specific tasks could get done. Several tasks that we would have liked to have completed earlier in development had to wait until the feature was implemented in the engine before they could be completed.

Luckily, we were able to get enough of the engine up and running early enough that designers could set up many aspects of levels without too many issues. We worked very hard to get the first level up, running, and to a somewhat polished state pretty quickly. This gave us a pretty good test of the features of the engine. For most features, we implemented support in the engine for the data files early on with support being added to the editor later so that designers could still use those features by hand-editing files without being blocked by editor development.

We were diligent throughout the project about cutting features that just weren't working or that wouldn't add much value to the game. But we were careful to not strip the game of things that make it fun and give it the personality that we wanted it to have. Despite all of this, by the end of the project, everyone was working some pretty crazy hours in order to polish up all of the features that we had.

For our next project, we have an existing engine and an editor that turned out very well so our designers can get started right away with setting up the game. We have lots of new features planned for the engine and editor but they shouldn't hold up progress for the designers. As a company, we need to make sure that this timeline is not the norm. Moving forward, it's looking like it won't be.

3 XML LOAD TIMES. While we are happy with the design of our engine, using XML files for nearly all of our data during development proved to really slow down our iteration time. Due to our data loading scheme for this game, we wasted a lot of time waiting for the application to startup.

From the very beginning, we designed our engine so that nearly all data could be read in from both an XML format and a binary format. The intention was that we would use the XML format during development. This allowed us to have human-readable files that we could glance at and easily modify, and which would give us the ability to see what had changed from previous versions of a file via our source control system's history. Whenever we made an official build of the game, the XML files were converted to their binary format which significantly decreased load times, the amount of RAM used by the data, and the disk space footprint.

This worked great throughout a large portion of the development, but as we got closer to the end of the project, we had so many large XML files that the time it took to parse and allocate the data from the files was nearly unbearable.

We still like the idea of using XML files for development for readability and historical comparisons; however we plan to make this process work a little better for future projects. During development, we plan to have the game check for a cached binary version of a file: if it exists and the corresponding xml version is not newer, then the cached binary version will be loaded. If the cached binary file does not exist or is older than the xml file, then the xml file will be converted to its binary version and cached before loading it into the game. This will make the initial load of the game take longer, but should significantly decrease the load times in subsequent loads.

For comparison, using the XML files as well as some other uncompiled data at the end of development, the game took roughly 2 minutes to load. However, in the shipped version using all binary files and other compiled data, the game takes roughly 15 seconds to load. By the time all of the logos at the game's launch have been displayed, all of the data for the game has been loaded and initialized. Obviously, this is a huge improvement and frankly I'm amazed that we're able to load and initialize all of the game data in such a short amount of time.

4 NOT BEING ABLE TO RUN THE GAME FROM WITHIN THE EDITOR. Our entire engine is built as a library that links in with our editor so that all features of the engine are available to the editor. So, one of the features that we planned was to be able to edit a level in the editor and then actually run the level in the editor so designers could quickly and easily iterate on changes. Regretfully, we did not implement running the game within the editor and instead pushed it off until after THE MAW was completed.

So, whenever a designer made changes to a level, the level would need to be saved and then the designer could manually launch the game to the specified level to test out his changes. At the beginning of development, this was not a problem because the data loading was quick so we thought it was safe to push this feature off until later. However, as the project got further into development, this became a problem because of the load times caused by our development XML files.

This caused tweaks that should have been very quick to take a lot longer than they should have. If we had taken the time to implement this feature early on, it would have had great effects on productivity and the load time issue wouldn't have been such a big deal.

5 LUA AND LUABIND'S MEMORY ALLOCATIONS. Despite the great benefits that using Lua and Luabind in our engine has given us, it took us a few tries to get them working smoothly in our engine. Using Lua and Luabind for all of our gameplay, UI, and generally everything specific to THE MAW has been a great help in development. It has helped shorten our iteration time and made it so that some of our designers are really doing the work of programmers. The flexibility and ease of use of Lua combined with the ease of binding C++ classes to Lua using Luabind really put the power of creating the game into the designer's hands. The problems came once we started paying attention to what was happening to memory.

At first, we went with a design that gave each actor its own virtual machine to run Lua. Our plan with this design was to keep each actor in its own isolated environment in order to enforce that all actors would communicate with each other only through our messaging system. This proved to be a mistake because there is a certain amount of overhead with having each VM plus there's the fact that many of the Lua scripts were duplicated across VMs which wasted a lot of memory. The other big issue was the performance hit caused by having to run the garbage collector on each VM.

We realized halfway through the project that these issues needed to be addressed so we reworked our scripting system so that all actors and entities in the game used the same VM. This was a great boon to performance and memory usage; however there was still more work to be done. Because we had so many objects in the same VM, it took even longer for the garbage collector to check all of the objects for collection and perform the necessary work to clean up dead objects, not to mention all of the fragmentation caused by this.

To address this issue, we did some logging to see what was causing the bulk of the memory allocations and with a fairly small effort we were able to rework our scripts a bit so that many fewer allocations were occurring each frame. We went from about 350–450 allocations made by Lua and Luabind per frame to roughly 0 to 15 depending on what was going on at any given time. Obviously, this is a great improvement, but it still caused some issues.

We still had to run the garbage collector when too much memory was in use by Lua even though this could easily cause a 40 second or more hang while the garbage collector did it's work. Obviously, this is unacceptable when a user is playing the game. We tried several things to help improve this, finally implementing a solution where we made modifications to Lua's garbage collector so that it continuously runs on another of the Xbox 360's hardware threads, taking into account thread safety and minimizing the number of required critical sections.

We still had an issue with fragmentation, but roping off an area of memory just for use by Lua safely kept the fragmentation from contaminating the rest of the system.

/// CHEWED UP, SPIT OUT

I'm very proud of what the whole MAW team has accomplished and am excited for us to push our next game to an even higher level of quality. For our freshman effort, I couldn't ask for better results. We have a game that's getting very positive press, that we're proud of, and technology that we can reuse on future projects.

BY LEE PURCELL

ALERT: LATEST CALL OF DUTY* RELEASE BREAKS NEW GROUND AND USES CORES WISELY

The target is in the sights: Take an enduring game franchise and create a new release that keeps gamers roundly entertained while exploiting the latest PC platform advances. To that end, a seasoned cross-platform development team at Treyarch, experienced in the genre of multiplayer shooters, approached *Call of Duty*: World at War* with a distinct mission: Release the PC and console versions, through the publisher Activision, on the same date and with comparable features. Boasting a track record that

Fire, particle effects, and physics are powerfully enhanced through threading and new algorithms.

includes *Spiderman* 1* and *2* and over five years of simultaneous cross-platform releases, the Treyarch development team had the chops to pull off this latest release.

The high-resolution, high-frame rate objectives for *Call of Duty*: World at War* put considerable demands on the target platforms, an area where the benefits of multi-core processing become evident. In an interview with *PC Games Hardware*, Cesar Stastny, Director of Tools and Libraries at Treyarch, commented on the strong push toward the multi-threading of applications. "Game developers are among the lucky ones that had to start doing this [multi-threading] earlier," Stastny said. "We have been developing games for the past three years for multi-processors. We learned a lot of lessons that other software developers have yet to go through."

Friends Don't Let Friends Get Fragged

For the first time, *Call of Duty: World at War* has a multi-player cooperative play (co-op) on the PC, a breakthrough that brings a new level of entertainment to gamers. Over the past ten years, some games have had a co-op mode that typically involved gamers playing with or against computer-generated bots, but if a gamer wanted to play through the entire campaign, they were on their own. With Treyarch's new multiplayer co-op mode, up to four human players can experience the entire campaign together as one single team. One of the technical challenges that was overcome during development involved the complexity of letting four players connect and interact over the Internet on four separate machines, each with its own individual strengths. The smoothness of gameplay in co-op attests to the success in this area.

A newly added competitive co-op mode combines the multiplayer and single-player approaches. An individual player is playing through the single-player storyline in real time, but also gains points for kills and accumulates streak modifiers, which increase the overall point number. However, if a player gets killed and goes down, a teammate can revive him, but the player loses points for getting shot, while at the same time the teammate gains points. This provides a very different dynamic to the gameplay.

Developing *Call of Duty**

The core development team producing *Call of Duty: World at War* has a storied history dating back to an earlier company called Grey Matter, which produced the perennial favorite *Castle Wolfenstein** for the PC, as well as another popular PC video game, *King Pin**, in the late 90s. The team also created an expansion pack for *Call of Duty 1* called *United Offensive**, which is widely considered the highest-rated expansion pack in game history.

This same team, after regrouping under the name Treyarch, has not rested on its laurels, despite having produced some of the most popular first-person shooters (FPS) in the history of gaming, including *Return to Castle*

*Wolfenstein**, which is still played today. Having devoted nearly two years to *Call of Duty: World at War*, Treyarch has focused its hard-won expertise on raising the reality quotient and packing this latest release with features that make the most of an advanced technology platform.

First Person Insights into Development

Visual Adrenaline talked to Cesar Stastny, director of Tools and Libraries, Krassimir Touevsky, lead programmer PC team, and Adam Saslow, associate producer PC, to gain insight into the development process for *Call of Duty: World at War*, which sometimes kept developers hammering away on their keyboards for 18 to 20 hours at a stretch during the demanding, yet rewarding, process.

Visual Adrenaline: *Have there been any particular advances in the Modern Warfare engine to adapt it for this latest* Call of Duty *release?*

Cesar: The *Call of Duty: Modern Warfare* engine was already multi-threaded and running really well on dual-core systems. We completely utilized its worker threads system and threw a lot more work at this thread to crunch. We added new full screen effects, dynamic water, fire and burning effects, dynamic foliage, our own physics library, sound occlusion, and vehicles. We added squads to multiplayer and an entirely new cooperative gameplay mode. And, because it is more fun to play with your friends, we added a Friends list and game invites to the PC version.

Visual Adrenaline: *How well versed is your development team in multi-threading techniques?*

Krassimir: The core of the team is comprised of engineers that spent the last three years working on high-profile, cross-platform, next-gen titles—so we're well versed in multi-threading. Intel supplied us with their latest tools, including the Intel® VTune™ Performance Analyzer and the Intel® Thread Profiler, which are absolutely best-in-class. They also provided formal training to introduce some of the engineers to the tools. And, our engineers who had used the tools in the past benefited from learning about all the latest features.

Improvements in the game physics are reflected in the behavior and movement of vehicles in the game.

Visual Adrenaline: *What kinds of challenges did you have to deal with to create the co-op feature?*

Adam: The biggest challenges we faced when implementing co-op were network traffic and the fact that one player's machine doubles as the server. Our multiplayer game comes with a dedicated server that can be run on a virtualized server at hosting companies. This unloads the players' machines from computing the game state. But for co-op we had to use the machine that is hosting the match to run both

Former *Quake* 3* world champion Kornelia Takacs brings a spirited woman's sensibilities and a sharp competitive drive to Treyarch design work. Kornelia works closely with Cesar Stastny, Krassimir Touevsky, and Adam Saslow to refine many of the PC-specific aspects of the game.

