United Business Media

>> GDC 2009 PREVIEW EDITORS' SHOW PICKS

FEBRUARY 2009

game developer

THE LEADING GAME INDUSTRY MAGAZINE

POSTMORTEM
SECRET LEVEL'S
GOLDEN AXE:
BEAST RIDER

+

SNAKES ALIVE
STACKLESS PYTHON
FOR CONSOLES

TOOL BOX
MOTIONBUILDER
2009 REVIEW

VISUAL LEARNING
INTRODUCING TOOLS
TO YOUR ART TEAM

SCANNING SOLUTIONS

CUSTOMERS INCLUDE:
EA, MIDWAY, SONY,2K,
I/O INT, BLUE CASTLE,
ROCKSTAR, UBISOFT

GDC BOOTH 6301NH

EYETRONICS
www.eyetronics.com
800.205.9808.EU +32.16.298309

game developer

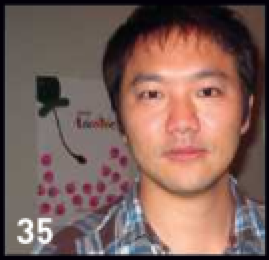
CONTENTS

FEBRUARY 2009
VOLUME 16, NUMBER 2

14

7

32


35

COVER ART: SECRET LEVEL ART TEAM

RECESSION-PROOF

ON THE SURFACE, THINGS DON'T LOOK TOO HOT for our industry right now. I was speaking with a friend recently about the current economic climate, saying that developers are losing their jobs across the globe, more studios are closing, and fewer seem to want to hire. I was being a bit of a doomsday prophet, but my friend suggested that this is actually a time of great opportunity, and thinking about it carefully, I may agree with him.

There are a lot of talented people out of work in a lot of major metropolitan areas. If you're the type of person who can motivate them to get together, this seems like a prime opportunity for new studios, new indies, and new ideas about the shape of the industry.

If you've always dreamed of implementing the Hollywood model, bringing the right person to the right project and then moving on, this is a good time for it. If you want to invoke the spirit of the bedroom programmer now that tools are getting to where that's actually viable without being a hardcore programmer/artist/designer hybrid, this is a good time for it. If you want to venture out and create your own IP, or revisit a lost genre, this is a good time for it. It might not be the best time to create a multi-million dollar epic—unless you're one of the big publishers, but even then it's a bit riskier than usual.

Games aren't going to go away during this recession. During the Great Depression, the legend goes that entertainment was the major industry that flourished. Books, movies, and condoms were among the best selling "non-essential" products. These all facilitated escapism from the poverty around them, without weighty consequences. While we're not quite at the depression level, games are an effective and appealing method of escapism in the best of times, and in times of trouble are even more enticing.

Of course it's rather difficult to do any of this if you haven't got a little bit of a nest egg to tide you over until you can release a game, or at least get funding. Some have said that the venture capital money has all dried up, but I don't think it's necessarily gone. Rather I think it's being hidden, Great Depression-style, under the proverbial mattresses. Games are still a good industry. People don't want to stop playing video games—but they might want to start paying less for them. If you can convince people with money that a smaller investment in a fledgling company making smaller games is a wise one, then you may be in business.

THE BEGINNING OF AN ERA?

It's very possible that this recession could usher in the next age of the indie. Smaller, less expensive games made by smaller, more agile teams seem like a very logical step, now that the industry structure is better able to support it, with no less than three venues on which to distribute content as a small team—downloadable console, direct to consumer PC downloads via Steam-like services, portals, or direct sale, and iPhone and potentially DSi downloads. Consumers have shown that they're willing to buy games like CASTLE CRASHERS in droves.

Indie developers have asked me on more than one occasion how to promote their products to the press without a big budget, or without a budget at all. It can seem daunting, but actually it's quite simple. What works for me, and for many other members of the press I've spoken to about the issue, is targeted personal emails. It doesn't cost anything more than time. Target the bigger blogs first, reading to see who writes about indies and in what context. See what they like and don't like, and choose a writer to contact. Send them an email explaining who you are, where you're coming from, what your game is about, and what you're trying to achieve. If you've got some nice production art to show, a snappy title, a YouTube video, or a playable demo, so much the better. Don't treat your game like it's the next big thing, or the most awesome Tower Defense clone ever, even if it may be. Be straightforward, humble, and realistic, and people will pay attention if your game seems interesting.

I'm hopeful that this recession will bring about a bit of an industry shakeup. It's up to the folks networking at this coming GDC and similar events. I would urge you to not think along the same old lines as before, and simply join an existing studio, or create a new traditionally structured team. This is a great time to experiment and try out new ideas, if you're the type who has them. If it doesn't work out, at least you won't have been idle by the time you have to take your next job in the trenches!

—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 Robert Steigleider
e: *rsteigleider@ubm-us.com*

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

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

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

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

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

United Business Media

WWW.CMPGAME.COM

IF YOU COULD CREATE A

VIRTUAL WORLD

FOR YOUR BUSINESS

GEORGIA

IS WHAT IT WOULD BE LIKE.

More than 60 video game companies—from Time Warner's GameTap and Atlanta's own Kaneva, Inc., to China's CDC Games and CCP North America, creators of EVE Online—have found Georgia is designed to help video game developers succeed. Our state's deep talent pool is fed by cutting-edge schools like the Savannah College of Art and Design, Georgia Tech, and Georgia State University. Financial support is obtainable from private investment capital and tax incentives provided by Georgia's Entertainment Industry Investment Act. Plus, Atlanta boasts the planet's premier airport and is the most heavily wired city in the U.S. with the fat pipes you need. Georgia is a world unlike any other, and we're ready to help you grow your game business. Contact the Georgia Film, Music & Digital Entertainment Office today.

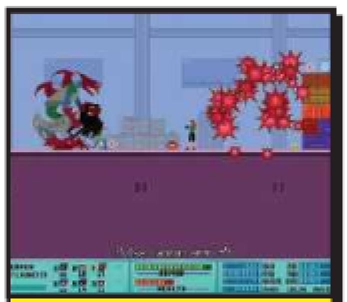
Georgia®

Visit georgia.org/gamedevelopment or call 404.962.4052.

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

TOP 10 INDIE FREEWARE GAMES

Picks by Indiegames.com editor Wee Tim Boon for the second half of 2008



IJI

Daniel Remar

http://tinyurl.com/ijigame

IJI is a 2D action platformer where players assume control of the titular character as she learns how to cope with waking up to find her world taken over by an alien race, guided by her brother who speaks through the research complex's loudspeaker system.

8BIT KILLER

Locomalito

www.locomalito.com/juegos_8bit_killer.php

This is a tribute to the classic Wolfenstein 3D in more than a few ways, complete with blocky graphics and an impressive chiptune soundtrack which pumps you up to fight Master Brain's multitudinous henchman.

MEAT BOY

Edmund McMillen and Jonathan McEntee

http://tinyurl.com/meatboy

Meat Boy is a challenging platformer created by the developers of Gish, Aether, and Coil, where players assume control of a nimble protagonist who must scale the walls of each area to reach the love of his life.

SPELUNKY

Derek Yu

http://tinyurl.com/spelunkygame

Spelunky is a procedurally generated 2D platformer created by Derek Yu, co-creator of the award-winning Aquaria. Here, players assume the role of a treasure hunter who ventures into the Colossal Cave in search of gold coins, gems, golden idols, and damsels to rescue.

VERGE

Kyle Pulver

http://kyle.nfshost.com/view.php?id=45

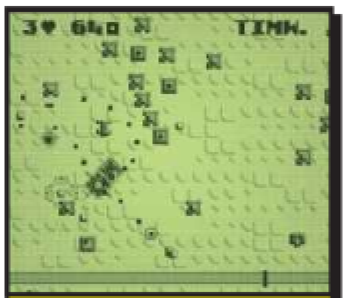
Verge is a 2D platformer created by the developer of Bonesaw: The Game and features music contributed by Alec Holowka (Aquaria). Players control an unnamed protagonist who has to traverse two worlds in order to solve rudimentary block puzzles and reach the exit of each stage.

YOU HAVE TO BURN THE ROPE

Kian Bashiri

www.mazapan.se/games/BurnTheRope.php

You Have to Burn the Rope is an extremely short game that features good pixel art and a catchy tune. There is only one solution to the problem, though the credits will be remembered long after you've managed to beat the final boss. This is also a IGF 2009 finalist in the Innovation category.

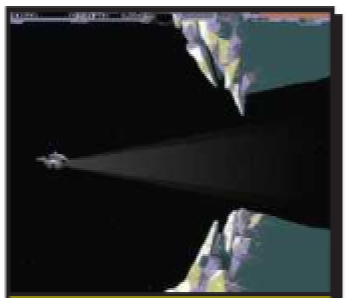


NOBODY SHOOTER

Orel

http://tinyurl.com/nobodyshooter

This is a remake of the award-winning Everyday Shooter, where players have to collect coins from the enemies they've defeated. Jonathan Mak, developer of the original, says it's "pretty awesome."

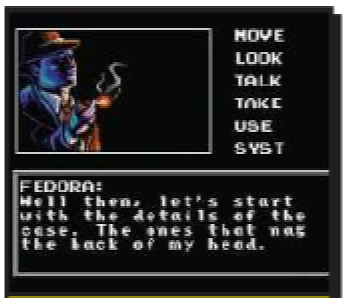


THRUSTBURST

Umlautgames

http://umlautgames.net/thrustburst/downloads

Thrustburst is a remake of an old arcade game called Urthwurm, where players must activate their thrusters prudently to navigate treacherous caverns. Thrustburst has great pixel art coupled with addictive gameplay, and they're not kidding around with the difficulty of this space adventure.



FEDORA:
Well then, let's start with the details of the case. The ones that nag the back of my head.

FEDORA SPADE

Orchard-L

http://studioeres.com/games/fedora

Fancy a murder mystery or two? Look no further than Fedora Spade, the Shadowgate series-inspired 8-bit adventure with a touch of humor and some plot twists that'll keep you hooked until the very end.

THE MANIPULATOR

Virtanen Games

http://koti.mbnet.fi/erkkavir/themanipulator.php

Here's a 2D puzzle platformer created by the developer of Seven Minutes and Virtual Silence. You play a psychic who has the ability to control the minds of others and influence them to carry out your orders.

CORRECTION

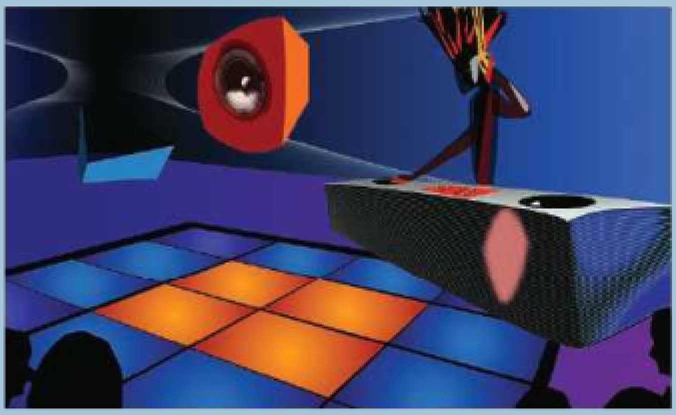
We would like to clarify last month's Gamescape column in regard to Monolith. The company informs us that its subsidiary Touchdown Entertainment utilizes a branch of the LithTech engine which is developed separately and marketed as Jupiter EX while Monolith continues development of its own engine technology.

FUN FOR EVERYONE
GAME ACCESSIBILITY HIGHLIGHTS

GAMERS WITH DISABILITIES can have a hard time playing games that weren't designed with their needs in mind. Recently there has been lots of exciting work done to make games playable for more people. Here are a few interesting projects:

AIBICOM
www.komodoopenlab.com/index.php
Komodo OpenLab's Aibicom, or Asynchronous Interpreter of Binary Commands, is a development library for making binary control applications that use a single button or switch. What makes Aibicom different from other binary control methods like scanning is that instead of users pushing a button to get the program to do what they want, they only interact with the software when it does something that they don't want it to do. For a thorough discussion of how Aibicom works, see the paper "A Novel Asynchronous Access Method With Binary Interfaces" by Jorge Silva, Jorge Torres-Solis, Tom Chau, and Alex Mihailidis in the October 2008 Journal of NeuroEngineering and Rehabilitation (www.jneuroengrehab.com/content/5/1/24).

WINTERMUTE ENGINE
www.dead-code.org
The Wintermute Engine Development Kit is a game engine designed to create graphical "point and click" adventure games. Designed by Czech programmer Jan Nedoma, the engine can work with traditional 2D games as well as 3D characters on a 2D background. It also has several features to make games more accessible to visually impaired players. Wintermute allows users to send text to a text-to-speech synthesizer, which can highlight active areas on the screen with keyboard shortcuts. The



engine lets players pause the game at any time so they can take as long as they need to hear text read aloud or examine their on-screen surroundings.

VOCAL JOYSTICK
http://ssli.ee.washington.edu/vj
Developed at the University of Washington, the Vocal Joystick (VJ) is software that will allow people with physical disabilities to use computers more freely. It maps vowel sounds to spatial directions, allowing people to interact with a computer or mechanical device such as a robotic arm using their voices. The VJ differs from traditional speech-recognition software because it uses volume, pitch and vowel quality to create continuous movement, similar to how one's hand controls a computer mouse.

AUDIODYSSEY
http://gambit.mit.edu/loadgame/audiodyssey.php
From Singapore-Gambit MIT Game Lab, AudiOdyssey is a music/rhythm game for Windows that can be played by both sighted and visually impaired gamers. While it has a basic and stylish graphical user interface, no visual information is needed to play the game. As a dance club DJ, the player uses the arrow keys or a Nintendo Wii remote to copy beats according to their rhythm and location in aural space. Copying them correctly lays down tracks, one on top of the other. The more tracks are laid, the more challenging and musically rich the game becomes. Just because a game is accessible doesn't mean it's easy.

These projects are just a sampling of what's being done to make games more accessible to players with disabilities. For more ideas, resources, and disability-friendly games, visit OneSwitch (www.oneswitch.org.uk) and 7128 Software (www.7128.com), as well as news/review sites AbleGamers (www.ablegamers.com) and Assistive Gaming (http://assistivegaming.com).

—Tera Kirk

RESEARCH: iPHONE DEVELOPMENT TIME INCREASING, PRICES DECREASING

GAME DEVELOPER RESEARCH HAS REVEALED select results of its first ever State of iPhone Game Development report, helping to illuminate the growing but still largely undocumented iPhone and iPod Touch game software market.

While iPhone game development is still in its relative infancy, evidence suggests projects are becoming more ambitious and studios are becoming more sophisticated in their approaches.

Select report metrics have revealed a trend toward longer development cycles for iPhone games currently in production.

For example, the number of games with 1–3 month development cycles among survey respondents dropped from 61 percent of completed projects to 41 percent of in-development projects, whereas the number of games with 4–6 month development cycles rose from 25 percent (completed) to 47 percent (in development).

The survey also asked developers whether they adjusted prices post-release, by how much, and why.

Most developers (56 percent) in the survey which had released iPhone games claimed to have adjusted the price of their game at some point following its launch. Of those, 69 percent said they reduced the price, while the remainder said they increased it.

Even though most developers did adjust the price of their game, zero adjustment was still the most common individual response. The most common actual price adjustments were –$2 (19 percent) and –$1 (14 percent), followed by +$1 (9 percent).

The full 39-page report, available for purchase at www.gamedevresearch.com, uses iPhone App Store data and an anonymous survey of almost 150 current iPhone developers to outline a number of important data points. These include purchase prices for iPhone games, the effects of and reasons for post-launch price adjustments on games, download number ranges, game porting trends, and many other vital stats.

—Chris Remo

HANSOFT

Download a free 2-user solution trial at www.hansoft.se

>> Today, game development teams in 20 countries are using Hansoft to leverage productivity and ship better games faster. This started when we revolutionised scheduling to be collaborative and made Agile scalable to the size and specifics of game development. More and more teams are also benefiting from using Hansoft for bug tracking. Fully integrated, easily customised, and with free QA accounts to help you bring Quality Assurance closer to development. Now, with Hansoft 5.3, art pipelines can be created in parallel with sign-off workflows to make asset creation more lean. <<

2009 Game Developers Conference Preview

THIS SHOULD PROVE TO BE A MOST interesting Game Developers Conference. Given the large number of recent layoffs, one presumes there will be a very crowded career pavilion, and an increased emphasis on the networking aspect of the show.

Perhaps now more than ever, it will be a time to pick up new techniques, as more folks don't have their nose to the grindstone on a specific project, which leads them open to experimentation. As such it would stand to reason that programming and visual arts talks on new subjects or techniques would be particularly worth attending this year.

The GDC begins as it often has in recent years, with two days of summits. The difference this year is that one summit pass now allows access to the entire suite of programs, so an iPhone developer with a casual interest in virtual worlds can attend a talk without having to hop the fence. See page 12 for more specifics about the summits as well as the official networking events that surround the main conference.

There are two keynotes this year, but we can only share one of them with you now. Satoru Iwata, president and CEO of Nintendo, will make his triumphant return to the GDC center stage, giving a talk entitled "Discovering New Development Opportunities." Vague though the title may be, Iwata comes from a development background, often touting his time at HAL Laboratory, and always has interesting or inspirational advice for the folks in the trenches. A second keynote is under wraps for the time being, but it's a pretty big name, so you likely won't be too disappointed.

In the following pages you will find editors' picks from the six full-time staff members of Game Developer magazine, Gamasutra.com, and Gamecareerguide.com (all owned by Think Services, which runs the GDC), comprising 30 talks, but keep in mind that more will have been added after press time, so there will still be plenty of gems to discover.

This is the time to bring your best stuff to the table. In a time of upheaval there is immense opportunity, if you look for it, or even better if you forge it yourself. With so many talented people no longer employed full time, this may be a good year to go indie, or to start a new studio, or even to consider shifting fields.

There is great potential in this industry right now, so attend some talks, meet some new people, and see about bending your corner of the game industry into what you always wanted it to be. See you at GDC!

—Brandon Sheffield

Editor-in-Chief **Brandon Sheffield**'s picks

THE ART OF BRAID (LECTURE)
David Hellman (freelance)
BRAID was David Hellman's first commercial game project—and what a way to start out! As an accomplished artist (try his old webcomic at www.alessonislearned.com) on a very small team, Hellman was able to make mood, tone, and style his primary concerns. In this talk, he discusses the rationale behind his stylistic choices, and the emotions he hoped to instill. Rare is the opportunity to control the entire mise-en-scène of a game, so this might be a nice chance to live vicariously.

EVERYTHING I LEARNED ABOUT LEVEL DESIGN I LEARNED FROM DISNEYLAND (LECTURE)
Scott Rogers (THQ)
A number of designers have mentioned to me how much Disneyland's layout can teach you about design, and this should prove to be an engaging talk on the subject. The theme park, from its impressive initial gating, to the way it entertains people waiting in line, to its long, medium, and short-term visual goals, to its intuitive layout and cohesive look, can teach us a lot about how to make a player work his or her way through a new world. Scott Rogers, a designer for GOD OF WAR and MAXIMO, shall show us the way.

LIGHTING WITH PURPOSE (LECTURE)
Jay Riddle (Disney Interactive)
Lighting makes a huge difference in games, more than even some game artists realize. A film background really helps in this case, and that's just what Jay Riddle's got, with films like *Terminator 2*, *Starship Troopers*, and *The Abyss* under his belt (plus a degree from the USC film school, just like me!), on top of his several years as a game art director. Here, Riddle will decode the process and aesthetics of lighting, showing (with examples) how artists can increase the intuitiveness, immersion, and overall look of a game.

BUILDING YOUR AIRPLANE WHILE FLYING: PRODUCTION AT BUNGIE (LECTURE)
Allen Murray (Bungie)
Bungie went from a zero producer studio to having a full production staff over the course of HALOs 1 through 3. In what could double as a studio postmortem, Bungie producer Allen Murray will discuss how the company integrated producers into a creative-led studio without breaking everyone's work cycles. Murray promises to be "brutally honest," which is always a good thing if true.