Kornelia unexpectedly got hooked on competitive gameplay when her roommate downloaded Qtest for *Quake 1*. "That was the first time that I experienced what it feels like to play an FPS game against another person!" she said. "Needless to say, once *Quake* was released I continued playing. I began gaming on a more serious level after I won a tournament that held its finals at GDC. I joined the Cyberathlete Professional League (CPL) when it was established and I continued competing at various tournaments for several years."

Calm nerves and a steady demeanor are prerequisites both for tournament play and game development. "You must have the ability to keep calm during your match whether you are winning or losing," Kornelia observed. "Strategy and timing are crucial. Working at Treyarch, I have the opportunity to play our game, but playing in team games at the office is a more relaxed and pressure-free environment compared to playing a one-on-one match in front of thousands of people."

Kornelia is feeling positive about the *Call of Duty: World at War* release. "We have great features on the PC," she said, "that make it easier for you to find servers that your friends and clan mates are playing on. I like that the multiplayer game mode settings on the PC have all of the options that appear on the Xbox* 360 version. Capture the flag . . . need I say more? Nazi Zombies mesmerized everyone at our office. Even people that were working on different titles were playing it for hours on end (after their regular work day, of course). When I play Nazi Zombies I feel like someone on a home improvement show cleaning up the place . . . with a double-barrel shotgun." ▪

the client and the server. Also, co-op maps are much larger than the multiplayer maps because they are actually the same as our single-player campaign maps. A lot of scripting events happen in these maps, which increases the load on the server as well as on the network layer. To reduce the impact on the server, we moved some of the scripting to the client. To some degree, this is possible because of the availability of more cores and more threads, providing additional headroom without compromising gameplay action.

Visual Adrenaline: *What were your programming objectives for the release of* Call of Duty: World at War*? What new technologies did you exploit?*

Adam: We wanted the PC version to match or exceed the game experience on the console. PC gamers who invest in better CPUs and GPUs will get higher resolutions (1650x1080, 1980x1200) and higher frame rates than are possible on the consoles. We believe the PC platform's ability to scale in both areas allows PC gamers to realize improved, more realistic gameplay.

Visual Adrenaline: *Which tools did you use to complete the product development and optimization?*

Cesar: We used VTune and the Thread Profiler. We already had a threading library in place, but we will definitely consider Intel® Threading Building Blocks in our future development toolset. Using Intel's tools, we managed to resolve some thread synchronization issues in our engine and to improve our threading model overall by reducing the amount of synchronization primitives and waits. Because of this, PC gamers will enjoy smoother gameplay, as well as a more immersive experience when playing *Call of Duty: World at War*.

Visual Adrenaline: *What is the most difficult problem in terms of keeping the frame rates and video quality high?*

Adam: You really need good frame rate for shooters. We started with the best FPS game

engine available, which was already performing well. All we had to do from a technical standpoint was add more visuals, physics, and other features without dropping the frame rate. We added ocean simulation, dynamic foliage, improved rag-doll physics, realistic vehicle simulation, and more. Physics, in particular, makes heavy use of the processor, so the use of parallelism serves up strong dividends. I am happy to report that the frame rate is still solid.

Visual Adrenaline: *Any tips or guidelines for other developers working on similar types of projects?*

Cesar: Get the Intel performance tools early in your development cycle—even if you are doing console development. They can help you optimize your tools and art pipeline before you hit full production.

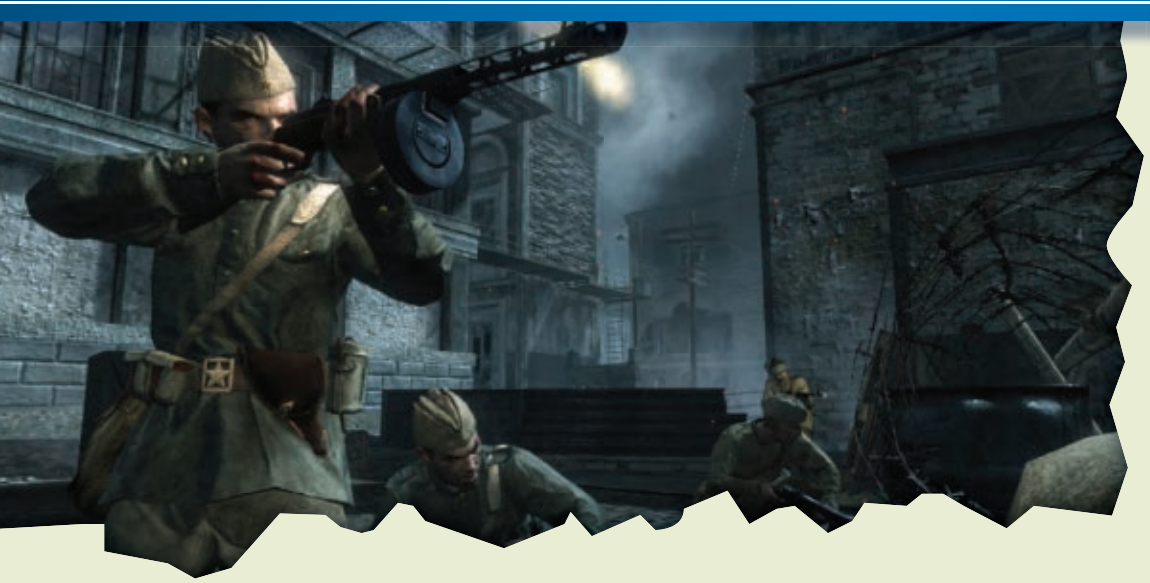
> *"I HAVE BEEN BUILDING SOFTWARE ON INTEL® HARDWARE SINCE THE EARLY DAYS OF PERSONAL COMPUTING. IT IS ALWAYS EXCITING TO SEE WHAT WE CAN ACHIEVE WITH THEIR CONTINUALLY EVOLVING TECHNOLOGY."*
>
> CESAR STASTNY,
> DIRECTOR OF TOOLS AND LIBRARIES, TREYARCH

Gaming Trends and Future Developments

Mark Lamia, studio head at Treyarch, sees a number of trends that are changing computer games. "Aside from great leaps in power that we as game developers will continue to harness in areas like graphics, physics, AI, and audio," he said, "we are seeing more and more gamers spending their time online playing our games and participating in online communities."

In response to this trend, Treyarch is investing heavily in all aspects of online gaming. *Call of Duty: World at War* was designed for online gaming, incorporating competitive multiplayer and—for the first time in the franchise

Whether battling alone or with a band of brothers, *Call of Duty: World at War* captures the grit and gruesome reality of war.



In co-op mode, as many as four players can link up to tackle a mission together.

history—online co-op play. To keep the online community grounded and informed, Treyarch also enhanced the CallofDuty.com site, where members gather to critique, comment, and contribute.

Mobile gaming is another prime area of interest. Mark noted, "I remember a couple of years back seeing some of the first laptops appear at a QuakeCon* and noticing how convenient it was for gamers to get together and play or have LAN parties."

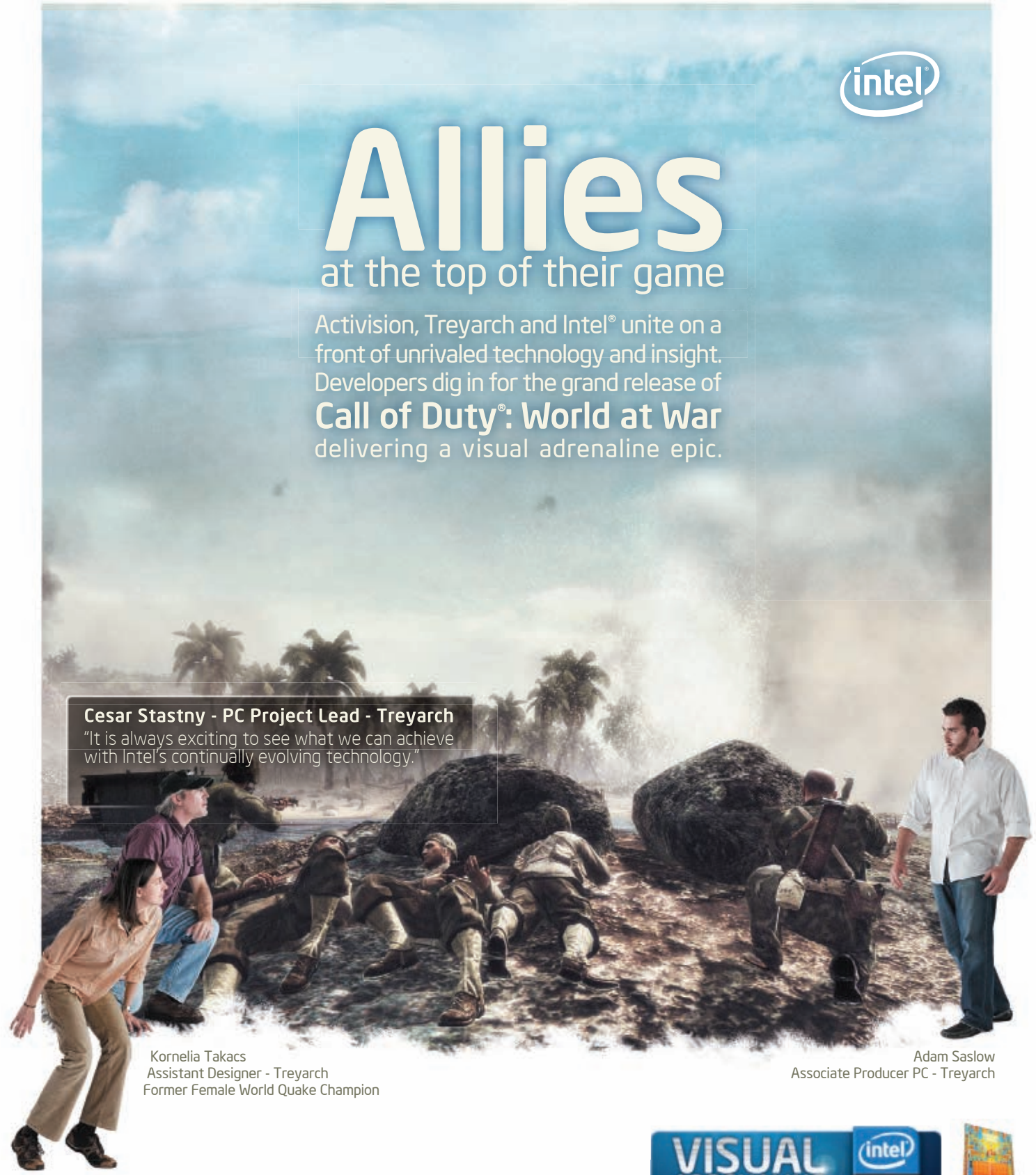
Mark was convinced at that point that as long as the hardware was powerful enough, laptops would be recognized as serious gaming rigs. "As CPU and graphic capabilities continue to improve in laptops," Mark observed, "they will continue to grow as a key platform for gaming." In support of this trend, Intel Software and Services Group offers the Intel® Laptop Gaming Technology Development Kit (TDK), a set of libraries that make game code more aware of the laptop environment (including battery life and wireless strength).

The future looks bright for PC gaming as platform advances continue to expand possibilities. "Treyarch is looking forward to supporting all of the exciting capabilities that the PC has to offer to gamers," Mark said, "and continuing to work with Intel to help us fully realize that power, so that PC users can continue to enjoy higher-resolution visuals, larger-scale multiplayer games hosted on customizable and dedicated servers, and more user-created mods." ▪

ABOUT THE AUTHOR: LEE PURCELL

Having survived the frenetic energy of Silicon Valley in its heyday, Lee Purcell now writes on high-tech and alternative energy topics from a rural outpost in the Green Mountain State. Lee blogs on alternative energy topics at lightspeedpub.blogspot.com.

To subscribe to *Intel® Visual Adrenaline*, go to www.intelsoftwaregraphics.com



Learn more about Intel® Visual Computing at: www.intel.com/software/visualadrenaline

VISUAL ADRENALINE intel Software

MATURE 17+
M
Blood and Gore
Intense Violence
Strong Language
ESRB CONTENT RATING www.esrb.org

treyarch

ACTIVISION®

BOSTON, MASSACHUSETTS

PHOTO BY ERIC BAETSCHER

THE HISTORY OF THE GAME DEVELOPMENT industry is a short one. But in the half century that people have been playing games on video screens the art form has undergone a Cambrian explosion of growth and diversification. Here we take a regional look at the industry's history, sifting through the various talent pools that have collected across North America to discover what exotic life forms have taken root. Here we start out in Boston, Massachusetts.

DON'T LET OUR YOUTH GO TO WASTE

Video gaming as we know it today can trace its birthplace to the Massachusetts Institute of Technology in Boston, MA.

During the fifties the university was a hotbed of computer hardware research with large-scale machines like the vacuum tube-based Whirlwind and smaller (though still room filling) transistor machines like the TX-0 housed at its campus and nearby Lincoln Laboratory. Splitting off from this research activity, two MIT engineers named Ken Olsen and Harlan Anderson formed the Digital Equipment Corporation (DEC) in 1957 to manufacture cutting edge solid-state computers. Soon, their first fully integrated machine was unveiled—the PDP-1. A radical new piece of technology, the PDP-1 could perform 100,000 additions per second and came equipped with magnetic core memory along with a variety of peripherals including a typewriter, a paper tape reader, and a CRT. Although it was housed in a unit approximately the size of two large refrigerators and cost $110,000 (in 1960 dollars), the machine could easily be turned on and off without the help of an engineer. Personal computing had arrived.

In the summer of 1961 MIT became the owner of a PDP-1 and installed it in the university's computer research lab. At that time, a loose knit group of faculty and students were gathered around MIT's student run Tech Model Railroad Club, drawn together by a love of gear hacking and science fiction. The group began to experiment with the PDP-1 and worked up an ad hoc plan to do something interesting with the computer during its off hours. Even though the machine was intended for complex scientific calculations such as nuclear weapon simulation, they looked at its capabilities and in a tremendous conceptual leap, decided that what it really needed to do was run a space game. The resulting two player SPACEWAR! game was completed in 1962. Steve Russell developed the initial version with contributions from J. M. Graetz, Alan Kotok, Dan Edwards, Peter Samson, and Wayne Wiitanen and it quickly became a favorite pastime at the research lab.

SPACEWAR! soon made its way to DEC's assembly floor where the game was used as the final test on outgoing PDP-1's. Because the computer's memory was magnetic, SPACEWAR! remained in memory after shut down, lying dormant until the computer was turned on in its new home, which more often than not was a university. SPACEWAR! spread across the country's higher education system inspiring new groups of young hackers to expand and refine its game play. Nolan Bushnell was an early convert to SPACEWAR!, first encountering the game at the University of Utah and later at Stanford. Seeing the enthusiasm for the game that sprung up wherever it was running inspired Bushnell to design his own arcade version called COMPUTER SPACE in 1971. By 1972 Bushnell created Atari. With that company's foundation, along with the introduction of Ralph Baer's Magnavox Odyssey in the same year, the video game revolution was underway.

MOVING IN STEREO

Throughout the seventies, as computer hardware dropped in price and grew exponentially in capability, a cottage industry of game developers flourished. Initially just one or two person teams, the most successful of these early developers often found themselves struggling to manage their company's growth as the personal computer market took off in the eighties.

Early in its history Richard Garriott's Origin Systems was a small game development team in Houston led by Richard while his brother Robert who lived in New Hampshire handled the company's finances. By 1985 Origin Systems had become large enough that it was decided to consolidate its entire operation to New Hampshire in order to complete work on ULTIMA IV. As the company continued to grow, Richard and the original programming team longed to return to Texas and in 1988 most of Origin Systems' development crew returned to their home state.

However, several key staff decided to remain behind and form their own development studio called Blue Sky Productions near Boston in Lexington, Massachusetts. Founded by Paul Neurath and Doug Church, Blue Sky's first game would be ULTIMA UNDERWORLD: THE STYGIAN ABYSS, which was published by Origin in 1992. The game was remarkably ahead of its time, and its smooth-running first-person engine combined texture mapped 3D polygons with the immersive role-playing of the ULTIMA series. id Software would release its ground-breaking WOLFENSTEIN 3D only a few months later and the two companies found themselves friendly competitors in the early uncharted territory of first person 3D games (indeed, John Romero worked as a Commodore 64 programmer

JEFFREY FLEMING *is patiently waiting for the release of* GUITAR HERO: SIEGE. *Email him at* **jfleming@gdmag.com**.

at Origin for several months at the New Hampshire location and he remained in contact with Paul Neurath after returning to Texas to co-found id). Though very different in tone, ULTIMA UNDERWORLD and WOLFENSTEIN 3D would together mark a sea change in video game tastes.

In 1992 Blue Sky merged with Ned Lerner's (creator of CHUCK YEAGER'S ADVANCED FLIGHT TRAINER) development studio and became Looking Glass Technologies (later Looking Glass Studios). The combined studio created a follow up to ULTIMA UNDERWORLD called ULTIMA UNDERWORLD II: LABYRINTH OF WORLDS and in 1994 set up shop in Cambridge, Massachusetts on the outskirts of Boston.

During its almost ten-year run, Looking Glass would create some of the most evocative 3D games of the modern era. The company built a reputation for designing games that were thoughtful, original, and emotionally resonant. Titles like SYSTEM SHOCK (1994), FLIGHT UNLIMITED (1995), TERRA NOVA: STRIKE FORCE CENTAURI (1996), THIEF: THE DARK PROJECT (1998), and SYSTEM SHOCK 2 (developed with Irrational Games in 1999) all carry the distinctive imprint of Looking Glass.

Unfortunately, in 2000 the company ran into financing difficulties and was forced to shut its doors. Because Looking Glass was home to an incredible stable of talent, its developers would become key figures in many of the high profile companies and games that we see today. Among the many alumni of Looking Glass are Seamus Blackley who would later spearhead Microsoft's Xbox project, Warren Spector, who went on to create the beloved DEUS EX, and Emil Pagliarulo, who became the lead designer on FALLOUT 3. Of the founders, Paul Neurath remained in the Boston area, forming Floodgate Entertainment, Doug Church moved on to California to work on TOMB RAIDER: LEGEND and BOOM BLOX, and Ned Lerner joined Sony Computer Entertainment's Tools and Technology group.

PLANET OF SOUND

MIT has always been an engine of new ideas and talent for the Boston game development community. Most recently two researchers from MIT's Media Lab, Alex Rigopulos and Eran Egozy, brought about a radical shift in the game industry with the creation of GUITAR HERO. When they formed Harmonix Music Systems in 1995, their goal was to bring the interactive music making research that they had been doing at the university to a wider audience. After creating FREQUENCY and AMPLITUDE as well as several titles in Konami's KARAOKE REVOLUTION line of music games, Harmonix teamed up with RedOctane to produce the guitar peripheral-based GUITAR HERO. Since its release in 2005, GUITAR HERO has become a cultural force, forever changing the audience for video games. Leaving the GUITAR HERO series in the hands of Activision, Harmonix has since moved on to create the ROCK BAND franchise, further expanding on its mission of bringing the joy of music creation to non-musicians.

DIRTY WATER

In 1997 a group of Looking Glass developers including Ken Levine, Jonathan Chey, and Robert Fermier split off to form their own studio called Irrational Games. Irrational's first effort was a co-production with Looking Glass on a SYSTEM SHOCK sequel. Released in 1999, SYSTEM SHOCK 2 revisited the deep space horror of the original game with improved graphics courtesy of a new engine based on the work done for THIEF.

Over the next several years Irrational would open a studio in Canberra Australia, create FREEDOM FORCE and its sequel, a new TRIBES game, as well as a new entry in the long-running SWAT series. In 2004 Irrational revealed plans for a new game called BIOSHOCK that would be a return to the themes and style of the great Looking Glass games of its roots. By 2006 Irrational had joined Take Two Interactive and its studios were renamed 2K Boston and 2K Australia. With the release of BIOSHOCK in 2007, the studio made good on its promise and delivered a game that was as intelligent and mature as it was visually striking. BIOSHOCK's overwhelming critical and commercial success was telling affirmation of the design principles first described by Looking Glass a decade earlier.

THE SPRAWL

The Boston area is home to many other notable developers as well. Blue Fang Games, creators of the ZOO TYCOON series, Turbine, the developers behind THE LORD OF THE RINGS ONLINE: SHADOWS OF ANGMAR, and Tilted Mill Entertainment, designers of the city building games CHILDREN OF THE NILE and SIMCITIES SOCIETIES have all established studios in the region. Baseball legend Curt Schilling's newly formed 38 Studios, which is working on a MMO with design contributions from R.A. Salvatore and Todd McFarlane, is also located in the Greater Boston area.

Rockstar New England is based just north of Boston in Andover. Formerly known as Mad Doc Software, the studio picked up development on JANE'S ATTACK SQUADRON after Looking Glass was forced to shelve the project. Since then they have worked on DUNGEON SIEGE: LEGENDS OF ARANNA, EMPIRE EARTH II, STAR TREK: LEGACY, and BULLY: SCHOLARSHIP EDITION.

Blue Sky's ULTIMA UNDERWORLD: THE STYGIAN ABYSS.

Contract work is an important part of the game industry and Boston area's Demiurge Studios is making a name for itself as Unreal Engine experts after work on BIOSHOCK, MASS EFFECT, and the BROTHERS IN ARMS series. Also near Boston is Orbus Gameworks, a company that produces metrics gathering middleware as well as metrics consulting

In addition to MIT there are a number of higher education resources in the Boston region that provide game development related programs. The Berklee College of Music and Northeastern University are both located within the city while the Center for Digital Imaging Arts at Boston University is in nearby Waltham and Worchester is home to the Polytechnic Institute.

Boston occupies a unique space in the history of game development. Talent has always been drawn to the city, lured there by educational and entrepreneurial opportunities. And wherever smart people gather to do interesting things, someone will surely make a game of it.

RESOURCES

COMPUTER HISTORY MUSEUM
www.computerhistory.org

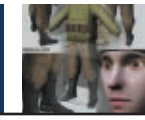
IGDA BOSTON
www.igda.org/wiki/IGDA_Boston

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
http://web.mit.edu

THROUGH THE LOOKING GLASS
www.ttlg.com

GAME DESIGN






GAME ON!

at BECKER COLLEGE

Join our Award Winning Game Design Program!