HITTING 60HZ WITH THE UNREAL ENGINE: INSIDE THE TECH OF MORTAL KOMBAT VS DC UNIVERSE (LECTURE)
Jonathan Greenberg (Midway)
It's no secret that Midway has had a rocky path to integrating the Unreal Engine into its pipeline—but that just makes for more educational success stories, when they come. Greenberg, programming lead on the MORTAL KOMBAT franchise, shows here how Midway Chicago got a fighting game running at 60Hz in Unreal Engine 3 on both the Xbox 360 and PlayStation 3. He's using MORTAL KOMBAT as the framing for this talk, but will also discuss the more general case, to prove he's not just bragging!

Publisher **Simon Carless**' picks

LIONHEAD EXPERIMENTS REVEALED (LECTURE)
Peter Molyneux (Lionhead)
Peter Molyneux—let's face it, he's a modern svengali, the god of god games, and a startlingly persuasive speaker. And given that FABLE II is one of the best games of last year, and that Molyneux seems to have focused back on design and away from multiple project-juggling, his "design experiments" should be very interesting to behold.

GAME DEVELOPMENT 2.0: MOVING FROM A BOXED PRODUCT TO A SERVICE-BASED MODEL (LECTURE)
Justin Quimby (Maxis)
The folks at Maxis shipped SPORE in a retail box, sure, but they certainly know a lot about online connectivity, thanks to the massive success of CREATURE CREATOR and the uniquely linked way in which creations access other users' worlds in the games. And it looks like they have expansions galore up their sleeves, too. So Quimby, having worked on both MMOs and the linked goodness of SPORE, is well-positioned to tell you how to schedule and organize things to keep your games flowing.

REAL-TIME DEFORMATION AND FRACTURE—FINITE ELEMENT SIMULATION AND ITS USE IN STAR WARS: THE FORCE UNLEASHED (LECTURE)
James O'Brien (University of California, Berkeley), Eric Parker (Pixelux)
THE FORCE UNLEASHED has some really interesting destruction effects—that much is known. But how did they actually carry out some of the neat splintering and destruction you'll see when you use your force powers in the game? Turns out we may find out in this postmortem, which should be particularly notable because it follows some similar lectures given before the game shipped—so they may be able to tell you how actuality compared to theory.

GDC MICROTALKS—ONE HOUR, TEN SPEAKERS, UNLIMITED IDEAS (LECTURE)
Richard Lemarchand (Naughty Dog), Robin Hunicke (Electronic Arts), Eric Zimmerman (Gamelab), N'Gai Croal (Newsweek), Frank Lantz (area/code), Jenova Chen (thatgamecompany), Tracy Fullerton (University of Southern California), John Sharp (Savannah College of Art and Design-Atlanta)
Many of today's tech conferences split out their missives into tiny, bite-sized chunks, and it turns out that this particular GDC lecture does the same in microcosm, with some pretty darn interesting people, including Naughty Dog's Richard Lemarchand and Area\Code's Frank Lantz, blasting wisdom at you in carefully crafted five minute chunks. Don't blink, you might miss it.

PUNCH THEM IN THE STOMACH IN THE FIRST 5 MINUTES! (LECTURE)
Cory Barlog, Eric Williams
Barlog and Williams are some of the key design talents behind the GOD OF WAR series, which, let's face it, got the all-out action-fest exactly right. So it'll be great to see some of the underpinnings behind the franchise and how it gets you involved. I'm hoping that Barlog will also spill about some of his plans for the *Mad Max* game that he's been working on with its creator George Miller.

Production Editor **Jeffrey Fleming**'s picks

THE ITERATIVE LEVEL DESIGN PROCESS OF BIOWARE'S MASS EFFECT 2 (LECTURE)
Corey Andruko (BioWare), Dusty Everman (BioWare)
The devil is in the details and a game's level design is often subjected to an endless succession of tweaks and reconfigurations. Could there be a formal methodology for reaching the optimal level design with a minimum of wasted effort? Sounds a bit Zen. Here the MASS EFFECT team will share with us the lessons learned from level design work on the first title, and how those principles are being applied to MASS EFFECT 2.

HOW HIGH DYNAMIC RANGE AUDIO MAKES BATTLEFIELD: BAD COMPANY GO BOOM (LECTURE)
Anders Clerwall (EA – DICE)
"Audio dynamics" refers to the range of volume over which the ear can distinguish individual sounds, from the softest to the loudest. While our expensive home theatre rigs are capable of reproducing sound with great articulation and subtlety, most video game sound design is still rooted in the "This one goes to eleven" philosophy. DICE took a different approach with BATTLEFIELD: BAD COMPANY, striving to immerse players in an aural landscape that included everything from the quiet rustle of grass against canvas to the hammer blow of fuel-air explosives.

CAMERA BASED GAMING: THE NEXT GENERATION (LECTURE)
Diarmid Campbell (Sony)
I've had my doubts about camera-based gaming, but where the Wii has its Wii remote, Sony has long had its EyeToy. I'm curious to see where the technology is going, and Diarmid Campbell from Sony's EyeToy Technology group will be giving us a look under the hood as well as showing off some techniques used in the upcoming EYEPET game.

METAL GEAR SOLID SERIES AUDIO POSTMORTEM (LECTURE)
Norihiko Hibino (GEM Impact)
With METAL GEAR SOLID 4's release date only six months away Kojima Productions realized that it still needed over 90 minutes of cut-

scene music to be composed and recorded. With hard deadlines fast approaching, the developers turned to Konami veteran Norihiko Hibino, who left Konami in 2005 to form GEM Impact, to complete the game's signature sound. Amazingly, his team was able to complete the work in only three weeks. In this lecture Hibino will describe how he approached the large-scale project on such an accelerated schedule.

BEND MICROSOFT PROJECT TO YOUR WILL—AGAIN (LECTURE)
Mike McShaffry (Freelance)
Microsoft Project has been in use for almost a quarter of a century, making it a popular choice for project management. The software's broad userbase practically ensures its use on most projects. Unfortunately, the vagaries of game development can make Project a poor fit, and producers often struggle to get useful results out of it. In this lecture, McShaffry will share some of his hard-won Project experience, gathered over many years in the trenches of game development working on games from ULTIMA VII to MUSHROOM MEN: THE SPORE WARS.

Senior Contributing Editor **Jill Duffy**'s picks

ADVANCED SCRUM AND AGILE DEVELOPMENT (LECTURE)
Clinton Keith (Clinton Keith Consulting)
Clinton Keith is the game industry's Scrum and agile development guru, the pilgrim who brought his religion to the New World as developers looked to their Manifest Destiny of conquering multi-threading, advanced A.I., HLSL, and beyond. Keith, who has spent the last six years preaching Scrum, has a knack for explaining how to not only adopt the methodology, but also convert a whole studio. The clincher is knowing when to put an ounce of blind faith in the process and when not to.

BUILDING A BUG-FREE TEAM: SUCCESSFULLY MANAGING Q/A TEAMS (LECTURE)
Jane Fraser (Electronic Arts)
As someone whose coverage of the game industry now focuses tightly on careers, I'm a big fan of GDC lectures that inspire people to not just do their current jobs better, but move up in the ladder, too. In addition to what the title of this talk promises, I have a feeling Jane Fraser will motivate a few attendees to become stewards of their teams and studios.

EARLY STAGE FUNDING FOR VIDEO GAME START UPS (LECTURE)
Matthew Le Merle (Gameplay Holdings and Keiretsu Forum)
If you're thinking of starting your own game development operation, it's going to take capital, and in this economic crisis, that second mortgage on your San Diego home ain't gonna cut it. So what the heck does this Matthew Le Merle guy think he knows about playing the VC funding game? You'll just have to come to this session to find out. (And that's how they get you.)

STRETCHING BEYOND ENTERTAINMENT: THE ROLE OF GAMES IN PERSONAL AND SOCIAL CHANGE (PANEL)
Rusel DeMaria (DeMaria Studio), Ed Fries (FigurePrints), Bing Gordon (Electronic Arts), Will Wright (Maxis), Lorne Lanning ("odd" fellow), Peter Molyneux (Lionhead)
Remember when television was going to be one of the great equalizers, bringing education right into people's homes and providing access to all? (Of course you don't. That was the 1940s.) Along the same lines, we still don't really know where games and interactive entertainment stand in terms of affecting social bounds. In this granddaddy of the design track, a lineup of game industry kings have agreed to share the secrets to the game industry universe—or at least what they think about how games influence players' lives and the public.

AARF! ARF ARF ARF: TALKING TO THE PLAYER WITH BARKS (LECTURE)
Patrick Redding (Ubisoft Montreal)
In a galloping 20-minute lecture, Patrick Redding promises to explain to game designers, sound designers, producers, and perhaps a wayward programmer or two, how to approach AI dialogue for NPCs and dialogue systems in a way that keeps your characters from sounding like those retarded robots at Chuck E. Cheese's. The arfing and barking in the title refers to dialogue "barks," or audio cues that tell the player something in the gameplay conditions has changed.

Gamasutra Features Director **Christian Nutt**'s picks

EXPERIENCES AND RARE INSIGHTS INTO THE VIDEO GAME MUSIC INDUSTRY (LECTURE)
Hitoshi Sakimoto (Basiscape International)
Sakimoto is hardly a household name, but is in fact one of the most talented composers working in game music today. Most notably, he was the composer for FINAL FANTASY XII, which required a film-like flair for drama while maintaining dungeon tunes that can underscore endless exploration. His soundtracks maintain an excellent balance—they buoy gameplay without eclipsing it, which is a tough act to pull off.

PRACTICAL SPU USAGE IN GOD OF WAR 3 (LECTURE)
Jim Tilander, Vassily Fillipov (Sony)
GOD OF WAR 3 has to deliver intense, beautiful, and visceral combat to the PlayStation 3. Not disappointing the series' massive fan base is a top priority. Getting in on the ground floor on this tech talk for an in-development game is an absolute must. Coming from a first party studio, these guys are bound to have all the most effective, up-to-date knowledge at their disposal.

DROP-IN CO-OP FOR OPEN WORLD GAMES (LECTURE)
Glenn Fiedler (Sony)
Fiedler was the multiplayer lead programmer on Pandemic/EA's MERCENARIES 2, which featured a

streaming open world and drop-in, drop-out co-op throughout its entire campaign—one of the things critics and fans liked best about the game. Making that work requires a huge amount of technical knowledge and skill. When I consider the great times I had with another co-op open worlder, Realtime Worlds/Microsoft's CRACKDOWN, which offered a similarly free-form co-op experience, I'm convinced more developers need to emphasize this in open-world games. Here's where to start.

FAULT TOLERANCE: FROM INTENTIONALITY TO IMPROVISATION (LECTURE)
Clint Hocking (Ubisoft)
FAR CRY 2 may not have worked 100% of the time, but when it did, it offered glimpses into what the future of open-world gaming might be like. Trust me, Gamasutra editor-at-large Chris Remo won't stop talking about the damn game, so I know. In this must-see talk, Hocking will relay his ideas, no doubt refined by a long and complicated development process, on making games that players can simply be liberated to play the way they want to—an excellent goal.

STAR OCEAN 4: FLEXIBLE SHADER MANAGEMENT AND POST-PROCESSING (LECTURE)
Yoshiharu Gotanda (tri-Ace)
Tri-Ace isn't a very well-known developer in the West, but among a select audience, they're synonymous with the best visual quality in the field. Here, the developers will speak on the development of sophisticated film-like post-processing techniques, including advanced depth of field, film and shutter simulation, and correctly simulating camera lenses, including focal length and aperture simulation. This is hardcore stuff, but when you see the stunning visuals of STAR OCEAN: THE LAST HOPE, which ships before GDC, you'll agree that the effort paid off.

Gamasutra Editor-at-Large **Chris Remo**'s picks

CRYSIS WARHEAD POSTMORTEM: BLOWING UP STUFF (LECTURE)
Bernd Diemer (Crytek)
Crytek followed up on 2007's ambitious CRYSIS with the pseudo-sequel CRYSIS WARHEAD, sharpening the original game into a more focused, action-packed experience that demonstrates a studio more comfortable with the design framework it set up with the series' debut. WARHEAD actually started out as a more traditional expansion before it morphed into its parallel narrative form, giving recently-created subsidiary Crytek Budapest a chance to flex its muscles.

DESTRUCTION OF DESIGN (LECTURE)
Luke Schneider (Volition)
A major legacy of Volition's RED FACTION series is its environments' destructibility, and with its upcoming RED FACTION: GUERRILLA the studio plans to keep pushing that envelope. But destructibility introduces all sorts of design and development issues that have traditionally kept it from becoming an integral part of games. Here, we'll find out if Volition addressed those issues. (Series fans, take note: the company has pledged it is following in the footsteps of the original RED FACTION, and not its less impactful sequel.)

PLAYER'S EXPRESSION: THE LEVEL DESIGN STRUCTURE BEHIND FAR CRY 2 AND BEYOND? (LECTURE)
Jonathan Morin (Ubisoft)
Designers frequently talk about how a video game's real story is the one the player tells while playing—but most gamers know that those emergent special stories don't always happen as frequently as they're supposed to. How can the likelihood of meaningful player expression be increased through level design? FAR CRY 2's lead level designer attempts to answer that question, not just as it relates to his own game, but in a broader design context.

HALO WARS: THE TERRAIN OF NEXT-GEN (LECTURE)
Colt McAnlis (Microsoft Ensemble Studios)
Screenshot and video marketing for real-time strategy titles tends to focus on how detailed the little men are when you push the camera all the way up to its maximum zoom (a level at which nobody will actually play the game), but Ensemble Studios knows that the terrain fidelity is important too. Here, HALO WARS' graphics programmer explains how the studio has moved from a height-field terrain system to a vector-field terrain system and multiplied its vector density by a factor of eight.

SPORE: FULFILLING THE MASSIVELY SINGLE-PLAYER PROMISE - HOW'D WE DO? (LECTURE)
Caryl Shaw (Maxis)
In Maxis' ambitious SPORE, the volume and range of user-generated content seeded throughout the game's servers ended up surprising even the game's developers, forcing them to go back and adjust the systems themselves. Here, SPORE's pollinated content producer appraises the experience so far and shares some lessons learned about community integration.

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 won Game Developer Magazine's Best Engine Front Line Award for three consecutive years, and it was inducted into the Hall of Fame this year.

Epic's internally developed titles include the 2006 Game of the Year "Gears of War" for Xbox 360 and PC; "Unreal Tournament 3" for PC, PlayStation 3 and Xbox 360; and "Gears of War 2" for Xbox 360.

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.

UNREAL ENGINE 3 DRIVES *WHEELMAN*

Vin Diesel's next adventure won't be seen on the big screen. The action star is appearing as Milo Burick, an undercover cop out to infiltrate the gangs of Barcelona in *Wheelman*.

Midway Games' collaboration with Diesel's Tigon Studios is being fueled by Unreal Engine 3 technology.

Shaun Himmerick, executive producer of *Wheelman* at Midway Newcastle, said that it's been great utilizing UE3 for the action driving genre. *Wheelman* is part of a cross-company shared technology endeavor that Midway employs in each of its global studios.

"With an open world game, there is both driving and on-foot action, so we borrowed designers from the *Stranglehold* team to help us," added Himmerick. "While it's a challenge to put every studio on a common engine, it makes it easier to share the technology."

Wheelman's virtual Barcelona is a massive city with over 4,000 miles of roads, including back alleys that can only be navigated by motorcycle.

One of the hurdles the *Wheelman* team tackled was streaming the virtual city on console. Himmerick said that UE3 was built to stream on-foot action, which might cover 10 feet per second, but his team needed the engine to accelerate up to 150 miles per hour for speeding vehicles.

"To build an engine that can stream the equivalent of a first-person shooter map every eight seconds was a massive challenge," said Himmerick. "We used a lot of core Unreal technology, and we also added a lot to it like Kynapse and the Havok engine."

In addition, the team modified toolsets within UE3 like Matinee, which was revised and referred to as the cinema tool by the team. That technology was originally designed for *Stranglehold*, another UE3 game, but was customized for *Wheelman*.

At its base, Midway used the stock Matinee engine and then added bolts onto it. The end result provided specific functionality that enabled the team's cinema group to precisely manipulate the cameras within the game world for the desired cinematic feel.

"We have something called super mesh, which allows for more variation of character than the Unreal Engine normally lets you do," continued Himmerick. "For us, it enables us to do more damage to cars. Super mesh doesn't work out of the box with Matinee, so we had to hijack the in-game car and pull it into our cinema engine to ensure a clean transition."

One innovative idea that came from conversations with Vin Deisel was to create car combat in the vein of a fighting game. Instead of bumping into another car over and over again, the game employs melee combat like punches that can knock out enemies immediately and dramatically.

Wheelman for Xbox 360, PlayStation 3 and PC

Wheelman also has a super move that allows the driver to spin 180 degrees, blast his enemies with guns, and then spin back around. Himmerick said all of the action is captured with very cinematic camera perspectives.

The open world game lets players exit vehicles and engage in gunfights, combat and exploration as well. Cars will come in handy as shields from bullets, but well-timed shots can also blow up vehicles and take out multiple enemies.

"Unreal has given us a great starting point," explained Himmerick. "The file structure, the way it's organized, and the fact that it was all done with the infrastructure that we were familiar with has allowed us to easily modify what we need, like the cinema tool, and then share it among all of the studios."

With fast and furious action, *Wheelman* will allow anyone to step into the starring role as Vin Diesel and take down the bad guys by any means necessary.

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

For UE3 licensing inquiries email:
licensing@epicgames.com

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

W W W . E P I C G A M E S . C O M

Week in Review Calendar

SUMMITS	MON.–TUE., MARCH 23-24	9AM–6PM
TUTORIALS	MON.–TUE., MARCH 23-24	10AM–6PM
GDC WORLD MIXER	TUE., MARCH 24	
IGF MOBILE AWARDS	TUE., MARCH 24	
GAME DEVELOPERS CHOICE AWARDS AND IGF AWARDS	WED., MARCH 25	6:30–8:30PM
EXPO	WED.–THURS., MARCH 25–26	9AM–6PM
	FRI., MARCH 27	9AM–3PM
GAME CAREER SEMINAR	FRI., MARCH 27	9AM-5PM

Summits

The first two days of GDC are given over to focused summits (and tutorials) that bring together developers with shared interests. To encourage the spread of knowledge across disciplines, the Summits and Tutorials Pass grants access to all summits and tutorials.

New to GDC this year is the **AI Summit**, a series of lectures and panels that will pull back the curtain on artificial intelligence. While game design and AI development have always been two sides of the same coin, recent advances in hardware capability, artificial intelligence theory, and AI middleware have brought a new importance to the field.

Casual Games as a category may soon become a misnomer. With the success of Nintendo's Wii and DS platforms, the total ubiquity of cell phones, thriving PC portals, and the rise of quick playing downloadable games on consoles, "casual" is looking more and more like "mainstream." This year's **Casual Games Summit** has the business side of things well covered along with what are sure to be some interesting design insights from Steve Meretzky.

The business of mobile gaming has long been held back from its full potential by the inherit difficulties in dealing with large telecom providers and a target platform that is continually in flux. With the arrival of Apple's iPhone SDK and App Store business model along with open platforms like Google Android you can bet that this year's **GDC Mobile Summit** will be generating real heat.

The idea of outsourcing conjures up images of cheap overseas labor sucking away jobs but the reality is that many successful domestic studios make their bread performing technical work for other studios. With that understanding, the question becomes not one of eliminating outsourcing but rather one of how to effectively integrate it into the development process. The **Game Outsourcing Summit** should be a good opportunity to share methods and experiences in what is quickly becoming an economic fact of life.

Although focused on resources for educators, the **IGDA Education Summit** has much to offer the development community at large including student IGF postmortems and a keynote by Front Line award winner Jesse Schell.