1st Place Winners! 2008

MASSACHUSETTS GAME CHALLENGE

Your passion is your career! The computer video industry is going strong – which means the demand for skilled, knowledgeable designers and developers continues to grow. Gain the skills and earn the degree you need to compete in this lucrative field at Becker College.

- Degrees in Game Design and Game Development
- Earn a Bachelor's degree from an established and growing program
- Learn to work as a valued member of a design team
- Internship programs offered
- Financial aid is available




BECKER COLLEGE

61 Sever Street, Worcester, MA 01609 (877) 5 BECKER
Visit us at www.becker.edu/gamedev

OUR RATING SYSTEM :

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

VISUAL ASSIST X

Noel Llopis

VISUAL ASSIST AND I, WE GO BACK A LONG ways. It all started on a fateful afternoon about eight years ago, during the Visual Studio 6.0 days. It was love at first sight and we quickly became inseparable. I even wrote a passionate review for *Game Developer* Magazine declaring our undying love to the world at large. Our love affair continued for a couple of years until Visual Studio .Net got in the way. Before long our relationship was marred by constant slow downs and crashes. Infatuation quickly gave way to frustration and then turned to loathing. We soon parted ways.

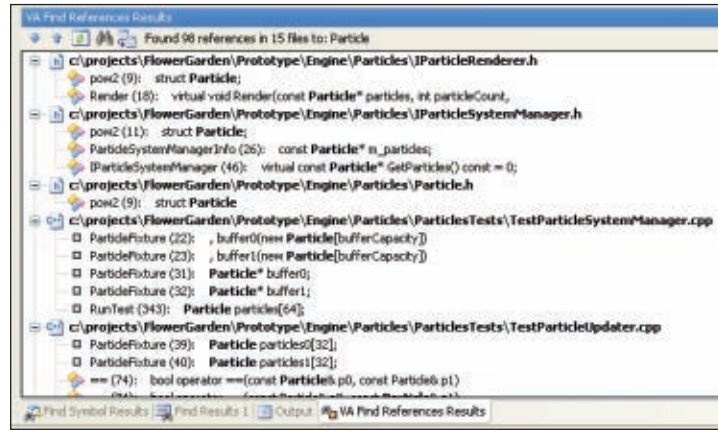
During the next several years, I thought fondly of my time with Visual Assist. I tried to replace the void it left in my life with IDE macros and homebrewed add-ins, but it wasn't the same. Every so often I would download an evaluation copy, but it confirmed what we both knew all along: We were just not right for each other. Far from being able to work things out, we had grown even further apart. Visual Assist was bogging down the IDE to a halt and crashes were more frequent than ever.

Summer gave way to winter, and winter to spring. Several more springs followed. New versions of Visual Studio came and went. Visual Assist tacked on an X to the end of its name. We both grew older and wiser and learned to live without each other. Then one day, friends and coworkers convinced me to give our relationship one last try. We were too good together to give up, they said.

REUNITED

To be fair, Visual Assist X had a lot of ground to make up. Not only did it have to address all past failures and disappointments, but it also had to compete with plain, vanilla Visual Studio. In its current version, Visual Studio 2005 is way ahead of .Net in terms of working Intellisense and extensibility. This was going to be a tough reunion.

Not really knowing what to expect, I installed Visual Assist X, and fired up Visual Studio. A new panel called VA View and VA Outline immediately greeted me. Apparently Visual Assist really changed during our time apart! Neither of these two views seemed to be worth the screen real estate they were taking up, so I turned them off.



Visual Assist X's Find in References results.

My biggest complaint with Visual Assist from a few years ago is that it slowed down the text editor very noticeably. I would type away furiously, trying to keep up with my train of thought, and Visual Assist would bog down the editor, slowing me down, and ultimately defeating the purpose of its own existence: To help me be a more effective programmer. So with that in mind, I immediately hit the options dialog box. And there it was: I could finally turn off the coloring of text. Awesome! However, I realized that even with text coloring on, Visual Assist X wasn't causing any slow downs anymore so I could type as fast as I wanted and it could keep up with the best. Even better then.

I then proceeded to put it through its paces. When someone installs Visual Assist, they might like the font coloring, but they're really looking for the improved autocomplete and navigation. Was it really an improvement over Visual Studio? Visual Studio's Intellisense has greatly improved in the last few years, but Visual Assist seemed a bit more reliable and its autocomplete has a better interface. The biggest improvement was that Visual Assist seems to be able to parse your code without causing any hitches or locking the IDE for a few seconds like Visual Assist's Intellisense does so frequently. It also was able to detect changes in my code right away, making the work session very smooth. Ironically, Visual Studio's Intellisense was still parsing my code and slowing everything down, and there's no option to turn it off, so I ended up disabling it the hard way by removing some dlls.

Visual Assist continues to provide all the functionality it made me fall in love with it in the first place. Open in Solution is essential to my everyday workflow. Even though it can be easily replaced with several free add-ins, Visual Assist X user interface is much more polished than any other alternative I have seen. Open Corresponding File allows you to toggle between cpp and header files and is also must-have functionality for me. Again, this one can be easily replaced with a macro, but this time the homebrewed approach wins out because I often want to toggle between a cpp or header file and its corresponding unit test file, which Visual Assist has no way of doing. Finally, I don't know how I managed to survive without the autocompletion of file names in `#include` directives. Especially when working

with deep file hierarchies from some middleware packages, it really is a lifesaver.

REFACTORING

In addition to the oldies-but-goodies, I was pleased to find some new features that became instantly useful. Find References proved itself invaluable for finding where particular symbols are used and completely replaced my usual "Find in files" with more accurate results and better display (even indicating whether they're being read or written and offering a pop up of surrounding code). Multiple Clipboards is also great and it completely blows away the counter-intuitive multi copy-and-paste built into Visual Studio.

The refactoring functionality is new to Visual Assist X, and it was one of the things that made me try it again. Even though most of the refactorings didn't seem particularly useful (Extract Method, Add Member, Document Method), one really stood out and became a welcome addition to my toolbox: Rename. With it, you can easily rename any symbol, and Visual Assist X will track down everywhere that the symbol is referenced and change it there as well. It's like a smart Find and Replace in Files operation. Other refactoring operations seemed intriguing, but failed to deliver: Change Signature had a lot of promise, but unfortunately it just updated the declaration and definition of a function, not any of the calling code (even when the signature change only involved removing a parameter).

The best feature of them all though, is that I did not get a single crash during the last few weeks with Visual Assist. It clearly has matured and become rock solid.

DON'T TRY TO CHANGE ME

Things aren't quite perfect though. Visual Assist X continues to violate the cardinal rule of productivity enhancers: Don't force me to change the way I work. In other words, I should be able to continue working with the exact same keystrokes I always do and everything should work the same way, as if Visual Assist weren't there. There's nothing more frustrating than selecting some text and pressing * to replace it with that symbol and have Visual Assist X take over and comment it out. Or start writing some code involving a variable called particleSystem to have Visual Assist X rename it to ParticleSystem because it matches the class name. Thanks, but that's not what I wanted! Fortunately, it's possible to work around most annoyances by turning off most "smart" autocorrect features.

Then there are the times when Visual Assist X becomes lost. And I mean totally lost. I don't blame it, really. This is usually in the depths of macro-ridden code, which would be hard for anybody to figure out what's going on. Unfortunately, all my unit tests involve such macros, so Visual Assist X loses most of its edge there, highlighting every variable as incorrect and constantly trying to rename them to something else.

Visual Assist X is loaded with lots of small features, intended to appeal to different programmers with different work styles. So you're guaranteed to develop your own set of favorite features. Most of them can be turned off, so the ones you don't like won't get in the way.

Any long-term relationship involves accepting the good with the bad, and this is no different. After setting a few ground rules by turning off most options, we're really enjoying each other's company again, and code seems to flow from my fingers faster than ever. We're back together for good.

VISUAL ASSIST X

★★★★

STATS

Whole Tomato Software, Inc.
1733 Fessler St.
Englewood, FL 34223
www.wholetomato.com

REQUIREMENTS

Visual Studio versions VS2008, VS2005, VS2003, VS2002, and VC6.

Configuration used in this review: Visual Assist X 10.4.1647 with Visual Studio 2005 SP1 under Windows XP SP3.

PRICE

$249

PROS

1. Great productivity help.
2. No crashes or slowdowns.
3. Can turn most things on and off depending on your preferences.

CONS

1. Still can't turn everything off.
2. Not customizable.
3. Sometimes gets "lost."

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**.

Book Review by Bijan Forutanpour

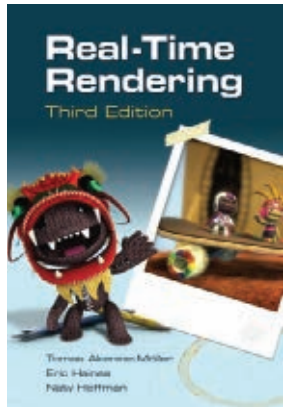
REAL TIME RENDERING, THIRD EDITION

BY TOMAS AKENINE-MOLLER, ERIC HAINES, NATY HOFFMAN

EVERY NEW GENERATION OF GRAPHICS hardware that hits the stores brings with it raised expectations for games to look better, play better, and get the adrenaline flowing. Usually, but not always, the goal is photorealism, at interactive speeds of 60 rendered frames a second. It often falls upon the graphics programmer's shoulders to deliver on this promise and be current with the

latest 3d programming techniques, and this is where the third edition of *Real Time Rendering* is invaluable.

The challenge for any graphics programmer is finding current, coherent information and obtaining an understanding of the big picture with sufficient detail. Frequently this means scouring the web and searching for papers from past SIGGRAPH conference proceedings, GDC developer conferences, hardware vendor SDK documentation, and miscellaneous articles and tutorials. Books such as the *Games Gems* and *GPU Gems* series, as valuable as they are, are hardcovered collections of papers that provide a good reference for solving specific problems. The old college computer graphics text by Foley and Van Dam is sorely out of date. The gap that *Real Time Rendering* fills is the need for a comprehensive, modern book that focuses on theory, current algorithms, and architecture that are key in understanding the field of computer graphics. It is written in an academic style where the goal is to make the reader understand the material without feeding the answers. It achieves this goal by avoiding code listings, excessive mathematical formulas, and keeping the big picture in mind without focusing excessively on DirectX or OpenGL details. The book assumes some previous knowledge of computer graphics and programming. As with any good university level textbook, each chapter concludes with pointers for further reading and references.



With the exception of a few introductory chapters and a few of the closing chapters, every chapter has been expanded and updated from the 2nd Edition. The order of presentation of some of the chapters has changed, and some chapters have been split into several new chapters to allow more depth of coverage. In some cases, the same material is presented, but has been rewritten for clarity.

The book begins with the usual fundamentals and introductory chapters, such as an overview of the graphics rendering pipeline, and the math behind different types of transformations. An interesting new addition to the 3rd edition is the chapter on the architecture, history and evolution of the graphics processing unit. Its amazing to think that in 2001 I purchased a GeForce 3 card for four hundred dollars, a card that is now considered six generations old.