As large studios downsize and sometimes shutter completely, ronin developers are taking a look at the indie life and wondering if there is place for them among the digital bohemians. The **Independent Games Summit** will be exploring the practicalities of going independent, from design to business with the question yet to be answered: can independent games' economic potential match their cultural cachet?

Also new this year is the **Localization Summit**. It's never been truer that video games are a global industry and developers have to think of their games as having a worldwide audience. Of course successfully crossing language and culture lines requires a talented localization staff and careful planning from the very start of a project.

Serious games have been around far longer than video games. One of the earliest was *Kriegsspiel*, whose design was finalized in 1824 to help train Prussian army officers. It could be argued that much of the complexity of modern games is derived from this long heritage of sims from the pre-video game era. The **Serious Games Summit** promises to be a unique mix of academic, corporate, and scientific voices, all with goal of moving games out of the living room and into the real world.

The **Worlds in Motion Summit** is in its second year at GDC and features a broad line up of speakers who will be discussing the burgeoning market for online games. Online games present an interesting challenge to the industry in that their business model and game design are inexorably linked together. Art and commerce are no longer at odds and both sides are pushing the field in new creative directions.

—Jeffrey Fleming

TRINIGY
www.trinigy.net

"A perfect fit for us."

Dietmar Meschede, Head of Programming at UBISOFT® Studio BlueByte

Vision Engine 7 - leading technology for leading platforms

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

www.trinigy.net

SECRET LEVEL'S Golden Axe: Beast Rider

WHAT DO GAME DEVELOPERS DREAM? DO THEY DREAM of seeing their name in lights? Do they dream of a boy or girl picking up a controller and playing their game for hours on end, eyes glued to the monitor? Do they dream of starting a studio of their own, or growing a team into a triple-A studio?

Different developers dream of different things. It would be stereotypical to say that all developers not currently working on big games hope to one day count themselves among the elite. Not everyone needs to work on the financial and critical blockbusters such as GRAND THEFT AUTO or HALO. Not all developers have the same career goals, nor does every development studio plan to become the next Bungie or BioWare.

Nevertheless, opportunities for major studio growth and expansion do come along every once in a while in this industry. Sometimes the chance to make a big-budget title is given to a fledgling studio. When an opportunity like that presents itself, will your studio's leadership make the jump and have the guts and savvy to survive?

This postmortem is the story of how our studio, San Francisco-based Secret Level, both succeeded and failed in making the transition from a small work-for-hire developer to a major first-party development studio, and how SEGA's GOLDEN AXE: BEAST RIDER and IRON MAN—which cross-pollinated BEAST RIDER with shared teams and technology—both ultimately shaped and were shaped by the studio's transition.

THE REVENGE OF DEATH ADDER

In July 2005, SEGA of America, Inc., announced the revival of an as-yet-unnamed vintage IP, helmed by developer Secret Level. At the time,

MICHAEL BOCCIERI *is a producer at SEGA Studios, San Francisco. Email him at* ***mboccieri@gdmag.com****.*

the studio was best known for its adept Xbox ports of JEDI STARFIGHTER and FINAL FIGHT: STREETWISE, as well as development of Xbox's MAGIC: THE GATHERING and AMERICA'S ARMY: RISE OF A SOLDIER.

In less than a year, SEGA announced its acquisition of Secret Level as part of a larger effort to grow its internal first-party development worldwide. Simon Jeffery, president of SEGA of America said of the acquisition, "We looked long and hard at building an internal studio from scratch, but were so impressed with the team at Secret Level and their next-gen technology that we decided to create our internal development infrastructure through a direct acquisition, one that could fulfill our dynamic growth plans and produce high quality games."

Shortly thereafter it was announced that the game was a new version of GOLDEN AXE, a fan favorite since its original 1989 arcade and SEGA Genesis release. At the outset, the internal goals of the development team were lofty: to create an epic, GRAND THEFT AUTO-inspired experience set in the world of GOLDEN AXE. Players would battle and commandeer massive war beasts and travel over the vast, streaming landscapes of Yuria in a fully immersive and persistent world.

At the time of the SEGA acquisition, the core GOLDEN AXE team within Secret Level comprised only five full-time employees. Suddenly, studio expansion became a pressing need (and a goal of SEGA's overall studio strategy), and within two years the studio as a whole ballooned to nearly 150 employees. Studio staff was not only dedicated to developing the newly minted GOLDEN AXE: BEAST RIDER, but also a second title, a movie tie-in of *Iron Man*.

In October, 2008, nearly four years after the project's inception, after numerous stops and starts, SEGA released GOLDEN AXE: BEAST RIDER (IRON MAN released earlier that year, in time for the spring movie release). Gone was the massive scope of the game's initial vision. In its place was a more austere, challenge-oriented game with a core that harmonized more closely with the original franchise than with the breadth-of-features experience found in the year's strong batch of fall titles. After years of development and a lot of money spent, critical reception of both games was poor.

How did this happen? Despite strong sales and favorable user reviews for both titles, the final outcome of this story seems bleak. However, for those of us in the trenches (and for our unwavering supporters at SEGA) the story of GOLDEN AXE: BEAST RIDER is also a very positive one.

It is an underdog story of how late-term efficiency and teamwork, forged alliances, and practical decision-making ultimately allowed this studio to prevail over a multitude of costly start-up mistakes. Although the stakes were high, with cancellation and studio closure potentially on the line, GOLDEN AXE: BEAST RIDER and Secret Level beat the odds and lived to fight on.

WHAT WENT WRONG

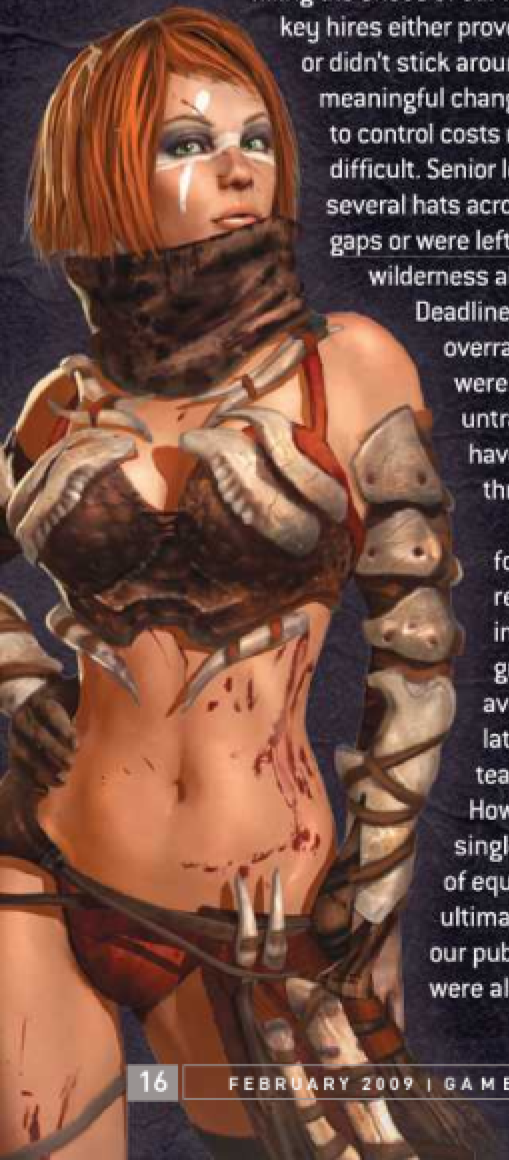
1) FILLING KEY DEVELOPMENT ROLES LATE. The single greatest detriment to the studio's ability to create a critically successful GOLDEN AXE release was failure to adhere to the adage "the right tool for the right job" when it came to hiring and retaining key personnel. While the studio's acquisition by SEGA was fueled in no small part by the quality of the talent in key positions, as soon as the ink on the deal was dry, issues began to arise as those same personnel moved out of implementation and into studio management roles.

This change didn't happen overnight, but the gradual rebalancing of skill sets from production to management at the start of development led to a widening gap in the studio's ability to deliver on publisher expectations. What followed was a long period of developer draught, plagued by a glut of serious issues.

Inexperienced team members floundered in their new leadership positions, trying to create order from chaos and filling the shoes of our rock star founders. Initial key hires either proved themselves ineffective or didn't stick around long enough to affect meaningful change. Layoffs and hiring freezes to control costs made retention and growth difficult. Senior leadership wound up wearing several hats across disciplines to fill the gaps or were left to wander the developer wilderness alone as solo implementers. Deadlines began to slip, costs overran, and milestone goals were increasingly left unmet or untracked. GOLDEN AXE did not have a vertical slice in the first three years of its development.

It was the decision to focus greater efforts and resources on attracting top industry talent, and team growth beyond a 15-person average, that saw an eventual late-term turnaround in the team and studio's fortunes. However, as a result of this single overarching issue, a series of equally grave problems arose, ultimately costing the studio and our publisher the quality product we were all ready to deliver.

CONTINUED ON PG 18



The Game is About to Change

Introducing Unity 2.5 for Windows and Mac

COMING SOON
VISIT OUR WEB SITE FOR MORE INFO
UNITY3D.COM

unity

CONTINUED FROM PG 16

2) LOSING FOCUS ON THE PROPERTY'S CORE. The loss of a cooperative multiplayer experience from BEAST RIDER did not happen overnight. At the project's outset GOLDEN AXE was designed to be a cooperative experience. However, the depth to which a cooperative experience was scoped and scheduled was far short of what was necessary to turn planning into reality.

As the early years of development dragged on, little work was done to focus co-op efforts. While some development occurred in this area, a series of unfortunate events ultimately led to the feature's demise. A poorly envisioned multiplayer design, key losses on the network technology front, and a lack of animation staff and support led to a general freeze on its implementation.

Rather than re-scope other areas of development to save this important feature, the team continued to focus elsewhere: on disparate elements of the single-player experience, on complicated beast mechanics, and even on other game designs that came across the team's desks.

When experienced personnel finally took hold of the reins around early 2007, it seemed clear that the inclusion of multiplayer was completely untenable for an early 2008 release. Work was done to prototype a multiplayer battle arena in mid-2007, but the decision was made at the parent level to cut the feature when assessment proved there would be additional Q/A costs to support it. Many people on the development team and external to the studio felt that we were cutting the heart out of the GOLDEN AXE franchise, but because they agreed that the team was unlikely to deliver the feature in time for release, everyone had to shift gears and find a new product identity for the game—thus the greater focus on beasts.

In hindsight, had the team been able to anticipate the our late term productivity gains, as well as the schedule extension that was granted to complete the project and increase quality, then cooperative multiplayer would not have been cut as a core feature. While the team made a valiant effort to retain other aspects of the classic franchise within BEAST RIDER—classic locales, a decidedly retro-flavored combat system, and the return of the beasts and gnomes from the original series—these elements proved ineffectual in piquing interest from the press and the hardcore consumer base, jaded by news of co-op's omission.

3) IF YOU CAN'T MAKE ONE GAME, DON'T MAKE TWO. By 2006, GOLDEN AXE was digging itself into an increasingly deeper hole. While slow progress was being made on the engine front, gameplay prototypes remained woefully inadequate on delivering expected quality. The studio was also grossly over budget. Secret Level—and ultimately SEGA—looked around for a hero to save the studio from itself.

That hero turned out to be IRON MAN. A deal for the studio to create movie tie-in games for Xbox 360 and PlayStation 3 platforms worked out, allowing the studio to spread early

IGNORING IT WON'T MAKE IT GO AWAY

Early previews for GOLDEN AXE: BEAST RIDER were already looking bad well before the game shipped. "It's too brown," and "no co-op," were major talking points in the preview press that largely went unanswered by Secret Level and SEGA, even though plenty of supporters were willing to counter these perceptions. Rather than aligning resources towards building a more positive press image, the call was made to ask the media to delay final reviews until a week after the game was on shelves. By acquiescing to the game's perceived faults and failing to publicly support the game's positive qualities—beasts, innovative defense system, framerate—the game even lost face among its die-hard supporters. Lesson learned: if you're going to put a product on the shelves you must be ready to address the criticism that early press might dwell on—justified or no—and spend the dollars and pound the pavement to realign buyer and media expectations to the positive aspects your product brings to the consumer.

engine development between two cost centers. While this helped put the studio on better financial footing, it remained to be seen whether Secret Level would be able to deliver two games within the allotted timeframe.

That the studio was able to accomplish this feat—shipping two next-gen titles within one year of each other—is no small accomplishment, and is a testament to the level of experience and professionalism the studio was able to attract.

However, the history books will also show that neither BEAST RIDER nor IRON MAN delivered on general consumer expectation, and both were critically panned. IRON MAN suffered from early personnel neglect when studio resources were still scarce, and GOLDEN AXE was all but abandoned by the engine team in the months leading up to IRON MAN's ship date. While neither game suffered complete paralysis during these times, they did not benefit either.

In a sense IRON MAN and GOLDEN AXE were conjoined twins. One could not have survived without the other, yet neither was able to fully blossom into its own element with the other one attached.

4) TURNING A DEAF EAR TO THE END USER. When development reaches the point where you are treading water just to stay afloat, your gut instinct is to hunker down. While useful in barreling through rough patches, this can often lead to major breakdowns in client-vendor team dynamics.

The tools development teams on both IRON MAN and GOLDEN AXE were unable to do much more than tread water, leading to a process breakdown. "Successes" like our excellent Terrain Editor tool were tainted by bad prioritization: thousands of man hours of tools programming for only a few hundred hours of team use. "Failures" were far worse, with major omissions like copy, paste, and undo from the team's level scripting tool, and an underlying code base so buggy that patches for it were sometimes rejected, lest new, deadlier bugs arise in their place.

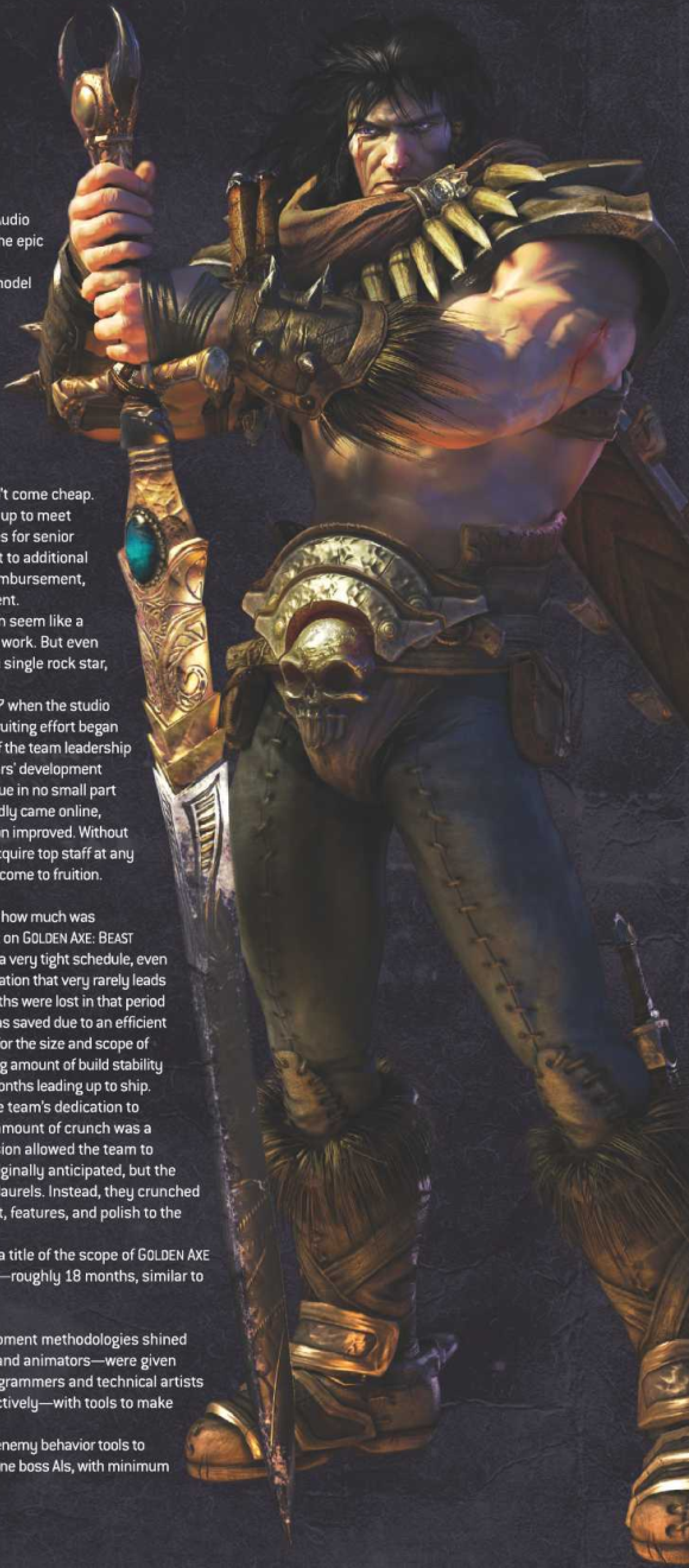
Everyone became so jaded with tool quality that new feature requests stopped surfacing. The tool developers were tired of not getting quality direction and started coding in a bubble. Both teams suffered greatly from a lack of communication and a cross-discipline dedication to improving the process.

Though it's a prime example, tools development was not the only area in which breakdowns occurred. Art and engine teams clashed on ignored feature requests. The design and production departments would butt heads on interpreting focus-test feedback. Though all these issues and more were resolved by the project's end, GOLDEN AXE: BEAST RIDER reached its darkest periods of "nightmare" status not because of crunch, but because the teams that should be constantly communicating would sometimes refuse to do so.

5) UNPREPARED FOR OUTSOURCING. A good outsourcing pipeline holds contractors accountable for delays and missed deadlines. If an external studio causes the internal schedule to slip, actionable steps can be taken. However, succeeding with an outsourcing model depends on maintaining a bulletproof internal milestone schedule. How can you expect contractors to hit development milestones if your internal team isn't either?

GOLDEN AXE dove into a production pipeline augmented with outsourced development from the start, with early concepting successes from partnerships with external studios like Gentle Giant. However, as the internal team began to lose focus on the game's overall vision, it became increasingly difficult to provide outsourcing partners with targeted direction and feedback. Collaboration with sister studios (SEGA China) in creating thousands of objects for the game was also ramping up well ahead of finalizing the game and technical design documents.

Outsourcing costs for audio, programming, and art began to overrun, eventually leading to budget lockdowns. Art assets had countless revisions when engine specs changed. Outsourced code was rejected wholesale when it didn't meet team needs. Endless contract revisions were reprocessed as the scope of work continued to balloon. Motion capture and audio sessions proved to be the worst offenders. Impending mo-cap sessions lead to hasty, untested combat designs. When these designs proved to



be uninspired, hundreds of hours of mo-cap were abandoned. Audio capture deadlines led to a rushed script that captured little of the epic tone of the original GOLDEN AXE.

While the team was able to realign its internal production model within the last 18 months of development, most outsourcing successes did not materialize until the final nine months before shipping. The right outsourcing partner is often capable of working miracles for a team when given the right tools. Secret Level learned the hard way that it is far better to delay outsourcing until you have all your ducks in a row internally, than to let these valued allies suffer at your expense.

WHAT WENT RIGHT

1) ACQUIRING INDUSTRY-LEADING TALENT. Top talent doesn't come cheap. That fact can be a bitter pill to swallow as a studio ramps up to meet the challenges of developing bigger games. Not only are salaries for senior positions significantly higher, but studios often need to commit to additional recruitment costs, too: recruiter contracts, airfare and meal reimbursement, and production time lost when meeting and assessing new talent.

Capitalizing on local hires with more junior track records can seem like a good low-cost alternative, and in some cases that strategy can work. But even a handful of junior staff can sometimes be less effective than a single rock star, especially when the chips are down.

The GOLDEN AXE team didn't really hit its stride until early 2007 when the studio finally swallowed the pill, and the first senior hires of a major recruiting effort began to trickle in. These senior hires ultimately filled out the majority of the team leadership by the time the game shipped. Compared to the previous two years' development progress, productivity from 2007 onward was vastly improved, due in no small part to the decision-making and leadership of the new staff. Tools rapidly came online, necessary scope reductions took place, and production stagnation improved. Without the presence of key personnel on the team and the decision to acquire top staff at any cost, this rapid change in BEAST RIDER's fortune would never have come to fruition.

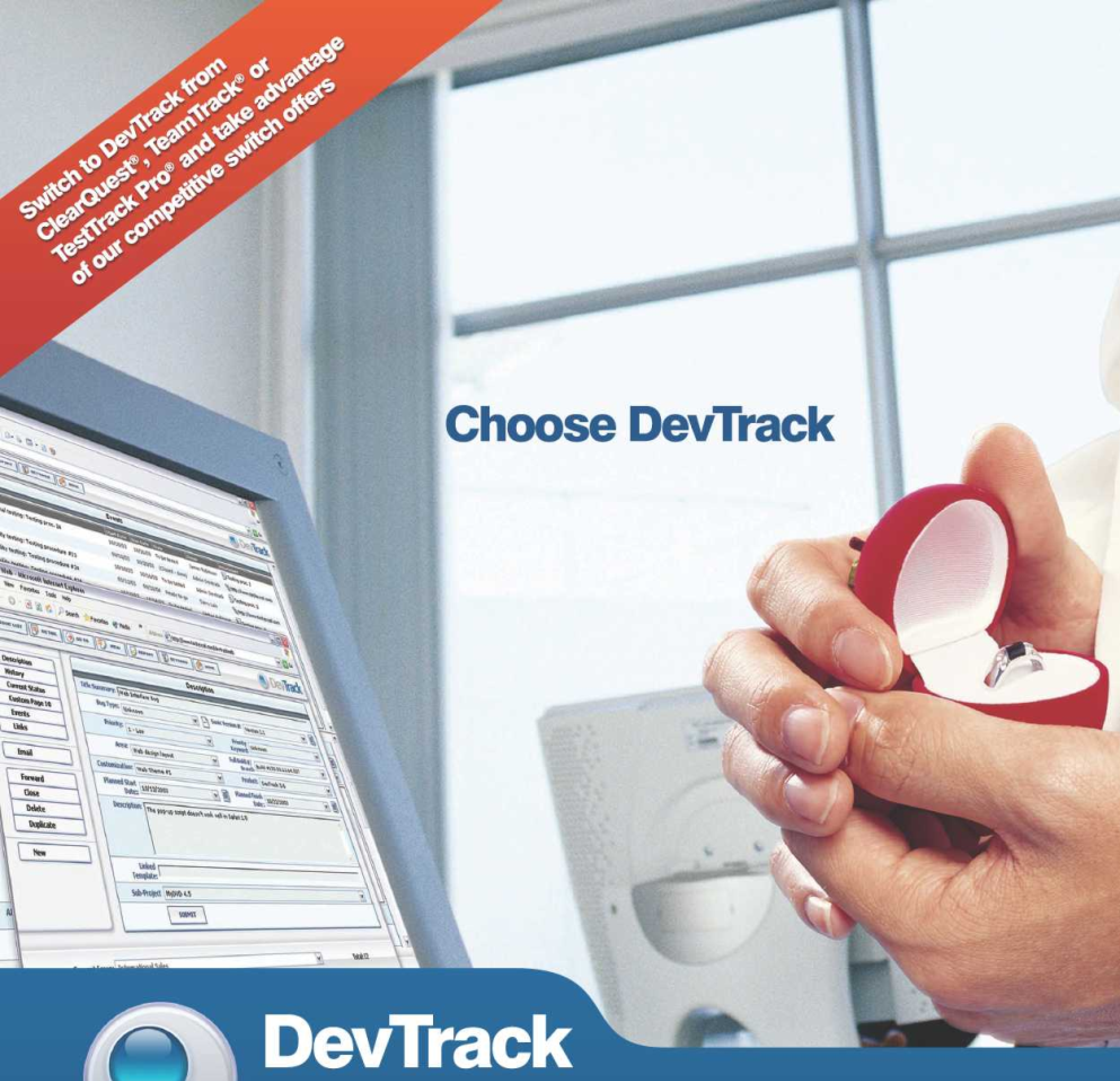
2) PERFORMING UNDER PRESSURE. I cannot stress enough how much was accomplished in the last 12 to 14 months of development on GOLDEN AXE: BEAST RIDER. The team made monumental strides in development under a very tight schedule, even as code and content were being developed simultaneously, a situation that very rarely leads to efficiency in product development. And while several man months were lost in that period due to revision and scope fluctuation, an equal amount of time was saved due to an efficient nightly build system, build monitors, and a branched code base. For the size and scope of changes being made per day on GOLDEN AXE, there was a surprising amount of build stability across disciplines, which kept teams working hard through the months leading up to ship.

While good practices lead to a modicum of sanity, it was the team's dedication to delivering the final product that ultimately won the day. Some amount of crunch was a reality for almost a full year prior to release. A six-month extension allowed the team to finish strong and deliver a much higher quality product than originally anticipated, but the extension would have been worthless if the team rested on its laurels. Instead, they crunched even harder in the final six months to bring more bonus content, features, and polish to the final product. It was a harrowing experience, to be sure.

Nevertheless, few developers can say that they developed a title of the scope of GOLDEN AXE in the time the game was truly in development with a full team—roughly 18 months, similar to some of the larger downloadable titles on the market today!

3) ENABLING CONTENT CREATORS. Our underlying development methodologies shined brightest when the end users—our designers, artists, and animators—were given robust tools to iterate. Specific credit goes to our gameplay programmers and technical artists for outfitting their end users—designers and animators, respectively—with tools to make great content on their own.

Systems designers were given a flexible and workable set of enemy behavior tools to create and script complex enemy encounters, including stand-alone boss AIs, with minimum



Switch to DevTrack from ClearQuest®, TeamTrack® or TestTrack Pro® and take advantage of our competitive switch offers

Choose DevTrack

DevTrack

The Defect Tracking Choice for Game Development

- Used by the world's largest publishers and studios
- Award-winning workflow design tool
- Integration with popular SCM systems (TFS, Subversion, Perforce, ClearCase, and more)
- Robust, web-service API
- Learn about the ways DevTrack can improve your business at **www.techexcel.com**

TechExcel

www.techexcel.com | 1-800-439-7782

programming support. The character animation, IGC, and interactive objects pipelines were also robust enough that artists could create content and get it in game on their own without too much trouble.

Although these two examples stand out in particular, enabling creative end users resulted from a bottom-up approach. It sprung from the core leadership of our tools and technology team and permeated the studio culture and harmonized with our premiere vendors, such as Havok. Although the resulting process was often mired by pitfalls, the successes it brought to development were immediately perceived and capitalized on by the studio as a whole. Future games will continue to build on this methodology with a greater focus on user-focused tools and features development.

4) OUR ENGINE, OUR WAY. Increasingly, studios are moving away from internal engine development and going with off-the-shelf solutions like Unreal and Source. GOLDEN AXE: BEAST RIDER's engine was developed internally, with our own methodologies for content and code creation.

While this approach has its share of challenges, it also brings tremendous benefits. During early development of GOLDEN AXE, we were able to deliver on some initial ideas for beast riding—giving the player a sense of speed, for example—that an off-the-shelf console engine could not have readily provided. GOLDEN AXE: BEAST RIDER also runs at a consistent 60fps on the Xbox 360, allowing us to deliver fast combat and smooth visuals. There are only a handful of current-gen games that can boast this feature, with the majority favoring a locked 30fps approach.

While we ran into some pitfalls when coding our own engine from scratch, at its core the final engine code is well crafted and capable of future expansion and modification as we see fit. Our use of LUA within the gameplay engine has also proven to be a boon for our programmers and scripters, who are able to rapidly prototype features, compared to working directly in C++. It is a testament to the efficiency of our customized engine and the benefits of "rolling your own" that the final game has so much runtime LUA, yet still hits 60fps.

5) CLUTCH PARTNERSHIPS. Some of our biggest successes came from strategic partnerships with external vendors. An early success was adopting Havok Physics and Havok Behavior Tools. Havok was an invaluable development partner, with teams dedicated to supporting our efforts and rolling our feature requests into their tool releases. Not only did GOLDEN AXE and IRON MAN benefit from using Havok, but Havok was able to take away some of our best practices and use cases for evaluating future versions of its toolsets.

Another partnership success was with San Francisco-based Orange Design, which provided external development for our user interface systems, writing menu code as well as assisting with more complicated HUD elements and save game systems. The team at Orange made itself available for onsite meetings whenever we requested them, and went out of its way to work closely with our key implementers. Finally San Diego-based Pendulum provided amazing pre-rendered cinematics for the opening and closing moments of our game. The quality was top notch, and the company was able to turn around the final assets under the pressure of a very tight schedule.

With a proven pipeline in place for the integration of external support studios, we will continue to seek top-tier partnerships wherever we see the need.

TITAN RESURRECTION

In the aftermath of a rather brutal and protracted dev cycle, Secret Level has emerged as a studio much wiser for the wear, ready to tackle the development challenges that lie ahead.

The studio has recently been re-christened SEGA Studios, San Francisco, which reflects our parent organization's ongoing integration of the studio with its larger company vision and goals. The team has also undergone many internal changes to strengthen our position in the years ahead. Rest assured, we are not going anywhere anytime soon! With SEGA's full blessing and support, we are already hard at work on our next major studio project.

The hard fought battles of both IRON MAN and GOLDEN AXE (successes in their own right) have helped us forge a veteran team ready for continued improvement and growth, and with our eyes firmly set on realizing our collective dream: a triple-A title the studio can one day call its own.

GAME DATA

DEVELOPER
Secret Level, Inc.

PUBLISHER
SEGA

PLATFORMS
Xbox 360,
PlayStation 3

RELEASE DATE
October 2008

NUMBER OF IN-HOUSE DEVELOPERS
125 full-time studio staff at maximum

NUMBER OF EXTERNAL CONTRACTORS
30–50 external vendors & contractors

BUDGET
$15 million

DEVELOPMENT TIME
Approximately 4 years

TYPICAL WORKSTATION
Intel Pentium D 3.20GHz, 2–4GB RAM, 300GB HD, NVidia GeForce 7900, Win XP

SOFTWARE USED
MS Visual Studio 2005, Autodesk Maya 7, 3ds Max 9, MotionBuilder; Zbrush; Adobe Photoshop, Adobe After Effects; Pro Tools; Microsoft Xbox, Sony PS3 SDKs. Internal suite of dev tools

NOTABLE TECHNOLOGIES
Havok Physics, Havok Behavior SDK, CRI Movie Player, Fmod

TOTAL LINES OF CODE
1,500,000 lines of code, 85,000 lines of LUA

So you're ready to start developing **the next killer title** that's going to make the big dogs say, **"Holy crap, where did this guy come from?"** but you're going to need a talented team of people to make it happen and you're hoping to find them in a place that's not overpriced and full of posers because you don't have the patience to deal with that on top of the hellish 18-hours-a-day crunch to meet your milestones and you're also thinking that **it wouldn't suck** if someone could **give you money** to help with expenses **before your title is actually launched.**

go to **bradic.org**

BRAD|C

Visit us at GDC Booth #5201 Contact: Stacey Simmons at 225.389.7182

GAME CAREERGUIDE.COM

THINK SERVICES

GET YOUR GAME ON!

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

- News and features for students and educators
- Getting Started section — an invaluable how-to guide
- Message Boards

GAME CAREER GUIDE
Search
Get the latest Education e-News
News
Features
Schools
Student Games
Careers
Ask the Experts: Does My School Suck?
by Jill Duffy
Register today!
GAME CAREER SEMINAR
Learn and Network with Game Industry Gurus
GAME CAREERGUIDE
GET IMMEDIATE ACCESS TO GAME DEVELOPER'S GAME CAREER GUIDE
Download Your FREE Digital Copy Today!
GETTING STARTED
Let us help you get started on the right track!
CLICK HERE!
NEW! Download & Play Student Games!
Click here
FEATURES
Postmortems | Books | Theses
INDUSTRY NEWS
Featured Schools

No More IT for Me: How One Tech Vet Became a Game Programmer [07.28.08]

This is the story of how a 25-year IT veteran was able to break into the game industry. Robert Madsen started programming in 1979. In 2003, his 16-year-old son told him he wanted to become a video game developer, and secretly, Madsen realized that's what he wanted to do, too. Now, five years later, they're both making games for a living.

Ask the Experts: Translating Video Games [07.28.08]

Losses Continue In THQ Q1 Despite Wall-E Sales Bump [07.30.08]

THQ has announced the results of its first quarter, showing sales up to $137.6 million driven by its Wall-E titles and Wii exclusive Big Beach Sports, but still saw Q1 losses up nearly three times over the year prior $27.2 million.

- Gameshastra, SCEE Partner For Traditional Indian PS2 Sports Title [07.30.08]
- Slitherine Raises $7.75m To Fund Self-Publishing Unit [07.30.08]

DOWNLOAD AND PLAY THE LATEST STUDENT GAMES!

DIGITAL COUNSELOR

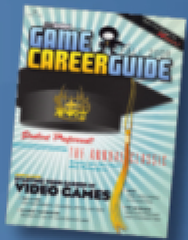
NEW!

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

Sponsored by Ai The Art Institute of Pittsburgh® Online Division

www.gamecareerguide.com

DOWNLOAD YOUR FREE DIGITAL EDITION OF THE 2008 gamedeveloper Game Career Guide ONLINE AT: www.gamecareerguide.com



GAME CONNECTION

Exhibitor Registration Deadline:

February 13th

GAME CONNECTION AMERICA

@ GDC, San Francisco, USA

March 24-26 2009

WWW.GAME-CONNECTION.COM

interstudio.fr - photo : © aurelien dupuis

>> david hawes

SNAKES ON A SEAMLESS LIVING WORLD

//////// **Embedding Stackless Python into a Console Engine**

STACKLESS PYTHON, CREATED by Christian Tismer, provides a lightweight threaded-style environment in which to run tasks. It allows hundreds of microthread "tasklets" to run with minimal thread management and provides super-fast context switching (compared to traditional threading) as well as a robust environment through exception handling.

This article describes how to embed the stackless Python scripting language into your engine. I'll be discussing Python and C++ bindings, embedding Python in a production console engine, and the architectural techniques needed to make the best use of stackless Python.

The information presented here is based on work we at Eutechnyx in the U.K. have done as part of our research and development. It is assumed the reader has a working knowledge of standard Python programming, C++, and game development on the Xbox 360 or PlayStation 3.

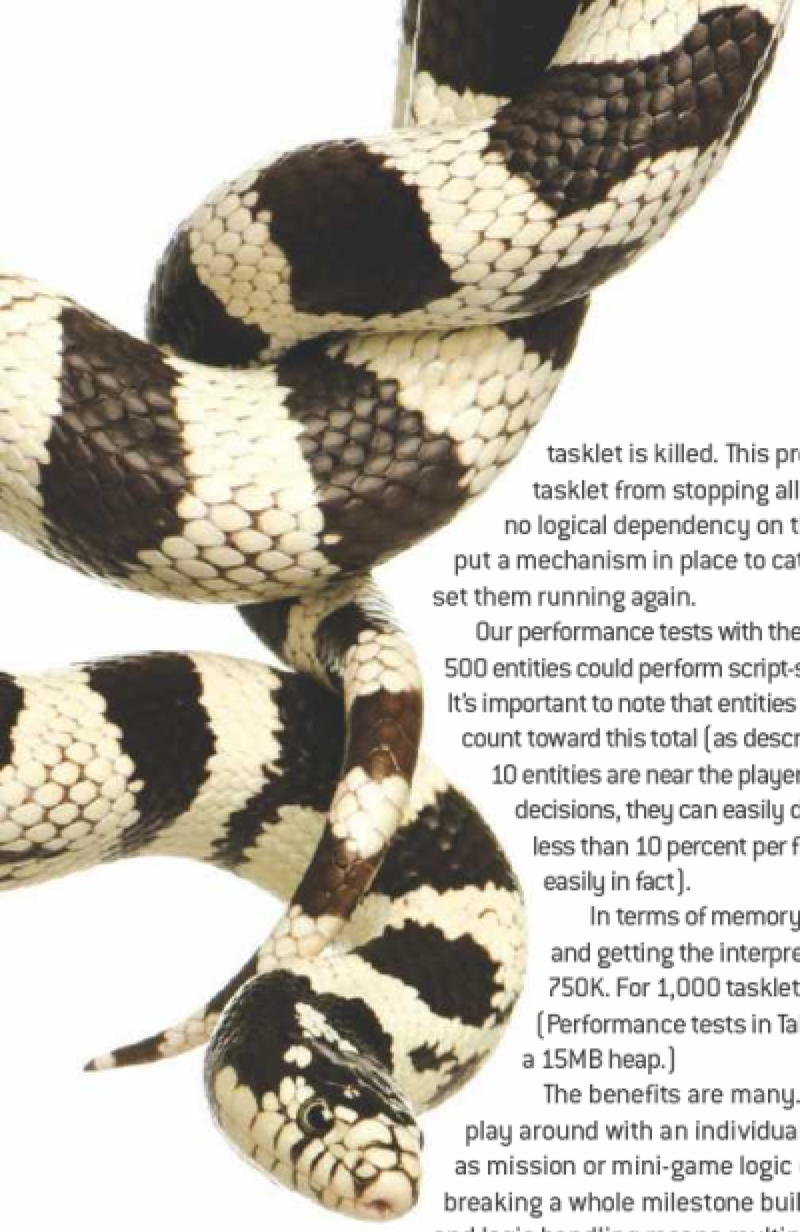
PYTHON BREAKDOWN

The primary synchronisation primitive of stackless is the channel. A tasklet represents a single block of code that will be executed at the whim of the scheduler. Numerous tasklets may be running, and the scheduler uses a simple round robin approach to determine which tasklet will run next.

A tasklet can always yield to the scheduler, and in doing so allows another tasklet to run. A tasklet can also block on a channel; that is to say, remove itself from the scheduler until something is put on that channel, at which point the tasklet will resume execution and be able to take whatever Python object was put on the channel. It doesn't necessarily do anything with the object it received, but clearly this communication mechanism between tasklets can be useful.

Exceptions thrown by tasklets (if not caught before reaching the scheduler) are caught and the

DAVID HAWES *is lead programmer and project manager at Eutechnyx UK, an independent driving game development studio. David has worked on numerous titles on a variety of current and next-gen platforms and is currently working on a triple A sandbox game for a major international publisher. Email him at* **dhawes@gdmag.com**.



tasklet is killed. This prevents errors in a single tasklet from stopping all tasklets, even those that had no logical dependency on that target, from failing. We can put a mechanism in place to catch such failed tasklets and set them running again.

Our performance tests with the Xbox 360 suggest around 500 entities could perform script-side logic every few minutes. It's important to note that entities blocked on a channel do not count toward this total (as described later). For example, if only 10 entities are near the player and need to make a lot of decisions, they can easily do so within our goal of spending less than 10 percent per frame on entity logic (quite easily in fact).

In terms of memory costs, loading a few scripts and getting the interpreter started up uses about 750K. For 1,000 tasklets, 15MB of heap is used. (Performance tests in Table1 were all carried out using a 15MB heap.)

The benefits are many. Designers are able to play around with an individual entity's behavior, as well as mission or mini-game logic on the fly, without fear of breaking a whole milestone build. Clear separation of asset and logic handling means multiple designers can work on disparate sections without fear of using up one another resources.

Producers can see changes in the game more quickly in the areas that matter most to them. And stability issues are reduced in deployments, as the most frequently changed logic now has a protective layer between it and the game core.

Putting this all together we aim to get a system that looks something like what's shown in Figure 1.

PYTHON AND C++ BINDINGS

Python provides the Python/C API, which allows you to write fully functional Python modules in C++ and decide what you want to expose. There are numerous intricacies involved in implementing this on any large scale—deciding whether C++ is currently in charge of an object's memory or Python not being the least of them.