The chapters on visual appearance and texturing come next, and improve upon the explanations of bump mapping and normal mapping, and are updated to include the topics of parallax, relief, and displacement mapping. One of the highlights of the Third Edition is its handling of the heavy subjects of lighting and shading, previously covered in a single chapter. The Third Edition has expanded on the subject matter and divided it into three new chapters, on advanced shading, environmental lighting, and global illumination respectively. Radiosity, colorimetry, and lighting are explained as well as a much-expanded section on the theory and practice of bidirectional reflectance distribution functions (BRDF). The new subject of subsurface scattering is broached and explained, a must for any good skin shader. The subject of global illumination is also handled very well, introducing the new topics of ambient occlusion and different forms of precomputed lighting and precomputed occlusion into the picture. Once again, the last section in this chapter, which includes further reading and resources is worthwhile reading as well.

The chapter on image based rendering has been updated with the latest methods, adding post-processing effects such as motion blur, depth of field, fog, and displacement techniques. One of the very valuable and more math-centric chapters is on intersection test methods, which provides a pretty thorough list of geometry intersection calculation methods. There are many more valuable chapters that have been carried over from the Second Edition, covering topics such as collision detection, pipeline optimization, culling methods, tessellation, curved and subdivision surfaces. The book concludes with discussions on frame buffer hardware and future directions. Finally, *Real Time Rendering* has an incredible wealth of references in its bibliography, with over one thousand four hundred references.

Overall, the Third Edition of *Real Time Rendering* is definitely a worthwhile upgrade from the Second Edition, as there is plenty of new information and improved presentation to justify the purchase. And for those who don't own the previous edition, the third edition is definitely a must-have, because the material covered is essential knowledge for anyone writing code for interactive 3D graphics applications—or games.

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**.

BOOK REVIEW

★★★★★

STATS

TITLE
Real Time Rendering, Third Edition

AUTHOR
Tomas Akenine-Moller, Eric Haines, Naty Hoffman

PRICE
$89 (hardcover)

PUBLISHER INFORMATION
A K Peters, Ltd.
888 Worcester Street
Suite 230
Wellesley, MA 02482
www.akpeters.com

Published: July 2008
1045 pp.
ISBN-10: 1568814240

NOEL LLOPIS

DATA ALIGNMENT

Part 1

I GREW UP ON Z80 AND X86 ASSEMBLY. I happily moved data into my registers from anywhere in memory and back without ever giving it a second thought. Life was nice and simple. It wasn't until years later, when I started dealing with caches, vector units, and different architectures that data alignment became a big deal. Today, data alignment is an inescapable reality, and we all have to deal with it one way or another.

Data alignment refers to where data is located in memory. All data is at least 1-byte aligned, meaning that it starts at any one byte in memory. Data that is *n*-byte aligned is located somewhere with a memory address that is an exact multiple of *n* bytes.

IT'S ALL ABOUT PERFORMANCE

We care about alignment for a single reason: Performance.

In modern hardware, memory can only be accessed on particular boundaries. Trying to read data from an unaligned memory address can result in two reads from main memory plus some logic to combine the data and present it to the user. Considering how slow main memory access is, that can be a major performance hit. Some platforms such as the PowerPC choose not to hide such an inefficient use of hardware and simply raise a hardware exception flagging the invalid memory access. Neither scenario is particularly attractive.

Even when reading from fast, level-one cache memory, unaligned access to data will use up more cache lines, making poor use of the available memory and evicting data that might be accessed again. As a result, sections of your code can thrash the cache and become major performance bottlenecks.

Vector operations usually force programmers to deal with data alignment explicitly. To squeeze more performance without increasing the frequency of CPU cores, hardware designers have been putting more emphasis on vector operations (which are just a type of SIMD instruction—single instruction multiple data). One vector instruction can operate on four or more data points at once, considerably improving the performance of a program that operates over each data point serially. Vector units often have a restriction that they can only work on data aligned on a particular boundary. For example, Intel's and AMD's SSE vector instructions operate on data aligned on 16-byte boundaries, so it is the programmer's responsibility to align the data properly before feeding it to the vector units.

In general, the more you interface with the hardware directly, the more careful you will have to be about the alignment of your data.

WORKING WITH ALIGNMENT

You can only deal with alignment in low-level languages such as assembly or C. Higher-level languages often don't expose exact memory locations or provide any means to change them, making all sorts of opportunities for optimization impossible.

In C/C++, finding the alignment of some data is very easy. Just examine the least significant bits of the memory address where the data is located. Memory alignment restrictions are almost always based on powers of two, so to detect whether something is aligned on a particular power of two, we can "&" the memory address with a particular bit mask and check for any non-zero bits.

For example, the following code checks if some data is aligned on a 16-byte boundary: `((uintptr_t)&data & 0x1F)`. Notice the `uintptr_t` cast to be able to perform a bit operation on a memory address. Casting the pointer to an `int` instead is almost guaranteed to cause problems in the future because it assumes that pointers and `int`s are the same size, which is not the case in most 64-bit architectures.

Generalizing that technique, Listing 1 (found online at www.gdmag.com/resources/code.htm) shows a function that verifies the alignment on a particular power of two boundary. We'll be using a lot of bit tricks when dealing with memory addresses at this level, so make sure you're nice and comfy with all that bit twiddling.

Setting the alignment for a piece of data, unfortunately, is not nearly as easy. The C/C++ language falls short again, without a standard way of dealing with alignment. Not only that, but we'll need to use different strategies to align data depending on how it is allocated: statically, on the heap, or on the stack. We're left at the mercy of compiler-specific extensions and our own home-brewed solutions.

STATIC DATA

Static variables are those not allocated on the heap or the stack. That includes global variables and class static variables.

The two most common compilers we have to deal with as game developers (Visual Studio and gcc) offer solutions to allow us to align data at a particular memory boundary. In both these cases, the alignment has to be a power of two.

Visual Studio provides the

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****.*

__declspec(align(#)) extension. Any variable or type tagged with that extension will be correctly allocated on the specified alignment. gcc also provides its own extension, but of course, with different syntax: __attribute__ ((aligned (#))).

We can create a macro that allows us to specify alignment attributes in a platform-independent way. Unfortunately, __declspec(align()) precedes the symbol it modifies, and __attribute__((aligned())) comes afterwards, making the macro slightly uglier because it needs to wrap the symbol (see Listing 2 online at www.gdmag.com/resources/code.htm). With this macro we can now write DATA_ALIGN(int buffer[16], 128); in any platform.

Sometimes we want to align all instances of a particular type on a certain boundary. For example, a 4x4 matrix class that we're going to manipulate with SSE instructions always needs to be aligned on a 16-byte boundary. In that case, we can apply the alignment attribute to the type, not just to specific instances.

```
__declspec(align(16)) struct
Matrix4x4
{
    float m[16];
};
```

Specifying the alignment of a member variable forces the compiler to align it relative to the start of the struct (or class) it belongs to. Going back to our previous example, if we add a Matrix4x4 structure to a Waypoint structure, the compiler will pad the Waypoint data so that our matrix is a multiple of 16 bytes from the beginning of the structure (see Figure 1).

But what if the Waypoint structure is placed at a random memory location? Does that mean that all bets are off as far as the absolute alignment of the matrix variable? Fortunately, the C standard comes to the rescue here because it states the following: the alignment for a struct has to be a multiple of the least common multiple of the alignments for all of its members. That's quite a mouthful, so a simpler way to look at it is that, if alignments are always powers of two, a struct will be aligned the same way as the member with the largest alignment.

That is really good news because now the Waypoint struct is guaranteed to be aligned on a 16-byte boundary, which means that the matrix member variable will also be correctly aligned on a 16-byte boundary.

HEAP ALIGNMENT

You would hope that tagging a type with __declspec(align(#)) (or the gcc equivalent) would also align the object correctly when it's created on the heap. No such luck. Alignment attributes are only used at compile-time, and the C runtime knows nothing about them when you request to allocate a particular type from the heap. Fortunately, we can take control of heap allocations and force the alignment ourselves.

In most cases, malloc will return a 4-byte aligned block of memory. We can create our own version of malloc, aligned_malloc, that allocates memory with any alignment by building on top of standard malloc and pad the front of the memory block until it's aligned correctly. The only catch is that we need to remember to call our version of free, aligned_free, on that memory when we're done with it.

aligned_malloc will allocate a block of memory large enough, find the first address within that block that has the correct alignment, and return that address. aligned_free needs to call standard free with the originally allocated memory block, so we need a way to go from our aligned pointer to the original pointer. To accomplish this, we'll store a pointer to the memory returned by malloc immediately preceding the start of our aligned memory block.

The implementation for aligned_malloc and aligned_free is shown in Listing 3 (found online at www.gdmag.com/resources/code.htm). There are a couple of interesting implementation details:

- We don't know what alignment malloc will return, so we need to account for the worst case. That means that our block has to be large enough to hold the requested allocation size, one pointer, and up to the full alignment size (minus one).
- More bit fun: To find the next address with a particular power of two alignment, we can add the full alignment minus one, and then round it down to the nearest alignment (by masking off the bits corresponding to the power of two of the alignment).

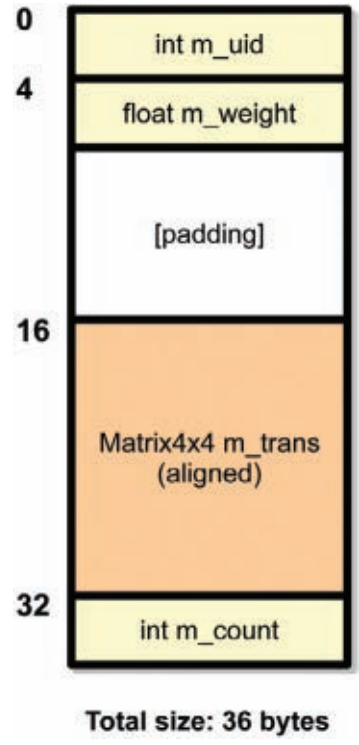


Figure 1 Waypoint containing aligned Matrix4x4.

Some platforms implement their own version of aligned_malloc. So if you don't need cross-platform compatibility, you should look at what your platform provides. For example, Microsoft has a function called _aligned_malloc, and other platforms have the semi-standard function memalign, both of which work in a very similar way to the version we just implemented.

Our aligned malloc and free are perfect for allocating plain memory blocks, but what about creating objects with a particular alignment? Next month we'll cover that and we'll explore the mysteries of stack alignment.

Example code for this article can be found at www.gdmag.com/resources/code.htm.

STEVE THEODORE

THE M-WORD

Consolidation hits the 3D software market

IF YOU'VE BEEN IN A CRUNCH-TIME MEDIA blackout for the past several months, or have shut down your internet connection to avoid election news, or are the only game artist on the planet who's never received a Youtube link via email, you may have missed an interesting little tidbit of news. On October 23 we learned that Avid is going to sell Softimage, the Montreal-based developer of Softimage XSI to Autodesk. Now that the deal has gone through all three of the biggest 3D modeling and animation packages belong to one single company.