Additionally, in many cases the code is clearly going to be similar or the same. Various solutions exist to automatically generate the binding between the languages. During early development, these were the options we considered: Boost, Pyrex, SWIG, and SIP. There are a whole host of situations in which each of these might be the best choice.

For our purposes, we picked SIP. SIP handles wrapping a very large set of C++ functionality. For example, you can create C++ class hierarchies that—as long as you provide some kind of RTTI equivalent information to your SIP scripts—can provide correct polymorphic functionality of all methods to objects wrapped in that hierarchy when a base pointer is passed into Python. It's very flexible. If it can't automatically map a type from the information you provide it in the SIP scripts, you can directly include Python/C API code in the script to do the conversion. There are many other useful features of SIP, some of which are shared by SWIG, but there is one area in which SIP is by far the best—speed. (See Figure 2, pg 28.)

While it's reasonable to argue that you will always get the fastest performance from manually wrapping your code, it quickly becomes evident that this is a huge amount of work when considering a significant body of code. For speed, and for build time performance and for meeting our requirements in terms of what we want to wrap, SIP is the ideal choice. You will need to download the source and build the SIP executable with parameters appropriate to your development environment and whatever version of Python you're embedding.

Listing 1 (found online at www.gdmag.com/resources/code.htm) contains an example of a SIP script that wraps a single C++ class that can be accessed via a Python Module named "GameEvent."

TABLE 1

XBOX 360	TASKLETS	ROUNDS
44.5	100	10
140.4	100	100
8,148.3	1,000	1
8,260.1	1,000	10
20,143.2	1,000	1,000

Time in milliseconds for a number of tasklets to perform a loop of message sending "Rounds" number of iterations.

EMBEDDING PYTHON IN A PRODUCTION CONSOLE ENGINE

Pick a distribution of Python that suits your needs. We opted for 2.5.1, the latest version at the time. Using the source code for the distribution, create a static library and link your executable. Add the stackless source code to this project.

From the same source code, we build our own version of Python.exe that can be used in the pipeline to create `.pyc` files from raw Python so we have precompiled byte code ready for the engine to use. We could use `.pyo` files, but at the moment, removing assertions is not a huge benefit to us. Then we just need to get our updated executable compiling and running on the consoles.

Modifications for PlayStation 3 and Xbox 360. Modify `import.c` to ensure that we only ever load `.pyc` files. This is to prevent us from trying to load non-byte-code compiled Python. Modify `_PyImport_StandardFiletab[] = {}`. This specifies which extensions are sought during an import. We never want to perform compile to byte code on the fly, so we should make sure we only expose `.pyc` files to our file system.

Interface with the filesystem for the platform by re-implementing `Py_ungetc`, `Py_putc`, `Py_getc`, `Py_fgets`, `Py_fopen`, `Py_fclose`, `Py_fwrite`, `Py_fread`, and `Py_fseek`. Ideally this should be done at a non-platform-specific level when working on multiple platforms. Now we have a chance to hook in and check for imports that we don't want to allow, or if we want to do something funky in the file system (maybe load a compressed file into memory and decompress on the fly via an SPU), we can hook that all in here.

Modifications for PlayStation 3. Assuming we have a platform definition available to the pre-processor (`#ifdef PlayStation 3`, for example), we need to make the following changes wrapped in such a definition:

CONTINUED ON PG 28

VICIOUS ENGINE®

making games. easier. faster. better.®

Visit us @ GDC Booth # 5840 North Hall.
Book appointments for GDC now.

www.viciousengine.com

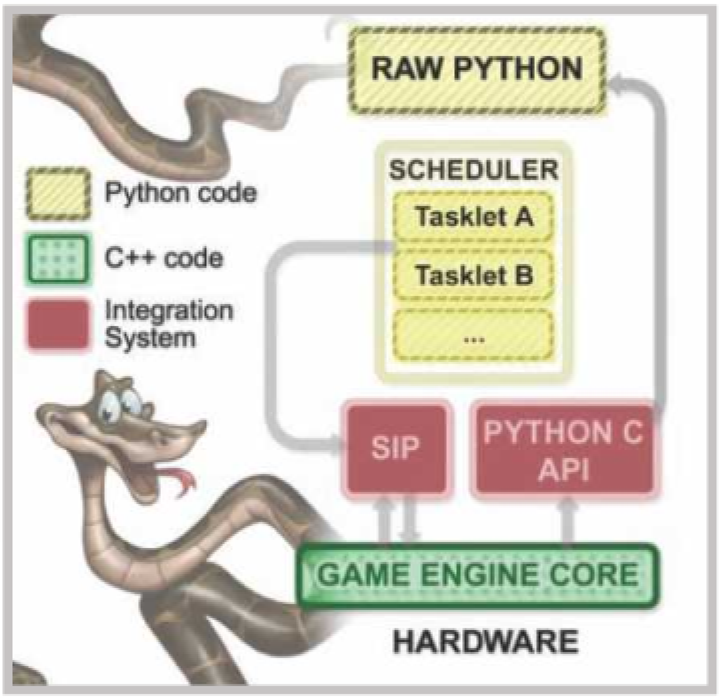


FIGURE 1 A Stackless Python Integration is shown.

CONTINUED FROM PG 26

- platform-specific implementation of signal handling needs to be provided for PlayStation 3 in Pythonrun.c
- PyOS_setsig and PyOS_getsig end up more or less empty on PlayStation 3 (we are not interested in this functionality)
- HAVESN_PRINTF needs to be disabled so that the Python interpreter emulates this behavior
- similarly isatty() function requires equivalent PlayStation 3 implementation
- disabling Unicode requires a slight modification to ast.c; decode_unicode needs to be ifdef'ed out
- portable Fseek and Ftell will require equivalent PlayStation 3 calls inserting into the function bodies
- PyConfig.h will require modification to inform the interpreter that on PlayStation 3, signals are not supported.

The crux of getting PlayStation 3 stackless working versus regular Python is implementing the stack switch assembler, which handles jumping between c-stacks when switching tasklets (See Listing 2, found online at www.gdmag.com/resources/code.htm).

In "slp_platformselect.h," we need to add an ifdef directive for the PlayStation 3 platform and include the all important switch_ps3.h.

Modifications for Xbox 360. Xbox 360 is a little more straightforward, possibly due to it being a touch more closely related to the PC. But it still requires its own implementation of the stack switch code (see Listing 3, found online at www.gdmag.com/resources/code.htm). Again, we will need to make the appropriate modification to "slp_platformselect.h."

HEAP USAGE IMPLICATIONS

What we have seen so far should be sufficient to get stackless Python running in our engine. At this point, though, we still are a long way away from being able to use it in a production title.

The next step is to take Python's heap usage and put it in a box that we control, allowing us to measure its usage, prevent fragmentation, and have an aid in both debugging and optimization.

We controlled memory allocation using our own internal pool allocators (we allocated a large block at the start of the application for Python and allocated chunks out of it) and the following Python functionality:

- PyMem_MALLOC macro redefined to call EtxMalloc
- PyMem_REALLOC macro redefined to call EtxRealloc
- PyMem_FREE macro redefined to call EtxFree

Obmalloc.c is modified so that direct calls to C library Malloc, realloc, and free are replaced with the Etx calls above. Now we are able to take advantage of Python's own speedy and cache-conscious object allocator, optimized for large numbers of small mallocs.

When using your own favorite managed heap, bear in mind that Python never expects to get NULL back, even when requesting zero bytes. This task is instead handled by the macros and functions that call the custom allocator (always

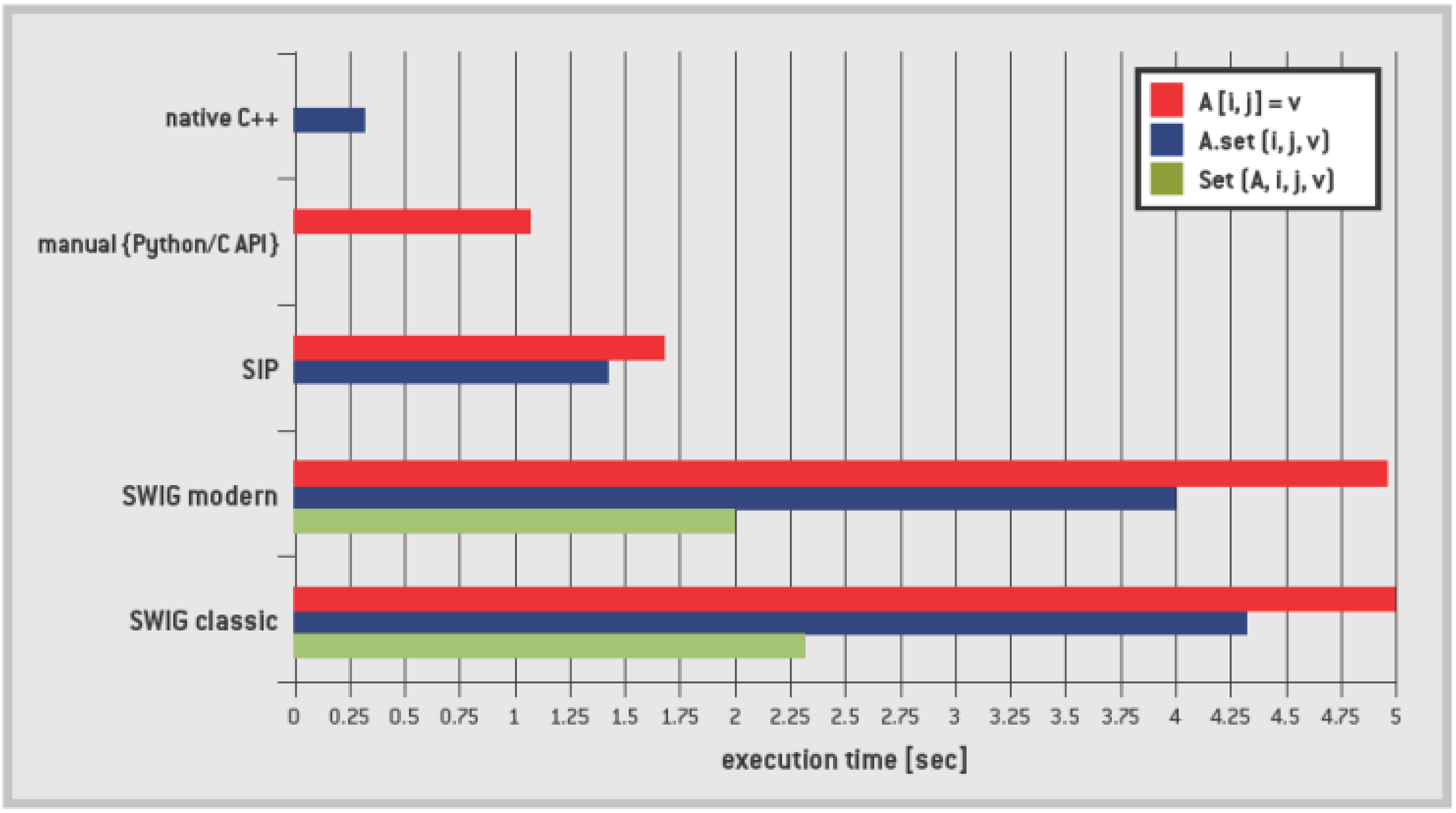


FIGURE 2 A performance comparison between various C++/Python bindings is shown (graphic from Paul Scherre Institut research presented at Europython 2004).

requesting at least 1 byte). Be sure to assert internally in your custom allocator prior to a NULL return.

Be wary when overriding the allocation behavior to make sure you don't ever return Python memory from different heaps or cause the Python object allocator to be used when it would not normally be. That is, don't change other `malloc` calls to be routed through the `obmalloc` interface; route them through the macros above. Either of these mistakes can cause very nasty bugs.

It's also useful to route any `malloc` calls you make on behalf of Python to this managed heap (using your own calls directly). For example, if you allocate memory for a file handle and want to keep Python import costs budgeted differently from your normal file loads, you might want to route allocation of the handler through here.

MAKING USE OF STACKLESS PYTHON

Once we have a working version of stackless Python in our engine and have its memory allocation under control, how do we actually make use of it?

First, we can use stackless Python as a debugging console. Implement a simple text console in your game so you can always pump commands to "`PyRun_SimpleString ()`" from in-game. This is such a powerful debugging tool once you have a good amount of your engine exposed.

But a word of warning: While in many situations it is handy to be able to just plug in a USB keyboard to an Xbox 360 and pull up the console, the USB keyboard support on the Xbox 360 goes through a debug driver at present, which can be quite unresponsive, making typing a chore. We used a virtual console on the PC that would send commands over to the text console on the Xbox via our network.

Second, we can use the application as an operating system. It was Harry Kalogirou, creator of the Sylphis3D game engine, who first proposed this model (http://harkal.sylphis3d.com/2005/08/10/multithreaded-game-scripting-with-stackless-python/). To get the most out of it, we need to change how we think of the application. The application is an operating system that provides high-speed, low-level (C++) services to its applications (Python tasklets) that run on it. It determines when they run, for how long, and whether it honors requests made to services.

Moving as closely as possible toward this model lets us make the core more robust and puts extra safety between requests made by game objects and the engine actually doing anything about it. The results can really help stability.

Every object that can be interacted with in the world should have a Python class, an instance of which is stored in a Python database at all times. It contains the persistent data about the entity. When a player is close to an entity's physical position, the relative asset needs to be loaded, and the database entry associated with it needs to be synced with this visible asset. In fact, for game terms, the instance in the database is what we consider "true"—it's the master data, much like the server's view of the world of an MMO.

A third use for the embedded stackless Python is loading assets by assigning processed assets (a "resource") a unique ID automatically. We do this using an offline tool (developed using wxPython, C++, and SWIG!) that maintains an XML database of assets for the game. In this case we are really talking about an object view of files. For example, an in-game car is made up of a mesh for its graphics, some textures, some shaders, a simpler mesh for its physics, and a script that describes its performance and handling characteristics. We can turn the whole bundle of defining information into a set of archives, and associate them with a resource ID that is visible to Python. The game engine gets some data loaded at the start that tells it which IDs map relate to which archive files in its file system. We also give Python a file that specifies how much memory needs to be allocated in-game for a particular archive to be loaded.

When an entity is added to the database, it becomes associated with a resource ID if it is to have a "physical" equivalent (cars and characters do, missions do not). Assigning individual entities unique IDs is done automatically when creating the database through a straightforward Python script that uses a set of criteria defined for our game to generate the required entries. Because we're dealing with an open-world game, the remainder of the system functionality reflects this.

In the Python script that describes a particular mission, entity IDs specifically relevant to the mission are listed. Python code calculates what resources are needed for that set of entities to be used, and requests that the game load them. Essentially, it asks for a list of resource IDs and a total amount of memory to be allocated for the files, using the information generated about the archives as described.

A tree-like structure of state machines that manages what code is currently running on the C++ side includes a state that is able to receive messages from Python requesting asset loads via resource IDs. In reality, we have a data structure that lets Python request all kinds of things that require asset loading (particle systems, music, and so forth). The structure is defined on the C++ side and is wrapped via SIP so that Python can create instances of this structure, fill it out with a request, and (via another SIP wrapped function) send the request to this loader state. The loader state queues up such requests and gradually loads them. It also retains information about what it has loaded so that Python can request that a particular package that was loaded previously now be unloaded.

The C++ side loader state determines exactly which assets to load as well as how and when to load them. For example, when Python asks for `sportscar3`, the C++ determines whether that should be `LOD0` or `LOD1`, and whether it should load now or prioritize a block of terrain that needs loading.

Upon completing the loading of an asset package (a resource), the C++ side calls into Python. This generates notifications that the resource can then be synchronised with the matching entity in the python database. Now the entity in the database has a reference to the high-fidelity C++ mesh and physics data that represent it in the world. This also means when other scripts interact with the entity, it can return exact data about its position and its velocity.

In a sandbox game, the world is often filled with trivial entities of one kind or another. Many of these entities will use the same assets as non-trivial ones in a mission. Consider a mission to kill 10 snakes by the river. If there are always some snakes by the river, why load 10 snakes for this mission when there are already, say, eight that could contribute to the total 10 without further burdening the streaming? Why not associate entities with any loaded resource that matches its resource ID and is close to the entity's database position?

This solution is not always suitable, and we will want the functionality to force a load from the Python side, but in many cases in which x number of trivial NPCs are part of a mission, this optimization method can be useful.

Another use for the embedded stackless Python is to implement event-driven systems. We really want to keep on top of performance. We also need a general way of communicating

key parts of what is happening in the high-fidelity C++ simulation and the low-fidelity Python database so that Python can react and make new requests for change in the simulation. An event-driven system is the key to meeting both.

Tasklets register for events that they are interested in upon creation. When an event is resolved, they can deregister or register for new events. These changes are handled by a callback-map in Python that periodically syncs with C++. This occasional sync really helps with performance and gives a good chokepoint for debugging. The game engine can generate thousands of events from the high-fidelity simulation every frame, and the events filter knows which Python entities are interested in what, so it can quickly throw away irrelevant events. It will build up a buffer of events that needs to be sent to Python, and periodically it will pump this whole buffer through a single Python call to the callback-map so it can disseminate the events among the relevant entities. (See Figure 3.)

Since tasklets can block while waiting for events, they reduce the processing load; by controlling how many events we register for (or how frequently they occur), we can balance performance against fidelity against number of entities.

Another job we need to support is the reloading of scripts on the fly. A naive approach would simply be to call "`reload()`" on modules you have changed (and hopefully compiled to pyc and copied onto your console's fileserver). The problem is in handling instances of objects created from those modules. Ensuring you recreate objects correctly can be very messy. If you're not careful, you can have two instances of a game object that are supposedly identical running different code.

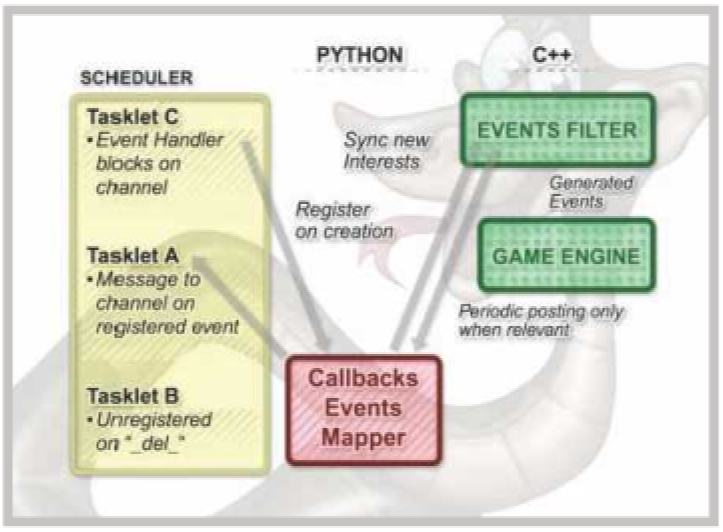


FIGURE 3 An event system diagram is shown.

We solved this by introducing a small but important limitation. We know that the scripts we most frequently want to reload are the event-handling components of individual entities in our database, and the logic for the missions the player can attempt. By making missions another type of entity in our database, we can handle them all using the same reloading rules.

Any object in the database can only ever be accessed externally via a proxy that automatically retrieves the correct object from the database via a unique entity ID. Every entry in the database has a unique ID. Now if the base object used for entities in the database knows which modules it was loaded from, we can have the database automatically handle the reloading of its objects and correctly tying up proxies, and the rest of the code never need worry about reloads.

TORONTO, ONTARIO IS HOME TO A remarkable assemblage of independent game developers, most of whom work alone or in small groups, driven by a passion for their art and free from the compromises so often associated with corporate life. We asked Nathan Vella of Capybara Games and Raigan Burns of Metanet Software to help us take the measure of Toronto's game development geography.

DO IT YOURSELF

"So far there isn't a big studio presence here, especially since Pseudo shut down," Burns told us. "Koei's around and there's a branch of Rockstar out beyond the suburbs. There are other companies, but they tend to be located outside of Toronto for economic reasons. And they're less fun," he added. "Other than that, it's mostly small or medium-sized groups, focusing on PC and mobile rather than console. There are a lot of people making games though."

Vella agreed, telling us, "There is a virtual ton of talent in Toronto, but the fact that there is almost no large-scale game development in Canada's largest city is one of the major reasons why the indie scene does so well. You have a massive talent pool of people who really want to be game developers with nowhere for them to work. In our case, we did the only thing we could—start our own company and make an opportunity for ourselves."

Capybara Games, the developer of PILLOWFIGHT and CRITTER CRUNCH, had its origins on the Toronto IGDA forum. "At the time there were lots of people in Toronto who loved and wanted to make games, and almost no gaming companies in the city for them to work for. After the thread on the forum, we started meeting at our local IGDA chapter meetings, then moved to meeting at a crappy bar every Monday for beer and game development talk. From there, we decided to try our hand at making our own games," Vella told us.

"We all had other fulltime jobs in enterprise software or television or something else remedial, and would work evenings and weekends developing SUPER SHOVE IT and S.M.A.B.U.," Vella said. "We got our first deal developing CARS for Disney/Pixar. We quit our day jobs, got an office and started making games for a living."

Metanet Software is also one of Toronto's many self-starters. Comprised of Raigan Burns and Mare Sheppard, Metanet used Flash to create the freeware gem N in 2005. As N's audience grew, Metanet made the decision to keep itself small and manageable by partnering with Slick Entertainment to produce an expanded version called N+ for Nintendo DS, Sony PSP, and Xbox Live Arcade (see the postmortem in the September 2008 issue).

HELP FROM ABOVE

The Ontario Media Development Corporation (OMDC), an agency of Ontario's Ministry of Culture, plays a significant role in the Toronto arts scene by providing local book, film, and music producers as well as game developers with grants and tax incentives.

"The OMDC has made a huge difference for us—both in terms of helping us get our ideas off the ground through financial and business support, as well proving to us that the Ontario government actually cares about the videogame industry," Vella told us. "Their funding also helped us get ideas off the ground that never would have seen the light of day had we needed to go through the traditional publisher green light route."

"However, it's not like they simply hand out money—it's a highly competitive pitch-like process. The upside, and reason that the OMDC does make a big difference, is that they treat us indies on equal footing with the larger more established companies when it comes to competing for grants," Vella added.

"Metanet would probably not be around if it wasn't for their support—the OMDC allowed us to afford to go to GDC when N was in the IGF!" Burns told us.

COMMUNITY

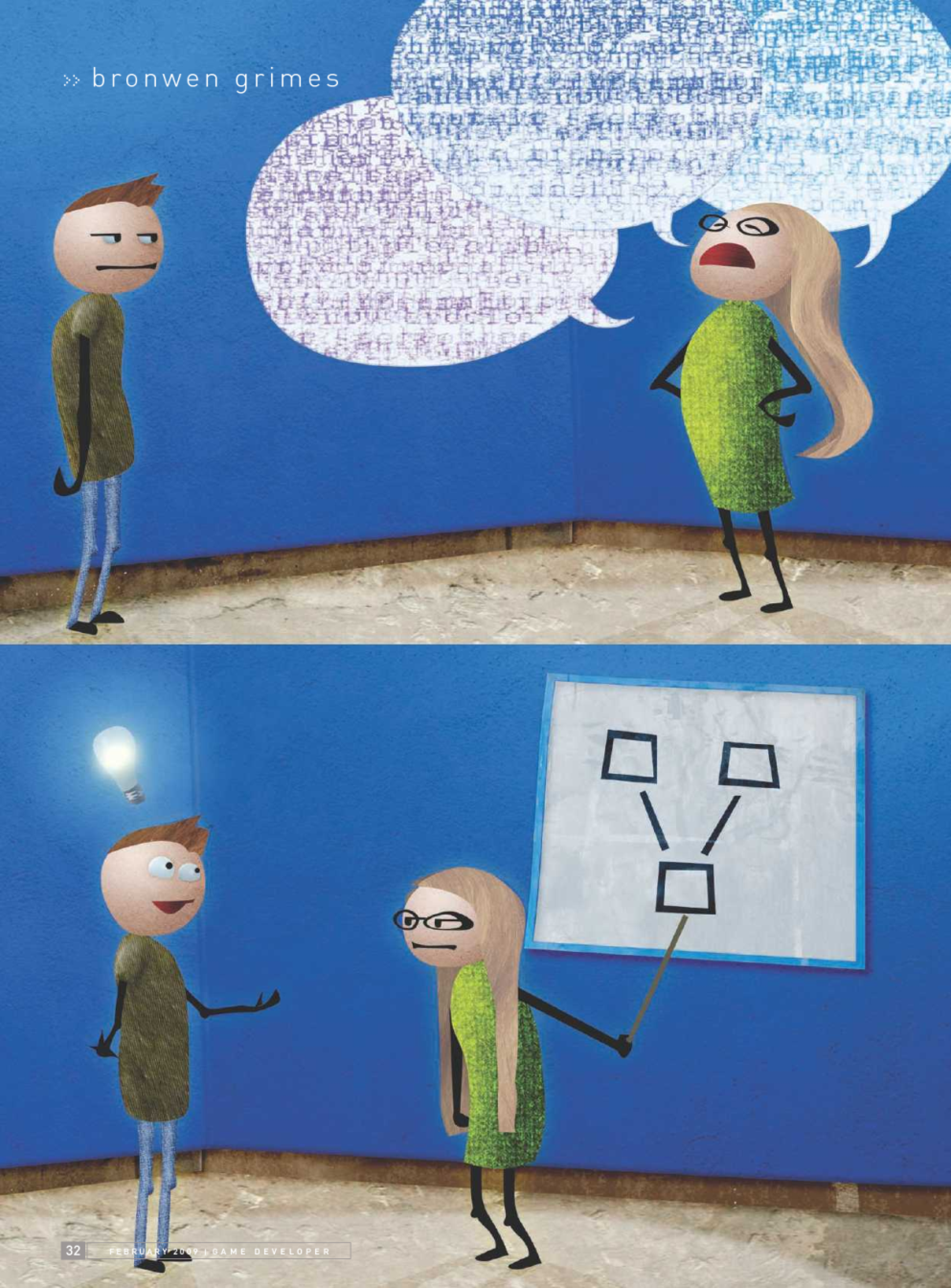
Perhaps more than anything, Toronto benefits from having a community of developers who are committed to helping each other out and are determined not to allow their nascent scene to fizzle. The Toronto Independent Game Development Jam (TOJam) organized by Jim McGinley of Big Pants Games (HOLD ME CLOSER, GIANT DANCER) brings together local developers to collectively bash game ideas into complete form over three days of intense coding. TOJam has seen the emergence of a number of developers who have gained a wider recognition since. Jonathan Mak of Queasy Games created TOJAM THING at the 2006 event and has since released EVERYDAY SHOOTER for PlayStation 3 and Windows. Shawn McGrath of][(right square bracket left square bracket) whose game CHAIN3 is currently available for iPhone, built the award winning A GAME ABOUT BOUNCING at TOJam 2008. Now in its fourth year, the next TOJam is set for May 2009.

The Artsy Games Incubator is another significant boost to Toronto's indie scene. Headed by Jim Munroe, the creator of EVERYBODY DIES, along with help from Metanet and Queasy Games, the Artsy Games Incubator provides a guided framework for creating independent games. Operating as a combination lesson plan, manifesto, and hands-on workshop, the Incubator has helped bring to fruition such games as Benjamin River's SNOW and Miguel Sternberg's NIGHT OF THE CEPHALOPODS.

"The Toronto scene is a pretty tight community," Vella said. "We help each other out where possible, mostly just through having a network of talented and successful people to talk games with. The success of local indies really fuels the scene and help us all believe that we can be successful as independent developers in Toronto. Honestly, it's hard not to be encouraged by games like N+ or EVERYDAY SHOOTER getting the recognition they deserve." ⁙

JEFFREY FLEMING *has not finished reading* Beautiful Losers *yet. Email him at* **jfleming@gdmag.com**.

>> bronwen grimes



GOOD MORNING CLASS!

////// Introducing new tools to artists without getting spitballed

NEW TOOLS AND TECHNIQUES HAVE the ability to revolutionize, or at least truly hone, the work of a game artist. The trouble is, few game artists want to stop making the game in order to investigate these new options. In steps the technical artist. For six years I worked at Pseudo Interactive, a game development studio in Toronto. In 2007 Pseudo rolled out a new tool called BRISE, and along the way, I wound up learning a number of valuable lessons about the introduction of new tools.

I want to start by noting that the technical artist represents a new profession in a young industry. There is no consistent job description if there is even a job description at all, and many of us are figuring things out on our own. A lot of technical artists have fallen into the role from some other position, mostly driven by an inability to leave broken things alone. The discussion of what makes a technical artist good at her job would be a long one. Problem solving, communication skills, patience, perseverance, multi-tasking, and the ability to keep an open mind are all important attributes. But equally important to how we do the job is who we do it for. After all, fundamentally, we provide a service.

BRONWEN GRIMES *is a technical artist at Valve. She also contributed the art for this article. Email her at* **bgrimes@gdmag.com**.

Technical artists are enablers. Although many of us are talented artists or programmers (or both) in our own right, very little of our work ends up being immediately visible in a game. So instead of thinking of the game's consumers as my customers, I've learned to think of my customers as the people who use the tools and shaders I create. And if my customers aren't happy with a product, they're not going to use it.

Now back to BRISE, which was a node-based shader editor (similar to ShaderFX, though not DCC-software specific). "Brise" means break in French, signifying that this tool was a break from our old system.

Boy was it ever. Our old system was flag-based, inflexible, and thoroughly entrenched. The new tool was completely open-ended. My task was to introduce entirely new concepts to the art team and get them productive again within a short time frame. As the development of the tool progressed, I had been documenting it and writing tutorials. I used the tutorials as a lesson plan, and took the artists through eight 3-hour classes. This brings me to the first lesson I learned in this process.

COMMUNICATE WITH THE ART TEAM, NOT AT THEM

Anyone who has been a trainer in the course of their jobs can sympathize here. I had a stressed team, a short timeframe, a complex tool, and a series of lessons on a projector. The temptation was to read from the screen just to get through it. After all, my job is to write tools and shaders. I'm not a teacher, right?

Dead wrong. The tool is useless without users. I had to force myself to ask questions, high-school teacher style, and use more examples than I'd actually prepared and documented. Some people got it quickly, more quickly even than I'd anticipated. Some lagged. I told myself this was normal. Not everyone would be an expert user. I'd focus my time later, one-on-one, on those I picked out to be shader-writing prodigies.

Everyone needed a base level of competency, though. Even if some of them weren't going to write shaders, they at least had to be able to use ones created by their teammates. One of the artists I was working with repeatedly failed to grasp a crucial concept, even to the point of becoming argumentative.

Unless you're using multiple render targets, pixel shaders output a single float4: red, green, blue, and alpha. We'd been using alpha not just for transparency but for masking different screenspace operations, based on the needs of the object being rendered. But it's a choice; either the shaded object uses the same alpha values for multiple different operations, or you pick just one operation.

The artist in question wanted to use two different alphas, one for transparency and one for a screen-space glow. I tried to get across the point that although he could source multiple textures, he could only output a single alpha value to the framebuffer. It didn't make sense to him; he had two different alphas. He could see them! Explaining that they had to be composited into a single channel to output to the framebuffer got me a blank stare.

After several trying minutes, I drew a simple inverted pyramid to show that although he could start with many images, he could end with only one pixel value. The diagram did what ten minutes of talking hadn't, and the light dawned on him. He responded immediately, "Oh, I get it. I'm a visual learner, you know."

He wasn't the only one who had an epiphany. In retrospect it seems obvious that an artist would be a visual learner. I had been trying to communicate in the way that I learn best instead of trying to understand how he sees things.

I think this is a common problem. I hear quite often from programmers and other technical artists that a certain artist (or sometimes artists in general) is not "technical." This is often used as an excuse to dismiss their concerns or to pass the work off to a different, more "technical" artist.

I have a hard time believing that excuse, though. We're talking about artists who regularly use complex software such as Maya and Photoshop. Every traditional artist I know would consider that work highly technical. I firmly believe that any of these artists can learn complicated processes up to and including scripting, if they have a good reason to do so. You just need to present the information in a way that motivates them.

PUT YOURSELF IN THE ARTIST'S SHOES

I'd had a difficult time, initially, convincing the production team and company owners that BRISE was a good investment. I'd be the only one able to use it, they argued. All my time would be taken up writing shaders, and I'd be under a lot of pressure to support the entire team. The current system was understood and useable by everyone, so why change?

I was determined that the artists could learn the system. Give them the power, I argued back, and they will do amazing things with it. Although I still maintain this as a basic tenet of my professional philosophy, I allowed my conviction to blind me. I structured not just the lessons themselves but the entire lesson plan in a way that made most sense to me—as a tool manual for creating shaders. How do you connect nodes? How do you make a basic diffuse shader? I wanted to turn out more shader writers, just to prove my point.

I had failed to ask myself the most basic of questions. What does the artist want to accomplish with this tool? I'd answered with what I wanted to accomplish: write shaders that support the art director's vision. But is that what the typical artist wanted to do?

The first thing our art team wanted to do was apply a generic shader that would let them see their objects rendering in-game. This is how I should have started my tutorials. Instead, it was the last lesson. The artists ended the tutorial sessions weary and over-challenged. We didn't get to the part that was most relevant to them until the very end. If I'd placed myself in their shoes, I could have better seen how this tool fit into the overall pipeline, and approached the tutorials from a top-down perspective, instead of from ground-up.

This was a really tough lesson for me to learn. Like many technical artists, I started out as an artist who could script. For several years in my career, this was how I worked: see a problem, fix it for myself, and clean it up for the team. Being your own customer makes tool or documentation writing really easy. Putting yourself in someone else's shoes is a challenge.

CONVINCE THEM THE PAYOFF IS WORTH THE PAIN

Technical artists are very process-oriented. We want to know everything about a tool, figure out new and creative ways to use it, and understand how it can be changed or extended to make the content development process better.

Every other artist wants to solve visual problems. That, correctly, is where the majority of their brain-power goes. They don't want to spend time thinking about how to make a tool do what they want. They want to think about mood, style, and composition. They are goal-oriented. They especially don't want to try something that takes them completely out of their comfort zone. When an artist is really familiar with a tool, using it is like using a pencil. The mechanics are automatic. Only the output is interesting to them. So convincing an artist to try a new tool can be difficult, as this makes it much harder, initially, for them to reach their goal.

Even if I think a tool will make an artist more productive, I have to remind myself that any energy they put toward learning a new tool is energy they aren't using to produce artwork. I ought to have considered my tutorials in the context of the pipeline. New tools must be considered in that context as well. Does the new addition take an artist out of a comfortable environment? How many applications have to be open for an artist to complete an asset? When switching between applications, how many opportunities are there for the content to break?

Sometimes you need to market a tool to your customers. Their opinions can only be as good as the information they are based on. Showing them features and time-saving shortcuts will help motivate them. If you can't convince the art team that the new tool will make them more productive, you may have to go back to the drawing board. Since they're the ones doing the work, they're in a good position to judge the value of a new addition.

I ought to take a moment here to clarify that I consider "more productive" not just to mean "make more stuff" but also "make better stuff, faster."

BRISE fell into the category of "better stuff, faster." It was pretty far outside the comfort zone of most of the artists, and was initially disappointing as a result. During the first week it looked to them like "mediocre stuff, slower." As I mentioned before, the time frame for training was really short. We were in the process of building a small level which would be the nucleus for a larger game-play area, and the level was due in a few days. We managed to get it looking like it had in the old shader system within two days, which in itself was a success. Without a grand leap forward, though, it was hard for the team to see the possibilities inherent in the new system.

One of the core features of BRISE was referenced groups. I'd instructed the team on them, and made sure they were used religiously throughout all the shaders in the level. I had assumed that everyone would understand the power behind this simple concept and get excited. Just as I had structured the tutorials from my perspective, I had marketed my tool as though to myself. I'd shown them the process, but what they really needed to see were the results.

After discussing the visuals with the art lead and art director, I gathered the team around my desk to show them how one quick change could dramatically impact the whole level. I opened the specular group node in one shader and added a new output—an "out of range" output that took over-bright values and passed them to what was termed the "radiance buffer" in our engine (essentially a glow pass). Switching back to the level editor, I enabled the model's ability to write values to the radiance buffer, causing the specular highlights to bloom.

The art team gave some gratifying "oohs," and started to divvy up who would change which shaders to have the same functionality. I had to interrupt them to remind them that the node I'd changed was referenced into all their shaders. With 10 seconds of work, the entire level's look had been updated. They watched while I copied the write-to-radiance-buffer settings to the whole level, and the whole team erupted into grins.

Speed, power, and the will to use it. Finally, success. If only I'd shown them that first.

TECH TO THE FUTURE

The process of rolling out a new tool is always fraught with challenges. As technical artists, we're in the unique position of sharing our offices with the customers for our tools and pipelines. We've got the chance to react quickly to feedback at all stages of development. Keeping your ego out of the equation and really listening to your customers makes for truly great improvements and innovations in your pipeline.

The best thing that we can do, as technical artists or as tools programmers, is to trust and respect our customers: our colleagues. Working to see things from their perspective, tailoring the tools to their preferences, and showing them how powerful the tool can be will turn them into power-users, creating better content and, ultimately, better games. ⁙

christian nutt

INTERVIEW: TSUTOMU KOUNO

TALKING TO TSUTOMU KOUNO ABOUT HIS ICONIC CREATION, LOCOROCO, one gets a sense that there's a lot of hidden depth to this developer of simple games. It's refreshing to speak to a developer who wants to create games that appeal broadly not because it's a way to cash in to a major market—the PSP is not the most mainstream platform after all—but because he feels inspired to deliver original, pleasurable experiences.

This interview, conducted at Sony Computer Entertainment's HQ in Tokyo, reveals a bit more about the man behind this adorable, accessible series, which recently had its second installment.

CHRISTIAN NUTT: *When it comes to LOCOROCO, the one thing you think about is the control, and I was wondering which came first— the game design or the control? From where did you proceed when you originally thought of the first game in the series?*

TSUTOMU KOUNO: When I came up with the concept for the first LOCOROCO, I was spending a lot of time sketching game ideas while riding the train. These weren't just ideas for LOCOROCO, there were a bunch I was working on. [Shows drawing] So, this is the one that became the jumping off point for LOCOROCO. When I looked at it, I saw that with so many characters on the screen, this wouldn't be the sort of game where you'd control a single guy directly. But I also thought it might be cool if you could rotate the landscape around all of them. The picture's about the same dimensions as the PSP screen, and although it didn't have L and R buttons, I'd tilt from side to side and think, "Hey, this might actually work." After this initial drawing, I started churning out more sketches at a crazy pace. The PSP was also being released right around the time I first had these ideas, and using the L and R triggers to get the rotation seemed like an obvious choice.

CN: *You made this game specifically for the PSP, not for a home console. How did your thinking change when creating a game for PSP rather than a home console where the user could sit and concentrate for a long time?*

TK: When the PSP was first announced, it seemed to me like everyone was gearing up to make these complicated games for it, like sequels to games that had been on the PS2. I wanted to break that mold and make a game that really seemed at home on the PSP. That had been my thinking ever since I heard about the portable. Coming up with the idea for LOCO ROCO right around the PSP launch was good timing, and I was lucky the control scheme worked so well.

With LOCOROCO, I wanted every aspect of the game, from the music to the first visual impression you get, to be unique. Polygons, complex character models, realistic shadows and lighting have become the standard for games today. I wanted to go the other way, and use a simple, effective 2D approach that would make even non-gamers say, "Hey, that's cute. I'd play that."