Even if you managed to miss the announcement, you can probably predict the immediate reactions anyway. In the XSI community, the dominant mode was shell-shocked. The "resistance is futile" jokes and Borg-themed Photoshop jobs could not disguise the level of emotion in the air. The poster on the XSI forums who simply wrote, "I think I'll cry," wasn't kidding. There was a smattering of optimists suggesting the deal would give more people access to some of XSI's best tech. A few pragmatists found consolation in the idea that the conglomeration would give cross-package data transfer the attention it deserves. But the most common reactions were shock and anxiety.

It's hardly surprising that the possibility of being forced to abandon the comfort and security of a familiar environment would give XSI users the heebie-jeebies. The official Pixel Pusher line has always been that any professional game artist should be competent in at least two packages. But even traditional artists are famous for being emotionally attached to their tools (never ever venture an opinion about Kolinsky sable brushes in mixed company!). For us, who spend so much of our lives poking at one particular set of dialogs or buttons, the thought of being forced to swap them for a different unfamiliar set of dialogs and buttons is deeply disturbing. The fact that some XSI fans were so distraught they'd consider switching to Blender out of pique is an index of how emotional this issue can be.

> **"Any professional game artist should be competent in at least two packages. But even traditional artists are famous for being emotionally attached to their tools."**

What's surprising, though, is that a similar miasma could be seen in the Max and Maya forums after the Great Buyout Announcement of 2005. Emotions ran high even for those not affected directly. More commonly, though, users were grimly pondering the future of graphics software in general, rather than the fate of any particular package. Some naysayers worried that technology would stagnate without the underdogs like XSI striving to gain an advantage through innovation. Others fretted that consolidation in the industry means the exciting, Wild West days of graphics are really over. And many users of all three packages speculated that the lack of competition would lead to price gouging.

ADVANCE TO GO

Before we pronounce the graphics software business dead, we ought to look at this deal in its historical context.

These kinds of corporate dramas are unsettling for artists because they are an unsubtle reminder that we creative types are dependent on huge impersonal corporations. "Masters of the Universe"-style MBA analysis isn't part of our job descriptions, so it's hard for us, as mere users, to figure out how to respond.

Reviewing a little bit of history is often a good way to get some perspective; here's a very abbreviated walkthrough of the life and times of Softimage to help you understand the news.

XSI may be the youngest of the big three graphics packages, but Softimage the company is one of the oldest firms in 3D graphics software. The original "Softimage Creative Environment" debuted in 1988, but in an economic environment very different from today's. 3D graphics was very closely akin to rocket science. For one thing, it was mysterious new high-tech discipline, and for another, you needed an exotic workstation that cost upward of $50,000 (that's $65,000 in today's dollar, for a machine less powerful than an iPhone) to do either one. It was a very esoteric, very pricey business.

Softimage 3D was the first commercial application to offer artist-friendly IK (1991), which quickly became the gold standard for computer animation. Many

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****.*

seminal CGI films of the early 1990s were animated in Softimage 3D, most famously Jurassic Park.

Those early days of the CG revolution were heady times. Hollywood stood ready to hose money onto anybody who could render a good-looking triangle. Siggraph veterans still murmur nostalgically about the heydays of studio parties. And those boom times were good for the company. In 1992, the Montreal firm went public to much acclaim.

Success also changed the way the industry worked. By the mid '90s, the explosion of 3D game development shifted the industry dynamic. The ranks of 3D artists and animators expanded enormously, but few games companies could afford to put the equivalent of a luxury sports car under every animator's desk. Affordable PCs with primitive graphics cards started stealing business from workstations, and PC-based packages like Autodesk's 3D Studio started making inroads against pricey workstation software.

RECEIVE $45 FROM SALE OF STOCK

Microsoft, naturally, wanted to see the PC forces prevail. In 1994 it bought Softimage for $130 million—a pretty high price given that the whole 3D software market was only around $60 million a year back then. But box sales weren't the real goal. Microsoft needed to port a top-end workstation graphics package to Windows and legitimize the market for high-end graphics on Windows.

For many Softimage veterans, the events of last month may have an eerily familiar ring, right down to the "you will be assimilated" jokes (although in 1994, the reference was forgivably fresh). The Microsoft acquisition was not a very pleasant experience for Softimage users. Not only were many Unix devotees forcibly converted to a new operating system, the demands of porting and cross-platform development shunted innovation to the sidelines. It took almost seven years for Softimage 3D to get from version 3.0 to version 4.0, and the package lost a lot of its technological edge to newer platforms like 3D Studio Max and Maya.

It's not surprising that the survivors of that first buyout react suspiciously to the latest.

By 1998, Microsoft had achieved its strategic objective: Windows had triumphed and graphics workstations were headed for the history books. Softimage became superfluous. Microsoft sought a buyer and found Avid, a rising power in the digital video editing business, and one that coveted the company's visual effects and compositing tech. Even if the core 3D business was losing steam, the deal still ran a cool $285 million, a price dot-com bubble, but pretty impressive nonetheless.

Avid's stewardship was a lot healthier for Softimage as a tech company. Softimage XSI, the long overdue gut-rehab of the aging Softimage 3D, was released in 2000. The product started out a bit slow. Version 1.0 had no poly modeling tools, but gained steam with impressive tech and clean new architecture. Many artists have lusted after XSI's Gator system for mapping one mesh onto another (supporting everything from texture transfers to skin weight matching), its non-linear animation mixer, FaceRobot facial animation technology, and most recently the high-performance ICE system for node-based custom object creation.

WIN SECOND PLACE IN A BEAUTY PAGEANT, COLLECT $11

Unfortunately, the engineering success of the product did not translate into success for Softimage's owners. Avid's core business has been hit hard by the proliferation of lower-cost video editing software like Apple's Final Cut Pro.

Even though the company's main product was profitable, it wasn't profitable enough. To get a sense of the

scale, you might note that the $35 million sale price for the company won't even cover Avid's losses for the third quarter of this year. As times got leaner, Avid needed to focus on protecting its core video editing business, so it started hunting for a buyer early this year. Autodesk, home to both of XSI's main rivals, was not the first buyer who was approached. But it was the final one, which is the one that counts.

What lessons can you learn from this little history?

First, it doesn't provide a lot of evidence for conspiracy theories about monopoly power. Supplying 3D software is pretty small potatoes in the grand scheme of capitalism. It's been said that there are only about half a million seats of full 3D packages in the world, which sounds like a lot, but is actually less than the number of people in the beta program alone for Photoshop. It's not a market in which achieving dominance is a huge financial win.

All three turnovers at Softimage have been driven by strategic concerns that didn't have anything to do with monopoly power or market domination. Microsoft bought Softimage to catalyze the switch from workstations to PCs. Avid bought it to solidify its effects and compositing business and saw modeling and animation through that prism. The most recent sale didn't originate with a sinister plot from inside of Autodesk—it originated with Avid's accountants.

YOU ARE ELECTED CHAIRMAN OF THE BOARD

The more interesting (but also more depressing) aspect of this story, though, isn't concerned with money. You could read the whole thing as a stirring tale of steadfast devotion. It was user loyalty that sustained Softimage during the drift of the Microsoft years, when technical sluggishness might have let Max and Maya completely marginalize the original Softimage. The emotional reaction to the news is proof of how viscerally loyal users are to their favorite tools.

Unfortunately, that loyalty is a double-edged sword. The last few versions of XSI were consistently excellent, but no combination of cool features and good design managed to seduce away enough users from other packages to secure Softimage's future. They competed on tech and features, and did a great job, but it wasn't enough to overcome the entrenched loyalties of Max and Maya fans.

Individual artists might admire this feature or that bit of UI, but collectively we're reactionaries. We stick with what we know. On top of that, most studios have tools and processes designed around a particular package, and no one is eager to chuck those investments for the sake of sexy icons or a cleaner interface.

We don't really reward tools companies for pushing the envelope. Even when something new breaks onto the scene, we try to shoehorn it into our existing workflows rather than embracing the new. We're the last people to denounce monopolies or phone the Federal Trade Commission. Most of us have already folded our hands by becoming emotionally attached or technically beholden to particular bits of software. If you're in the same camp as the XSI user who wrote, "They'll have to pry my license from my cold dead fingers," you live in a virtual monopoly already.

GET OUT OF JAIL FREE?

That's not to say things aren't going to change. The absence of major-league alternatives will definitely give Autodesk a much freer hand in choosing both its price points and research directions.

The fact that the company's track record to date is pretty benign is comforting, but the knowledge that we're dependent on corporate altruism from here on out should give us pause. Autodesk has put some genuine effort into trying to explain the deal to users (there's an interesting interview featuring the general managers of Autodesk and Softimage up on the Autodesk web site, with more info promised as the deal solidifies), but apart from reassurances that XSI isn't going to go away overnight, the Magic 8-Ball is pretty cloudy.

The uncertainty is tough, particularly for anxious XSI users, but really for us all.

We know the mantra: "It's the artist, not the tools." But in practice, it's sometimes hard to say where the artist leaves off and the tool begins. The feeling that such an important part of our lives is out of our control is unnerving.

What can we do about it? As individuals, we have to be open to new software and new ways of working, which makes for an environment where companies have a real incentive to give us new and better tools. As studios, we must invest in in-house tools rather than rely too faithfully on any single vendor. As an industry, we must push harder for more consistent open standards in data formats and for open-source tools so we can make our pipelines less dependent on the ups and downs of individual companies.

None of these steps will magically unwind the clock, but they will give us a little more input into this critical part of our lives. Or, we could all switch to Blender ... but man, I hate those menus! It's not like Max. You couldn't pay me enough to switch.

DAMION SCHUBERT

WRITING BETTER, SHORTER SYSTEM DOCS

IT'S ALWAYS ASTONISHING TO SEE THE vast disparity in standards for game design documentation. Every team and company seems to have its own ideas of how to present ideas. I've also seen hundreds of sample design documents from dozens of would-be designers when they submit them as work samples along with their resumes.

All these documents seem to have at least one thing in common, though. Most game design documents I've seen really stink.

The lack of standards in writing good game design documentation has resulted in most designers and design teams shooting from the hip, throwing everything but the kitchen sink into a game design document, and then being flabbergasted when programmers choose not to read them.

Here's a hint: if programmers are asking you to rewrite your 10 page design document into a half-page of bullet points—and you can—your design document presentation probably has room for improvement. So what makes a good game design document? Here's another hint: what did your programmer just ask you to do?

CONSIDER THE AUDIENCE

The most important thing to remember for any document is to consider the audience that it is written for. Design documents for game systems are usually written for multiple audiences. Other designers read it to get design consensus, producers and project managers read for scheduling, Q/A reads for building test plans, and programmers read for actually building the thing.

While the other audiences are important, the most important goal of a design document is to ensure that the design is communicated clearly, and the feature is developed to spec. As a result, a good place to start is to actually ask your coders to evaluate your design documents, and ask them if there is a way that you could make these documents more effective.

> **"Design documents that are too long and too detailed are almost impossible to keep accurate and up-to-date."**

Sometimes, getting the answer to this is like pulling teeth, but when I ask the question, I typically get the same three requests: keep it short, keep it up-to-date, and preferably deliver it in a bullet point list. Simple.

And now, paradoxically, I'm going to spread this into a two page article.

KEEP IT SHORT

The most common mistake I see in game design documentation is that most of them are way too long, and way too in-depth. Too many designers try to write books, thinking that every word is important. We are, as tradesmen, too often in love with the sounds of our own voices.