It's also common in games for the player to have direct control over their character, but this isn't the case in LOCOROCO. The LocoRocos themselves often don't do what you want them to. It was sort of an experiment looking back, but my hope was to create something that, if done well, would introduce a new way to enjoy playing games.

CN: *What games inspired you at such a young age to want to become a game creator?*

TK: There were a couple of Japanese games for the PC like HYDLIDE and XANADU, as well as a text adventure game for Apple computers called TRANSYLVANIA. You know, the type of game where you enter a noun and a verb to say, "Do this with this." It had werewolves and all sorts of good stuff in it. At any rate, I had a lot of games.

CN: *What's funny is that the games you listed are relatively complicated genres like RPGs or text adventures. I find it interesting how you got from enjoying games like XANADU to making a game like LOCOROCO.*

TK: Well, I've been a big fan of games since way back, but I also sort of kept an eye on the industry. From where I stood, it seemed like a lot of the games coming out were very similar to each other, and that didn't seem like a good thing to me. Once I entered the industry myself, I felt like I had a responsibility to create something new. I remember feeling that really strongly: that I had to do something, anything, so long as it was different.

Christian Nutt *is features director of Gamasutra.com and had his theories about LOCOROCO's design shot down by Kouno. Email him at* **cnutt@gdmag.com**.

GDC® 09

learn
network
inspire

www.GDConf.com

Game Developers Conference®

March 23–27, 2009 | Moscone Center, San Francisco

Early Bird Ends Thursday, February 12, 2009

Register now and save up to 30% on conference passes

Visit GDConf.com for the latest sessions, keynotes, and news

THINK SERVICES

A DIVISION OF UNITED BUSINESS MEDIA LLC

OUR RATING SYSTEM :

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

AUTODESK MOTIONBUILDER 2009

Adding MotionBuilder to Your Pipeline

Review by Marc-André Guindon

WHEN IT COMES TO GAMING, PRODUCTION studios are always looking for a faster and more reliable way to animate. After all, animation is one of the most time-consuming tasks in 3D production. With that in mind, it's hard to see the benefit of adding yet another piece of software to your pipeline's shopping list.

In our animation studio NeoReel, we have used Autodesk MotionBuilder with several game and film projects, such games as PREY (2K Games), FOOTBALL ARENA (Electronic Arts), and the OUTLAW SPORTS series (Global Star).

MotionBuilder stood out for us for three reasons. First, it has a real-time display engine, period. It is absolutely wonderful to scrub a scene regardless of the content and see it play in real time. Second, the software was designed to be a character animation tool, so it's fast and isn't slowed down with complex features. Third, both keyframing and motion capture workflow are supported, and the final animation can be retrieved with most 3D packages, such as Maya, 3ds Max, and Softimage XSI, using the FBX file format.

Why should you add a character animation package to your pipeline when your other packages can already do the job? My answer is because MotionBuilder does it faster and in real time.

MOTIONBUILDER AS PLAYER

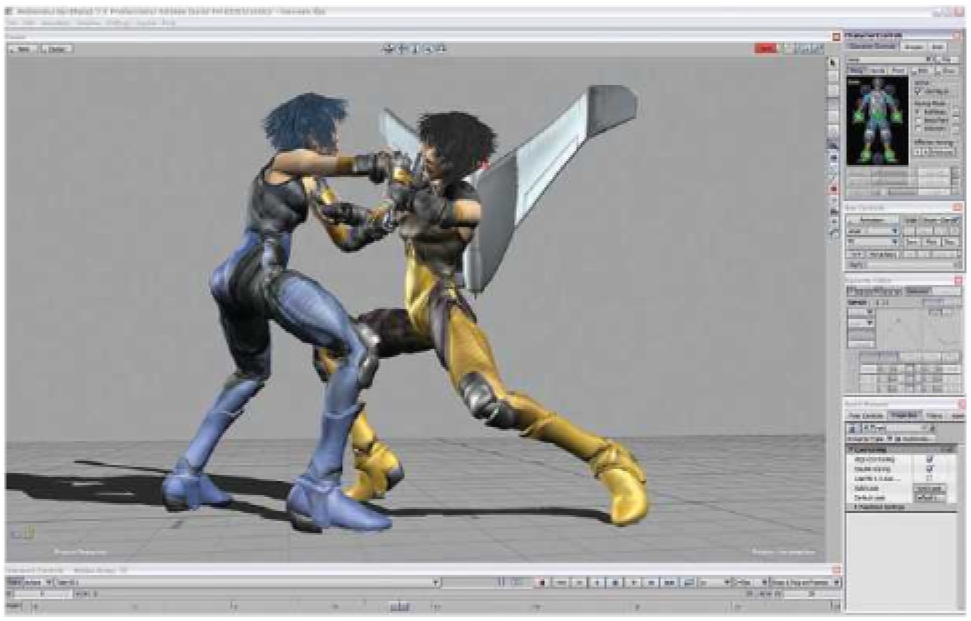
If you're using motion capture, you're probably already familiar with MotionBuilder, as it's one of the leading pieces of software in that field. If not, then you're probably stuck with slow scenes and feel cluttered by the thousands of keyframes that are hard to modify. And if you're using keyframe animation only, you might be slowed down by heavy character rigs and uncomfortable playback rates.

MotionBuilder has an easily implemented, one-size-fits-all, intelligent character rig that can accommodate most biped and quadruped characters. Animators won't have to learn new rigs, as all characters will be using the same fluid workflow throughout all your projects.

Because MotionBuilder is a character animation tool, you will still need to do all the modeling, texturing, and skinning of characters in your native 3D software. In our case, we use Autodesk Maya, in which the character is skinned to a FK joint hierarchy and uses blend shapes.

Once you're ready to export a character, you can use the FBX plug-in to bring the character in MotionBuilder. Next, simply drag and drop different body parts in the character rig, characterize it, and start animating. There's no need to render small movie previews of your animation because everything will be in real time.

As soon as you're happy with your animation, you can import it back into your native 3D software using the FBX plug-in.



THE BRIGHT SIDE OF MOTIONBUILDER

The latest version of the tool, MotionBuilder 2009, now has the ability to simulate rigid body and ragdoll dynamics, which can add a great touch of realism to your canned animation as the character rig and props can be blended in and out of a dynamic mode. Though the results are often generated on-the-fly by the game engine, it's always good to have a preview of the animation in the viewport itself.

Since MotionBuilder makes intensive use of your graphics hardware, it's possible to use CgFX shaders and per-pixel dynamic lighting to mimic the look of your game engine directly in the viewport.

The powerful animation system can greatly help you modify, reuse, and retarget existing animations. Further within, tools such as Animation Layers, Motion Blend, and Story offer outstanding non-linear animation tricks.

I've been using these cornerstone features from as far as version 1.0, and they are probably the top reason why I prefer to animate with MotionBuilder over Maya, 3ds Max, or Softimage XSI.

AUTODESK

★★★★

STATS

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

PRICE

$3,995

SYSTEM REQUIREMENTS

32-bit version: Windows XP Professional (SP2 or higher), Intel Pentium 4 or AMD Athlon processor, 512MB RAM, 1GB free hard drive space, OpenGL graphics card with a minimum of 64MB RAM.

64-bit version: Microsoft Windows Vista Business or Microsoft Windows XP x64 Edition, Intel EM64T, AMD Athlon 64, or AMD Opteron processor, 2GB RAM, 1GB free hard drive space, OpenGL graphics card with a minimum of 64MB RAM.

PROS

1 Real-time graphic engine.
2 Animation focused software.
3 One-size-fits-all character rig (biped and quadraped).

CONS

1 Often crashes without scene recovery.
2 Immature interface, scripting language, and SDK.
3 Dysfunctional undo system.

THE DARK SIDE OF MOTIONBUILDER

As with any tool, MotionBuilder has a few features and implementations that have left me with an unpleasant taste in my mouth. In some cases, aspects of several tools seem to require polishing, as if they were implemented as works-in-progress, and it seems Autodesk is waiting for the community to point out the problems, rather than implementing stable tools little by little.

To that extent, I tend to become frustrated with tiny problems as soon as I try to think outside the box. For example, the character rig is very easy to setup and use for standard characters, but it's cumbersome to use when replicating an existing rig or building on top of the basic MotionBuilder rig. Implementing a set of extra controls requires a good understanding of the software and can lead to frustration since it lacks some essential features of other favorite 3D software packages.

For instance, to set up a pair of wings on a character, you would need to build controllers from scratch without evolved rigging tools (in Maya, I would defer to the Set Driven Keys), and implement a character extension to plug into your current character rig.

In another example, if you want to create a simple snake character, you need to disregard the character rig completely and create a custom setup, which could use last resort techniques or non-animator friendly controllers, thus slowing down your animation tasks.

From a production standpoint, I've always opted to use MotionBuilder only on standard characters, and I would strongly discourage using characters that don't fall into the symmetrical biped or quadruped mold.

While MotionBuilder 2009 is making progress in terms of opening its interface and features to programmers, the SDK and scripting language are, in my opinion, very immature compared to other packages. If we compare it to Maya, for which the software was built from the ground up using the MEL scripting language and C++ SDK, it seems like MotionBuilder does the exact opposite, trying to deconstruct the software to give access to some features (but not all) with a Python scripting language and C++ SDK.

For instance, there is no easy way of creating a custom hotkey. We had to use brute force keyboard hooks to access some menu items and tools in order to speed up our workflow.

Using a software with a dysfunctional Undo system can be quite frustrating too. For most tasks, Undo works as intended in MotionBuilder, but the button does not work for everything. When you come across certain actions that you can't undo, it forces you to revert to the last saved file, a painful moment in any game developer's workflow.

Along the same line, I have in the past been working on a file when suddenly MotionBuilder crashes—and with no last-minute save or memory dump—prevents me from retrieving a portion of my work. MotionBuilder does have a Save Reminder setting that warns you when you haven't saved for the past few minutes, but that's hardly a replacement for a proper crash recovery function.

AROUND THE BEND

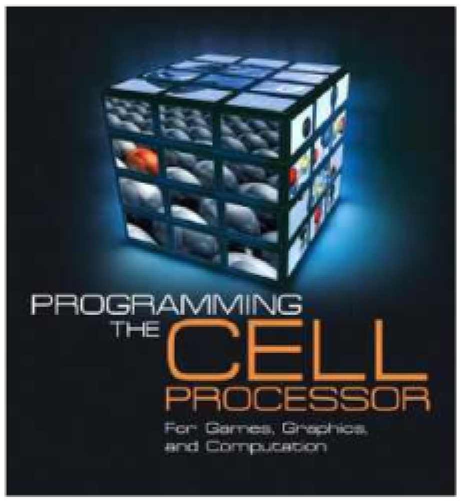
MotionBuilder is simply awesome for creating animation for video games. It's fast, easy to learn, and supports FBX file format across a wide variety of 3D packages. But you might want to take it for a test run before going full throttle into production (details about the 30 day free trial are available on Autodesk's website). Make sure you know what MotionBuilder can do for you.

From my experience, MotionBuilder was a time saver in most cases, but in certain situations, it was best to omit it from the pipeline.

MARC-ANDRÉ GUINDON *has been in the visual effects and games industries for 10 years, specializing in pipelines using MotionBuilder and Maya. Ambitious to redefine the industry and explore new ways of working, he has founded NeoReel, Inc., a production and consultation studio located in Montreal. Email him at* **mguindon@gdmag.com**.

Book Review by Bijan Forutanpour

PROGRAMMING THE CELL PROCESSOR FOR GAMES, GRAPHICS, AND COMPUTATION *BY MATTHEW SCARPINO*


PROGRAMMING THE CELL PROCESSOR
For Games, Graphics, and Computation

WHAT DO SPIDER-MAN AND THE PLAYSTATION 3 HAVE IN COMMON? THEY SHARE A common theme: "With great power comes great responsibility." According to the *Guinness Book of World Records*, the honor of the world's most powerful distributed system goes to Folding@Home, a worldwide network of PCs and PlayStation 3 consoles. As of April 2008, the network consisted of roughly 2.4 million PCs, and half a million PlayStation 3 consoles. Yet the PCs contribute "only" .259 PetaFlops, compared to 1.235 PetaFlops from the PlayStation 3 consoles. The PlayStation 3 delivers several orders of magnitude more performance than a single PC. With all that processing power comes the responsibility and complexity of programming the PlayStation 3's Cell Broadband Engine, which is where *Programming the Cell Processor For Games, Graphics, and Computation* by Matthew Scarpino proves to be very valuable.

The Cell processor was developed by the STI Alliance, which is composed of Sony, Toshiba, and IBM. The processor is used in Sony's PlayStation 3, IBM's Roadrunner supercomputer and Blade Servers, and Toshiba's SPURS engine. Both IBM and

Sony have released Cell Software Development Kits. It was amusing to read the author's description of his reaction in trying to understand the material covered in IBM's Cell SDK, since I could relate: "The material flew over my head with a fierce whoosh."

The book is a brave attempt at reducing the whoosh effect associated with Cell development. Three main topics are covered in order to give an understanding of the big picture. The first is the hands-on development tool set, the second is the hardware architecture, and finally the available software libraries. It is important to note that this book is not for the novice programmer and requires a very solid background in computer science. Knowledge of some form of Assembly is very helpful, as is some understanding of hardware architectures, operating systems, mathematics, and computer graphics.

Programming the Cell Processor For Games, Graphics, and Computation is very well written and organized, and does a great job describing the hardware architecture and how things work. That is not to say it is easy to read, because the subject matter is pretty complex. The majority of the book is wisely spent discussing in detail the internals of the PowerPC Processor Unit (PPU), the Synergistic Processor Unit (SPU), and SIMD (Single Instruction, Multiple Data) programming on the PPU and SPU. Communication and synchronization with the SPUs are discussed in the chapters covering Direct Memory Access, events, signals, and mailboxes. The chapter on advanced topics discusses SPU overlays, software caching, and SPU isolation for security purposes. Finally, the material on SPUs concludes with an exhaustive coverage of SPU assembly. At some point during reading the many pages involving hardware architecture, as things became blurry, the question inevitably crossed my mind: Why should I care? The author must have sensed the same, and explained: Because the main goal in having all this hardware is to use it, and to do so simultaneously whenever possible. Knowing how the hardware pipeline works helps prevent "pipeline bubbles," or wasted stages and wasted time.

There are a few very important distinctions that must be made regarding the context of this book. The focus is on the Cell Broadband Engine, and not the PlayStation 3 specifically. The SDK covered is the IBM Cell SDK, and not the Sony PlayStation 3 SDK. Given that the book's title includes the words "For Games and Graphics" assumptions may arise that this is a book specifically written for the PS3 game developer. This is not the case. The book does not describe tools that are standard in the game development community. Important elements not discussed include the PlayStation 3 developer hardware, Microsoft Visual Studio as the Integrated Developer Environment (IDE), combined with the suite of tools by SN Systems such as the ProDG compiler, debugger, tuner, and target manager tools. Additionally, the libraries provided by the IBM SDK are different from those provided by the Sony SDK in many significant areas.

In fact, the book focuses on Linux as the development operating system of choice, which is uncommon in the game development industry, at least for PlayStation 3 development. But this does explain the choice of software tools covered, as the IBM Cell SDK is only available for Linux. The IDE described is Eclipse, working in tandem with the GNU GCC compiler. The author describes how to install Linux on a standard PlayStation 3 and install the needed software tools. Instead of the SN Systems Target Manager, it is suggested to use PuTTY or WinSCP. Obviously, there is more than one way to skin a cat, and it is interesting to see how it is done.

After a little while I was struck by one amusing fact. All the tools described by the author are open source or are freely available at no cost. The IBM Cell SDK, the GNU compiler, GDB debugger, Eclipse, and finally Linux, all come at a total cost of zero. All that is needed is an off-the-shelf PlayStation 3. The last few chapters in the book address game development lightly, and suggest the use of the open source graphics engine Ogre3D and Collada for game data representation.

Realizing the actual goal and focus of the book, it comes as no surprise that the PlayStation 3's NVidia RSX graphics chip is not discussed. Instead there is a chapter on accessing the Linux frame buffer as a device directly, which I found to be interesting. Usually one is bound to using OpenGL or DirectX. The next chapter is a brief introduction to OpenGL, and how to download and compile the open source implementations called Mesa or Gallium.

In summary, *Programming the Cell Processor For Games, Graphics, and Computation* is an excellent book that tackles a difficult subject admirably. The only point of caution is that not all chapters may relate to what the reader is looking for, depending on preset expectations. In either case, the chapters describing programming the Cell Broadband Engine are very effective and well worth the read.

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

★★★★

TITLE
Programming the Cell Processor For Games, Graphics, and Computation

AUTHOR
Matthew Scarpino

PRICE
$69.99 (hardcover)

PUBLISHER INFORMATION
Prentice Hall
1330 Avenue of the Americas
New York, NY 10019
www.informit.com

Published: October 2008
744 pp.
ISBN-10: 0136008860

www.rtpatch.com

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

NOEL LLOPIS

START PRE-ALLOCATING

And Stop Worrying

WE ALL HAVE THINGS NAGGING IN THE back of our minds. It's never anything we have to worry about this very instant, just something we need to do sometime in the future. Maybe it's changing those worn tires on the car or making an appointment with the dentist about that tooth that has been aching on and off.

Dynamic memory allocation is something that falls in that category for most programmers. We all know we can't just go on allocating memory willy-nilly whenever we need it, yet we put off dealing with it until the end of the project. By that time, deadlines are piling on the pressure, and it's usually too late to make significant changes.

With a little bit of forethought and planning, we can avoid those problems and be confident our game is not going to run out of memory at the most inopportune moment.

ON-DEMAND DYNAMIC MEMORY ALLOCATION

The easiest way to get started with memory management is to allocate memory dynamically whenever you need it. This is the approach many software engineering books consider ideal, and it's often encouraged in computer science classes.

It's certainly an easy approach to use. Need a new animation instance when the player is landing on a ledge? Allocate it. Need a new sound when we reach the goal? Just allocate another one!

On-demand dynamic memory allocation can help to keep memory usage to a minimum because you only allocate the memory that you need and no more. In practice, it's not quite as neat and tidy because there can be a surprisingly large amount of overhead per allocation, which adds up if programmers become allocation-happy.

It's also a good way to shoot yourself in the foot.

Games don't live inside a computer science textbook, so we have to deal with real world limitations, which make this approach cumbersome, clunky, and potentially disastrous. What can go wrong with on-demand dynamic memory allocation? Almost everything!

LIMITED MEMORY

Games, or any software for that matter, run on machines with limited amounts of memory. As long as you know what that limit is and you keep extremely careful track of your memory usage, you can remain under the limit. However, since the game is allocating memory any time it needs it, there will most likely come a time when the game tries to allocate a new block but there is no memory left. What can you do then?

Nothing easy, I'm afraid. You can try to free an older memory block and allocate the new one there, or you can try to make your game tolerant to running out of memory. Both those solutions are very complex and difficult to implement correctly.

Even setting memory budgets and sticking to them can be very difficult. How can a designer know that a given particle system isn't going to run out of memory? Are these AI units going to create too many pathfinding objects and crash the game? It's hard to know until we run the game in all possible combinations. And even then, how do you know it isn't going to crash five minutes later? Or ten? It's almost impossible to know for certain.

If you insist on using this approach, at the very least, you should tag all memory allocations so you have an idea of how memory is being used. You can either tag each allocation based on what system initiated it (physics, textures, animation, sound, AI) or even on the filename where it originated, which has the advantage that it can be automated and should still give you a good picture of the overall memory usage.

MEMORY FRAGMENTATION

Even if you take strenuous pains not to exceed the available memory, you still could run into trouble because of memory fragmentation. For example, you might have enough memory for a new allocation, but in the form of many small memory blocks instead of a large contiguous one. Unless you provide your own memory allocation mechanism, fragmentation is something that is very hard to track on your own—you can't even be ready for it until the allocation fails.

VIRTUAL MEMORY

Virtual memory could solve all those problems. In theory, if you run out of real memory, the operating system swaps out some older, unused pages to disk and makes room for the new memory you requested.