Who cares about the history of the weapon factory that devised the Huge Honking Gun? Who wants to know why the Gods had decided to allow players to unlearn all the boxes in their skill tree? Most programmers I know have better things to do than read all that piffle. Such as, say, code the game.

And let's not forget: shorter design documents are easier to write and maintain. It's always faster to write a concise two-pager than a 30 page design bible—as well as faster and more accurate to incorporate changes into as the design shifts throughout the creative process.

Longer doesn't necessarily mean better. The Gettysburg Address was only 256 words long. The U.S. Declaration of Independence: 1337. Both seemed to do a pretty good job of getting their point across.

KEEP IT RELEVANT

I was once being interviewed by a creative director when, midsentence, another designer literally wheeled in a few copies of the design document. On a trolley. The design document was more than a foot thick, and required multiple binders for a single edition.

These design documents are almost always doomed. I know of another studio that referred to a similar tome as the 'Big Book of Stupid'. It had been written in preproduction, but was so huge and unwieldy that the programmers never referred to it, which resulted ultimately in nobody taking the document seriously. Attempts to refer programmers to the document resulted in responses akin to, "Yeah, whatever, what do I really need to do?"

DAMION SCHUBERT *is the lead combat designer for Star Wars: The Old Republic at BioWare Austin. He has spent nearly a decade working on the design of games, with experience on Meridian59 and Shadowbane as well as other virtual worlds. Damion also is responsible for Zen of Design, a blog devoted to game design issues. Email him at* ***dschubert@gdmag.com****.*

Needless to say, the final game looked nothing like the design document.

Design documents that are too long and too detailed are almost impossible to keep accurate and up-to-date. As this happens, the documents lose their credibility, which results in them being used less, which ultimately ends up in design documentation largely being a huge time sink that most projects cannot afford.

LEARN WHAT NEEDS NO EXPLANATION

Yes, your guild system may require 6 different confirmation boxes, but each of those confirmation boxes does not require its own GUI mockup and page-long explanation of what each button does. It can even cloud the issue—if five of the dialogs are identical but one is crucially different, you run the risk that the difference will be lost to an implementer not reading carefully.

Don't skimp on your design documents: if you care about a detail that's different, include it. But by the same token, don't document features that are obvious or trivial. We know that the little X in the corner of the GUI will close it.

Don't try to solve problems that aren't yours to solve, such as how much memory needs to be reserved for quest variables. Tell the coders, as simply as you can, what the end user requirements are ("the player needs to have 25 quests at a time, and may complete 5000 quests in his lifetime that can't be repeated"). Don't forget, programmers like solving problems, and when it comes to nitty gritty technical issues, they're probably better at it than you are anyway.

And don't cut and paste from other documents. Doing so is a good way to ensure that one of the two docs will go out of date as your design shifts via iteration. If another document has crucial information to help people understand your system, link it instead.

ILLUSTRATE

A picture is worth a thousand words, and nobody likes reading design documents anyway. It's on you, the designer, to find the simplest, most concise way to communicate your system to others. Consider any possible avenue: Visio diagrams, video clips from other movies and games, mocked up screenshots, excel graphs. Visual aids can often get the point across the user much quicker and more effectively than text. These are also much easier to discuss freely in a meeting with implementers or other designers.

And sometimes, text is the best way to illustrate. Writing a short piece of prose can help make complex system interactions more clear, and can also help to make the feature more exciting. But use this sparingly: Anytime you "sex up" a document with whiz-bang descriptions of coolness, ask yourself if you are making it harder for programmers to find their task list. I've found the most effective way to use such textual illustrations is in clearly-defined callout boxes. This allows programmers to safely ignore the prose while they complete the feature.

> **"Callout boxes (like this one!) are a great way to emphasize or sex up a document without making it harder for programmers to find the information they really need."**

PLAN FOR ITERATION

When tasked to spec out a new system, most designers over-design. They build fantasy land wish lists with every possible bell, whistle, and kitchen sink that they can think of, and throw it into the design document—just in case. This approach is usually flawed on two fronts. The first is that you never have time to do everything you want to do, and those features just end up bloating the document, making the feature seem bigger and harder to code. The second, and far more crucial, is that once the feature is partially coded and suddenly tangible, other, more crucial and obvious improvements and fixes become clear.

Don't cut those ideas, but plan for the iteration. Mark core ideas that must be completed no matter what at the front of the document, and give them all necessary detail. Put less crucial or well-formed ideas in an appendix near the end, clearly marked as a place where design iteration could lead. Don't spend too much time designing for these features that are likely to be cut. And don't get emotionally involved with any idea that is way down on the horizon.

THE POLITICAL IMPLICATIONS

Designers, you are not being paid by the word. More painfully, the opposite is true: there are severe political ramifications for design documentation that is too long and too in-depth. Big design documents feel like big features that take a big time investment. I have literally seen programmers call a feature a six-month feature simply by feeling the heft of the document. I've also seen projects get doomed because the sum of all their design documentation made the project an insurmountable mountain that could never be completed.

Lead designers and producers have very limited budgets and usually intractable time restraints to work within. Proposing a wacky system? If you're a lead and you have limited resources, what are you going to be drawn toward, the simple system described in a two-page document (counting illustrations!), or the 30 page manifesto, complete with pompous backstory, nonsensical design theory, and unworkable crack?

Ideas are cheap in this industry. What separates the good designers from the bad is the ability to get those ideas made, and then make those ideas fun. It's not enough to have a great idea – you have to figure out how to communicate it as well. Design small systems, plan for iteration, and identify places for expansion.

The best way to make this happen will vary from team to team, and project to project. But when in doubt, try brevity.

JESSE HARLIN

SQUASHED

iPhone compression options for music

THE CASUAL GAMES MARKET CONTINUES to boom, and with it come enormous leaps in mobile gaming technology. Apple's iPhone is now one of the most popular mobile gaming devices and has turned into yet another SKU that audio content creators may need to support in a multi-platform development cycle.

iPhone games, due to a 2GB maximum size limit, require an intelligent approach to audio footprint management. Thankfully, the iPhone supports a wide array of audio formats including Linear PCM, MP3, AAC, Apple Lossless, IMA/ADPCM, A-law, μ-Law, AMR, and iLBC. The iPhone also supports Audible.com's audible formats 2, 3, and 4. However, as they are a proprietary format with proprietary encoding technology, these options aren't available for game development. Noticeably missing, unfortunately, is the ogg vorbis format.

Music files are most likely to be the largest audio files in an iPhone game, so compression formats become that much more important. With so many options available, I decided to take a sample file, convert it into each of the available formats, and compare the results in order to find out which format offers the highest quality at the smallest size.

MY GRAND EXPERIMENT

For my control music file, I used a 44.1 kHz/16-bit .WAV file with a length of 30 seconds and a size of 5MB. Additionally, the file was authored so as to loop seamlessly from end to end upon playback. Following the conversion process, the largest files were the A-law and μ-Law files, both of which compressed with a 2:1 ratio down to 2.52MB. To encode the A- and μ-Law files within Peak, the source file needed to be converted into an .AIF file. On the positive side, following compression, both the A- and μ-Law files retained their seamless loop points perfectly. However, there was a noticeably loud continuous crackle of compression artifacts which seriously degraded the sonic quality of both files.

The next smallest compressed file used Apple's own Apple Lossless Audio Codec (or ALAC). A number of programs will convert into Apple Lossless such as dBpoweramp Music Converter or SoundConverter, but by far the easiest to use and most common program is a free download of iTunes. Apple Lossless compressed the source file down to 2.39MB and maintained the seamless loop while offering no audible degradation in sound quality.

Much like the A-law and μ-Law files, the IMA/ADPCM conversion within Peak necessitated a conversion to .AIF. Following conversion, the IMA/ADPCM file also retained a seamless end-to-end loop but had a much smaller file size than A-Law, μ-Law, or ALAC files with a decrease in its file size down to 1.35MB. While I didn't hear any noticeable change in the quality of the audio, it may be worth noting that the audio file was actually 8 samples longer after conversion into IMA/ADPCM.

For Apple's own AAC format, I again tackled the conversion process within iTunes which gave me two different quality settings: 128 kbps or 256 kbps. The quality of both was decent with even the 128 kbps file showing little difference from the source .WAV, even though some clarity was noticeably lost particularly in higher frequencies. The 256 kbps file compressed down to 950k while the 128 kbps file shrunk to an impressive 480k. However, conversion to AAC destroyed the seamless loop with an additional 3080 samples of silence tacked onto the end of the file in both instances as the conversion goes through a process of rounding the data up to the nearest frame boundary.

To test the .MP3 format, I again used iTunes for the conversion and compared two different quality settings: 192 kbps and again 128 kbps. While both files were smaller than the AAC files—707k and 472k, respectively—the same need to round up to the nearest frame boundary resulted in an additional 1764 samples of silence added onto the end of both files and an audible skip in the loop point. Sonically, the compression was beginning to take its toll and while the 192 kbps file sounded better than the 128 kbps file, neither sounded as good as the 128 kbps AAC file.

Lastly, since I was interested in figuring out what would have the smallest size but also the greatest quality, I decided to not test the iLBC or AMR file formats. The iLBC, or internet Low Bit-Rate Codec, is a format specifically designed for streaming speech online with a set quality level of 8 kHz/16-bit audio. AMR, or Adaptive Multi-Rate compression, is the standardized speech codec in the cellphone industry and uses a set quality level of 8 kHz/13-bit audio.

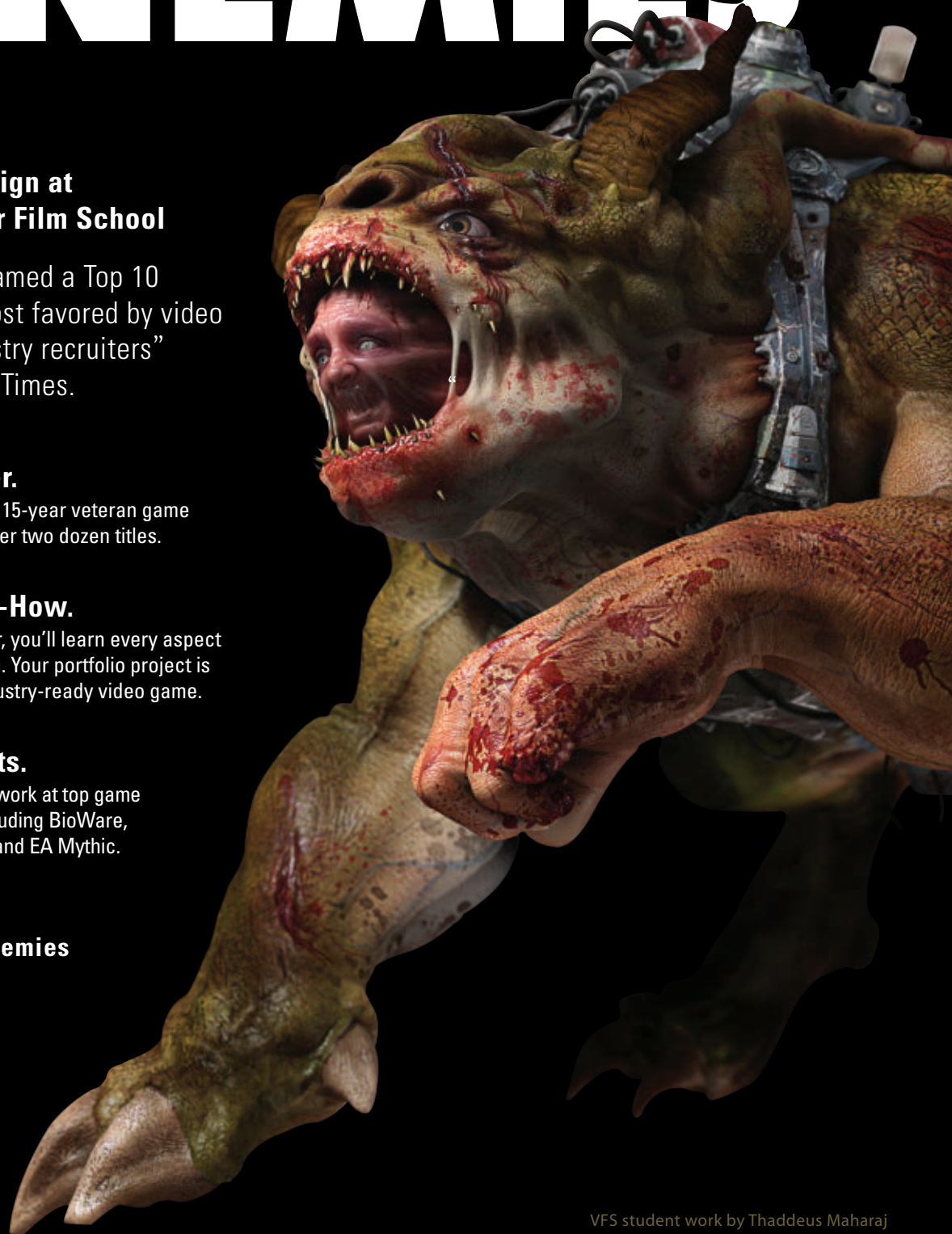
PARSING THE DATA

Of the larger files, with a serious degradation in audio quality and the worst compression ratio of all of the options, both A- and μ-Law are easily ruled out. Apple Lossless sounds excellent along side the original file and maintains the loop point cleanly, but doesn't save much in terms of file size. IMA/ADPCM compression sounds great and maintains the loop cleanly, but may not compress files enough if you're using multiple music files within the game.

As for the smaller files, AAC definitely outperforms MP3 compression—even at 128 kbps. However, both formats will require additional programming support to help skirt the issue of frame boundary limitations and the additional silence this adds onto the end of each file if music files are intended to loop.

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**.

Make More Enemies



Game Design at Vancouver Film School

Recently named a Top 10 School "most favored by video game industry recruiters" by the L.A. Times.

The Leader.
Dave Warfield, 15-year veteran game designer for over two dozen titles.

The Know-How.
In just one year, you'll learn every aspect of game design. Your portfolio project is a playable, industry-ready video game.

The Results.
Our graduates work at top game companies including BioWare, Radical, Relic, and EA Mythic.

vfs.com/enemies

VFS

VFS student work by Thaddeus Maharaj

ADVERTISER INDEX

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.

ARRESTED DEVELOPMENT

BUG ME NOT

Q/A reports from the depths of hell

THE PRESENCE OF THE MULTI-BORE LASER CANNON IS CLEARLY ANACHRONISTIC AND MAKES NO SENSE

Description: According to Space Chronicles of the Horsehead Nebula, vol. IV, the Multi-Bore Laser Cannon was invented by Granolian scientists in 8304 S. G. E. (pp. 115-117). However, the events of the game, particularly those that occur on Exodus Mining Station, appear to be occurring well before 8304, considering the fact that 1. Commander Banson states early on that, "the war back home has taken a large toll on us all," which seems to indicate that the Fourth Great War of Redemption is still ongoing at this time, and 2. the player can come across numerous objects which resemble the "hexafractal cuboids" as seen in Space Chronicles: The Chronicles (e.g. season 2 episode 3, season 2 episode 7, etc.) and it is understood to be canon that Space Chronicles: The Chronicles starts in 8228 S. G. E. (see Wikipedia article on Space Chronicles for detailed timeline), and 3. there is no satisfactory explanation for how Granolian science would find its way into United Space League territory given the punishing trade embargo that has been in place ever since the Treaty of Overt Purpose was signed by the Intergalactic Power Council (as described in detail on the back of the blister card packaging for McFarlane Toys' excellent action figure of High Rhetor Vindicto), which was over six Astroyears prior to the events depicted in the game. Please do something about this because if this is not fixed the player's immersion will be irrevocably shattered.

WHEN I WAS PLAYING THE GAME, THE GAME FROZE...

DESCRIPTION: I WAS PLAYING THE GAME AND IT FROZE.

Steps to reproduce:

1. Put the game in the disc tray and start playing it.
2. Walk 56 steps to the left and drink the green potion.
3. Explore the caves and proceed to the point where the wizard talks to you, then skip the dialog after about the third panel. I forgot what it said.
4. Hit the attack button repeatedly for five minutes.
5. Pause the game and leave it on the Pause menu for about as long as it takes to make and eat a microwave burrito, I think like 15 minutes
6. Unpause the game, turn on all the cheats, warp to the vampire's lair and shoot the panzerfaust at him over and over as fast as possible.
7. Notice that the game has frozen.

Repro rate: 100% (1 out of 1 try)

THE LATEST BUILD HAS A BAD TEXTURE

Description: the latest build of the game has a bad texture in the first room. Instead of wallpaper on the wall there is just a white image with text that reads "F—K THIS, I QUIT". Is that intentional? Just checking

GAME IS UNRESPONSIVE WHEN PLAYING OVER A 2400 BAUD MODEM

Description: When attempting the play the game while connected to the Internet via a 2400 baud modem, the user experienced multiple instances of lag and slowdown. It was so bad as to make the game unplayable, even on my state-of-the-art Pentium Pro system.

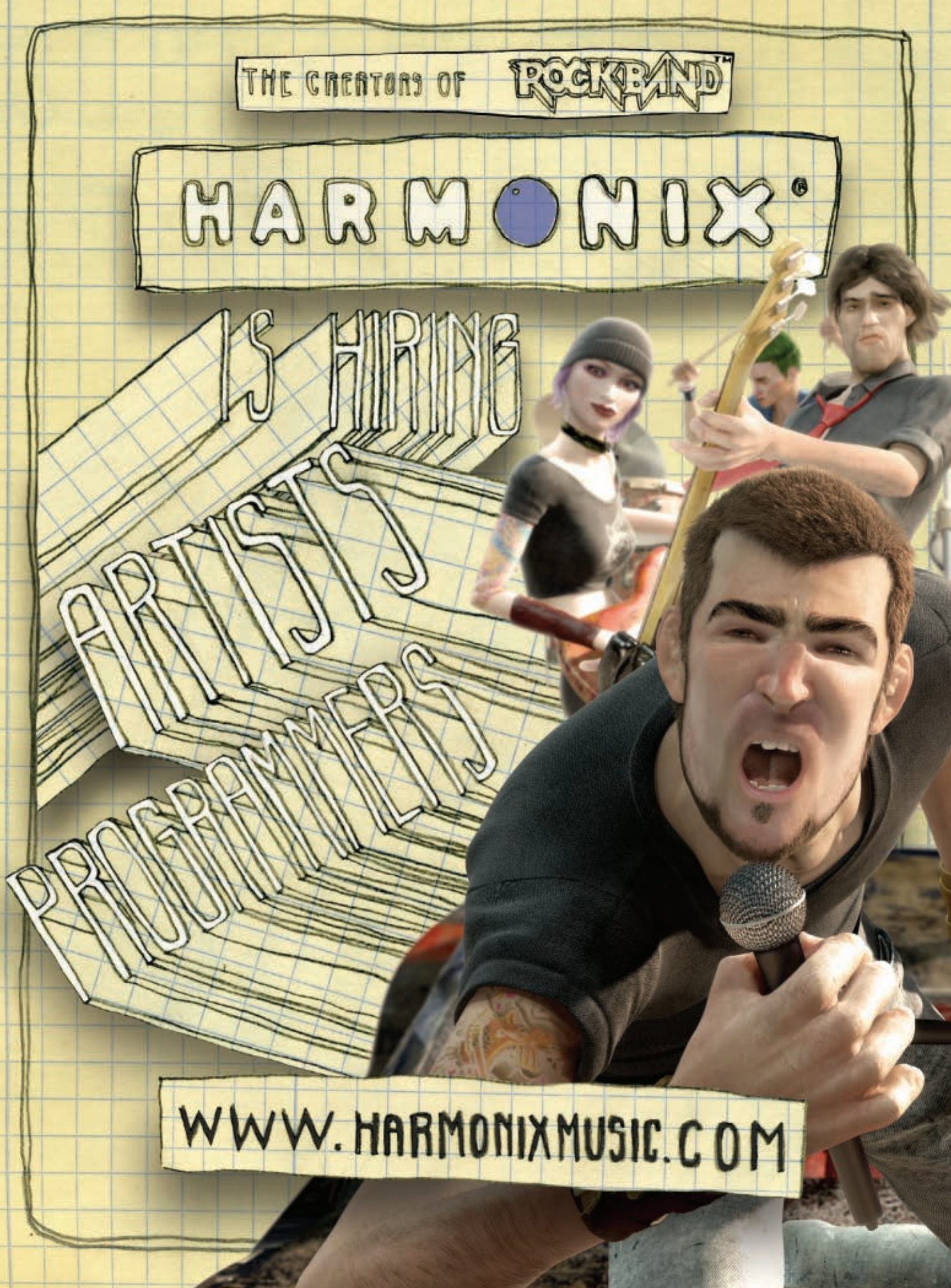
THE SNIPER RIFLE IS COMPLETELY OVERPOWERED

Description: Every time I play in a multiplayer match, other people keep sniping me and I lose the round. I don't understand why such an unbalanced weapon like the sniper rifle is even in this game. Nobody should be allowed to kill me instantly like that with one shot just because they practiced at the game and now have the skill to aim the gun while it's zoomed in and precisely target my head. It's just not fair.

IT'S SHOULD BE ITS, MORANS!!

Description: As anyone with half or maybe I should say a quarter or less of a brain already know, "it's" a possessive adjective and "its" the contraction of "it is." This mistake is so common and one of my biggest pet peeves in life. Unfortunately I have to say this game is riddles with these stupid and frankly amateur mistake. Every single time I receive a quest from a quest-giver I will read his text and he will say something like: "The forest? It's right around the corner" and obviously he should be saying "its right around the corner," when I inspect a item I read the descriptions like "this sword is magical, its power burns bright" but in correct English it would be supposed to say "it's power burns bright." Sorry to the team that made the game, but your idiots.

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


THE CREATORS OF ROCK BAND™
HARMONIX®
IS HIRING
ARTISTS
PROGRAMMERS
WWW.HARMONIXMUSIC.COM

RAD GAME TOOLS ©

WARNING: This product can make you more efficient.

425-893-4300 www.radgametools.com/granny3d

GRANNY 3D

BINK VIDEO

DYNOMATIC

MILES