In practice, it's just a bad caching scheme because it can be triggered at the worst possible moment, and it doesn't know what data it's swapping out or how your game uses them.

Games, unlike most other software, have a "soft real time" requirement. The game needs to keep updating at an acceptable interactive rate, which is somewhere around 15 or more frames per second. That means consumers are

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.*

going to make a trip to the store to return your game if it pauses for a couple of seconds every few minutes while it's "making room" for new memory. Relying on virtual memory isn't a particularly attractive solution.

Additionally, a lot of games run in platforms with fixed amounts of RAM and no virtual memory. When memory runs out, things won't get slow and chuggy—they'll crash hard. When the memory is gone, it's really gone.

PERFORMANCE PROBLEMS

There are some performance issues that are relatively easy to track down and fix, usually ones that occur every frame and are happening in a single spot: some expensive operations, a `O(n3)` algorithm, and so forth. Then there are performance problems introduced by dynamic memory allocations, which can be really hard to track down.

Standard `malloc` returns memory pretty quickly, and usually doesn't ever register on the profiler. Every so often though, whenever the game has been running for a while and memory is pretty fragmented, it can spike up and cause a significant delay for just a frame. Trying to track down those spikes has caused more than one programmer to age prematurely. You can avoid some of those problems by using your own memory manager, but don't attempt to write a generic one yourself from scratch. Instead start with some of the ones listed in the references.

`malloc` spikes are not the only source of performance problems. Allocating many small blocks of memory can lead to bad cache coherence when the game code accesses them sequentially. This problem usually manifests itself as a general slowdown that can't be narrowed down in the profiler. With today's hardware of slow memory systems and deep caches, good memory access patterns are more important than ever.

KEEPING TRACK OF MEMORY

Another source of problems with dynamic memory allocation are bugs in the logic that keeps track of the allocated memory blocks. If we forget to free some of them, our program will have memory leaks and has the potential to run out of memory.

The flip side of memory leaks is invalid memory access. If we free a memory block and later access it as if it were allocated, we'll either get a memory access exception or we'll manage to corrupt our own game.

Some techniques, such as reference counting and garbage collection can help keep track of memory allocations, but introduce their own complexities and overhead.

INTRODUCING PRE-ALLOCATION

In the opposite corner is the purely pre-allocated game. It excels at everything that the dynamically-allocated game is weak at, but it has a few weaknesses of its own. But all in all, it's probably a much safer approach for most games.

The idea behind a pre-allocation memory strategy is to allocate everything once and never have to do any dynamic allocations in the middle of the game. Usually you grab the biggest block of memory you can, and then you carve it out to suit your game's needs.

Some advantages are very clear: no performance penalties, knowing exactly how your memory is used, never running out of memory, and no memory fragmentation to worry about. There are some other, more subtle advantages, too, such as being able to put data in contiguous areas of memory to get best cache coherency, or having blazingly fast load times by loading a baked image of a level directly into memory.

The main drawback of pre-allocation is that is more complex to implement than the dynamic allocation approach, and it takes some planning ahead.

KNOW YOUR DATA

For pre-allocation to work, you need to know ahead of time how much of every type of data you will need in the game. That can be a daunting proposition, especially to those used to a more dynamic approach. However, with a good data baking system (see "Inner Product," October and November 2008), you can get a global view of each level and figure out how big things need to be.

There is one important design philosophy that needs to be adopted for pre-allocation to work: Everything in the game has to be bounded. That shouldn't feel too restrictive; after all, the memory in your target platform is bounded, as well as every single resource. That means that everything that can create new objects, including high-level game constructs, should operate on a fixed number of them. This might seem like an implementation detail, but it often bubbles up to what's exposed to game designers. A common example is an enemy spawner. Instead of designing a spawner with an emission rate, it should have a fixed number of enemies it can spawn (and potentially reuse them after they're dead).

"The purely pre-allocated game excels at everything that the dynamically-allocated game is weak at, but it has a few weaknesses of its own. But all in all, it's probably a much safer approach for most games."

POTENTIALLY WASTED SPACE

If you allocate enough data for the worst case in your game, it can lead to a lot of unused data most of the time. That's only an issue if that unused data is preventing you from adding more content to the game. We might initially balk at the idea of having 2,000 pre-allocated enemies when we're only going to see 10 of them at once. But remember that each of those enemies is only taking 256 bytes and the total overhead is 500KB, which can be easily accommodated in most modern platforms today.

Pre-allocation doesn't have to be as draconian as it sounds. You could relax this approach and commit to having each level pre-allocated and never having dynamic memory allocations while the game is running. That still allows you to dynamically allocate the memory needed for each level and keep wasted space to a minimum. Or you can take it further and pre-allocate the contents of memory blocks that are streamed in memory. That way each block can be divided in the best way for that content and wasted space is kept to a minimum.

REUSE, RECYCLE

If you don't want to pre-allocate every single object you'll ever use, then you can create a smaller set and reuse them as needed. This can be tricky though. First of all, it needs to be very much specific to the type of object that is reused. Particles are easy to reuse (just drop the oldest one, or the ones not in view), but enemy units or active projectiles will be more difficult. It's going to take some game knowledge of those objects to decide which ones to reuse and how.

It also means that systems need to be prepared to either fail an allocation (if your current set of objects is full and you don't want to reuse an existing one), or they need to cope with an object disappearing from one frame to another. That's a relatively easy problem to solve by using handles or other weak references instead of direct pointers.

Then there's the issue that reusing an object isn't as simple as constructing a new one. You really need to make sure that when you reuse it, there's nothing left from the object it replaced. This is easy when your objects are just plain data in a table, but can be more complicated when they're complex C++ classes tied together with pointers. In any case, you can't apply the "resource acquisition is initialization" (RAII) pattern, but it doesn't seem to be a pattern very well suited for games, and it's a small price to pay for the simplicity that pre-allocation provides.

SPECIALIZED HEAPS

Truth be told, a pure pre-allocated approach can be hard to pull off, especially with highly dynamic environments or games with user-created content. Specialized heaps is a combination of dynamic memory allocation and pre-allocation that takes the best of both worlds.

The idea behind specialized heaps is that the heaps themselves are pre-allocated, but they allow some form of specialized dynamic allocation within them. That way you avoid the problems of running out of memory, or memory fragmentation globally, but you still can perform some sort of dynamic allocation when needed.

One type of specialized heaps is based on the object type. If you can guarantee that all objects allocated in that heap are going to be of the same size, or at least a multiple of a certain size, memory management becomes much easier and less error prone, and removes a lot of the complexity of a general memory manager.

My favorite approach for games is to create specialized heaps based on the lifetime of the objects allocated in them. These heaps use sequential allocators, always allocating memory from the beginning of a memory block. When the lifetime of the objects is up, the heap is reset and allocations can start from the beginning again. The use of a simple sequential allocator bypasses all the insidious problems of general memory management: fragmentation, compaction, leaks, and so on. (See the code in http://gdmag.com/resources/code.htm for an implementation of a Sequential Allocator class.)

The heap types most often used in games are:

- **LEVEL HEAP.** Here you allocate all the assets and data for the level at load time. When the level is unloaded, all objects are destroyed at once. If your game makes heavy use of streaming, this can be a streaming block instead of a full level.
- **FRAME HEAP.** Any temporary objects that only need to last a frame or less get allocated here and are destroyed at the end of the frame.
- **STACK HEAP.** This one is a bit different from the others. Like the other heaps, it uses a sequential allocator and objects are allocated from the beginning, but instead of destroying all objects at once, it only destroys objects up to the marker that is popped from the stack.

WHAT ABOUT TOOLS?

You can take everything I've written here and (almost) completely ignore it for tools. I fall in the camp of programmers who consider the runtime as a totally separate beast from the tools. That means that the runtime can be lean and mean and minimalistic, but I can relax and use whatever technique makes me more productive when writing tools—I will allocate memory any time I want, and I will use complex libraries like STL and Boost, if I feel the need. Most tools are going to run on a beefy PC and a few extra allocations here and there won't make any difference.

Be careful with performance-sensitive tools, though. Tools that build assets or compute complex lighting calculations might be a bottleneck in the build process. In that case, performance becomes crucial again and you might want to be a bit more careful about memory layout and cache coherency.

On the other hand, if the tool you're writing is not performance sensitive, you should ask yourself if it really needs to be written in C++. Maybe C# or Python would be a better language if all you're doing is transforming XML files or verifying that a file format is correct. Trading performance for ease of development is almost always a win with development tools.

Next time you reach out for a `malloc` inside your main loop, think about how it can fail. Maybe you can pre-allocate that memory and stop worrying about what's going to happen the day you prepare the release candidate. ⁙

RESOURCES

DOUG LEA'S MEMORY ALLOCATOR
http://g.oswego.edu/dl/html/malloc.html

LOKI FIXED SIZE ALLOCATOR
http://www.aoc.nrao.edu/php/tjuerges/ALMA/Loki/html/a00674.html

THINK SERVICES
GAME GROUP

NETWORK OF SITES

GAMASUTRA
The Art & Business of Making Games

GAME CAREERGUIDE

indiegames.com

WORLDS IN MOTION.BIZ

GAMES ON DECK

FG FINGER GAMING

GAME SET WATCH

gamedeveloper The Leading Game Industry Magazine

EVENTS CALENDAR 2009

MARCH 23–27, 2009

GDC 09

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

MARCH 27, 2009

GAME CAREER SEMINAR

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

MAY 12–13, 2009

GDC Canada

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

AUGUST 17–19, 2009

GDC 09 Europe

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

SEPTEMBER 14–18, 2009

GDC 09 Austin

Game Developers Conference® Austin
Austin Convention Center, Austin, TX
www.gdcaustin.com

OCTOBER 11–13, 2009

GDC 09 China

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

FOR UPDATES AND MORE INFORMATION ON OUR EVENTS VISIT:
www.tsgamegroup.com

STEVE THEODORE

OVER AND OVER AND OVER AND OVER

The Misadventures of Tiled Textures

HARDWARE COMES AND GOES, BUT THE struggle to stuff our endless imaginings into a finite set of RAM chips is with us always. VRAM is like time and money: It's precious and there's never enough of it.

Many of the gimmicks that distinguish game artists come from the fact that, unlike our cousins in the offline graphics world, we are always trying to stretch that limited runtime memory budget. We rely on all sorts of tricks—UV mirroring, texture compression, and vertex lighting—but in the end the biggest single tool we use for managing memory is also the humblest. It's that granddaddy of all the tricks in the game artist's carpet bag: texture tiling.

And as annoying as texture repeats can be, they're going to be with us for a while. All those impressive heaps of VRAM in a shiny new PlayStation 3 are about enough to pave a thousand square yards at one pixel per inch. That's not a very high resolution or a very big area; players can cover it in about three minutes. (Though three minutes might seem like a long time if all they could see was a textured ground plane.)

Let me put it plainly: Repeat textures are pretty lame. Over-repetition just vacuums all the goodness out of even the slickest texturing job. By the twentieth replay of the exact same "natural-looking" rock pattern or "distressed" wall texture, the visual itself has ceased to be an image and has turned into an icon, or worse, a meaningless and forgettable wallpaper.

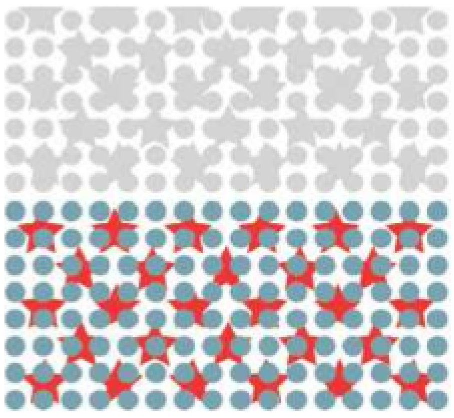


FIGURE 1 The interaction of mismatched repeats helps to obscure the repetitive nature of tiling (top). However, when there are strong contrasts of form or color, the eye is very good at penetrating the overlay and disentangling the two patterns (bottom).

Luckily, the current generation of graphics hardware actually offers a little hope to artists who are sick of seeing the same pixels over and over again. Now that pixel shaders are common, it's possible to get a little more bang out of this most venerable of graphics techniques. Let's consider some ways to make tiling textures suck a little less.

PAINT BY PRIME NUMBERS

The simplest way to finesse tiling textures is to combine them for greater variety. This trick has been around for a long time, but the rise of fast modern pixel shaders makes it much more interesting and flexible. If repetition is so deadly, it would make sense to composite textures to make more unique looks over larger areas.

The simplest option is to layer two repeating textures on top of each other. If they don't repeat at the same rate, you've technically made the visual equivalent of a much larger texture with a much longer repeat rate. (Though, as simple blends they are rarely enough to really create the illusion.) If you pick your tiling sizes well, the next exact repetition could be quite far off.

For example, if you overlay an image that tiles once every 11 feet on one that tiles once every 29 feet, the repeat size of the combined texture is 319 feet (11x29=319). You have to make sure the textures repeat at quite different rates and that the rates don't share a common multiple. A 12-foot texture repeating against a 24-foot texture is just a new 24-foot texture, because the second texture's integer is a multiple of the first's. If you haven't forgotten your high school math, stick to prime numbers, like 11 and 29, when you can.

Unfortunately, this trick by itself doesn't take you very far. Blending two textures together will prevent the repetition of the exact same pixel images, but it won't beat the human eye's uncanny ability to pick out patterns. As you can see in Figure 1, the two overlaid patterns can follow the prime number repeat rule and still look like two overlaid patterns rather than a single unique texture.

FRISKET CASE

The problem is that the human visual system is as sensitive to shape as it is to simple repetition. Overlaying two textures helps disguise the frequency of repetition, which slows down our ability to pick up patterns—but as long as the pattern contains repeated copies of the same shapes, our eyes will eventually ferret out the underlying pattern. Once they do catch it out, the texture is effectively neutered.

You can minimize this kind of problem by eliminating strong features from

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****.*

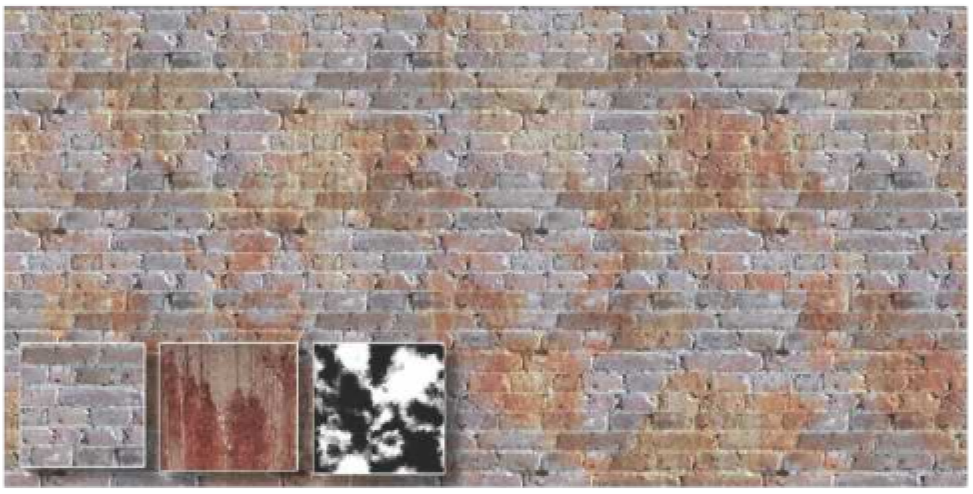


FIGURE 2 Combining overlay textures with masks to break up the outlines of strong features helps hide repetitions much more effectively than simple overlays. The large image is made from scaled, tiled, and overlaid combinations of the three smaller images. Look closely to see that the brick texture repeats 16 times, but the combination of overlay and masks at different repeats obscures the repetition.

your textures. Mushy patterns are much harder to decipher. Unfortunately they're also pretty dull to look at, and few artists want to settle for a game made entirely of formless and noisy textures.

Here's where modern graphics come to the rescue, at least in a small way. As we've noted before (see "Clone Wars," October 2008), modern pixel shaders can do many of the same Photoshop tricks every artist knows by heart. Some of those old tricks can be easily extended to help make tiled textures a bit more tolerable.

Pixel shaders can reproduce a lot of the familiar Photoshop blend modes, which means you can use nonlinear effects like the old standby "soft light" filter to get different effects out of a single overlay texture. But you can also use the pixel shaders to create masks on the overlaid texture, which lets you not only add another source of visual variety, but also break up some of the landmarks in the overlay texture. If necessary, you could add alpha information in the base texture as a kind of frisket to protect parts of the base that should not be over-written by overlay textures.

Combining blend modes and masks is a very powerful way to enliven repeating textures, as shown in Figure 2. Naturally, none of this is a real substitute for carefully handcrafted textures, but it beats the pants off tiling the same texture a zillion times, and the costs are relatively slight. The example shown here requires just two extra texture lookups and some cheap pixel math. If a trick like this lets you hit your visual marks without storing a larger conventional texture, it can be worth it. Best of all, most artists can visualize the results fairly well based on their Photoshop experience, even if they don't grasp the subtleties of HLSL, making it a trick that's likely to be well used.

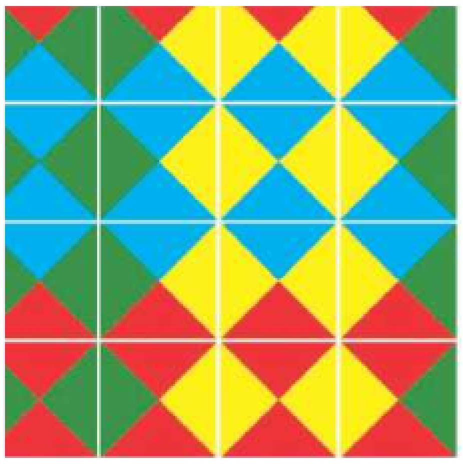


FIGURE 3 In this 16-tile Wang set, the colors along each edge indicate which tiles can be placed together. The arrangement in this image is designed to provide good texture filtering in a hardware implementation.

WHO WANTS SOME WANG?

If overlaying alone still doesn't satisfy your hunger for endless fields of textures on an affordable budget, you may still be in luck.

In the last few years, academic graphics researchers have been playing with systems for covering an infinite space with a finite amount of content but no repeats. The most common strategy is based on the little colored squares shown in Figure 3, which go by the name of Wang tiles. The name is a bit of a backhanded complement. Dr. Hao Wang came up with the idea of tiles with matching edges in the 1960s while trying to prove that tiling an infinite plane without repetitions was impossible. Unfortunately for him (but luckily for us), his theory was proved wrong a few years later. But the name stuck.

Dr. Wang's metaphorical tiles are squares with colored edges (refer to Figure 3). Wang's method was to create patterns by laying the tiles out one at a time, choosing the new tiles with edge colors that matched those you've just laid down, so a red edge on the left matches a red edge on the right, and so on. As long as you lay the tiles in row-column order, you only need to match two colors at any given time: the colors "above" and the color to the "left" of your next tile. As you can see in the example, our set contains four tiles for each possible combination of "above" and "left." That, by the way, is where Wang's theory broke down. Because there are two possible tiles that fit the next slot, you can choose the next one randomly—and if it is truly random, then by definition it doesn't repeat.

In the academic world, the hot trick is to analyze a picture and create a set of these tiles automatically, which allows you to create an infinite, non-repeating version of the original image (see Figure 4). You can see why this gets the hearts of tiled-texture authors to beat faster.

Although we're not quite able to extract infinite textures on the fly in real time, a simplified version of the technique can produce finite, but still impressively large, non-repeating textures. The technique We'll discuss here is from Li-Yi Wei's "Tile-Based Texture Mapping" published in *GPU Gems 2* (Addison-Wesley Professional, 2005), in case you want to leave it out on the coffee table for your coders to see. While it's not the only approach to the problem it's an easy one for most artists to understand and author for.

Figure 4 shows how the tiling works. The texture used by the Wang tile pixel

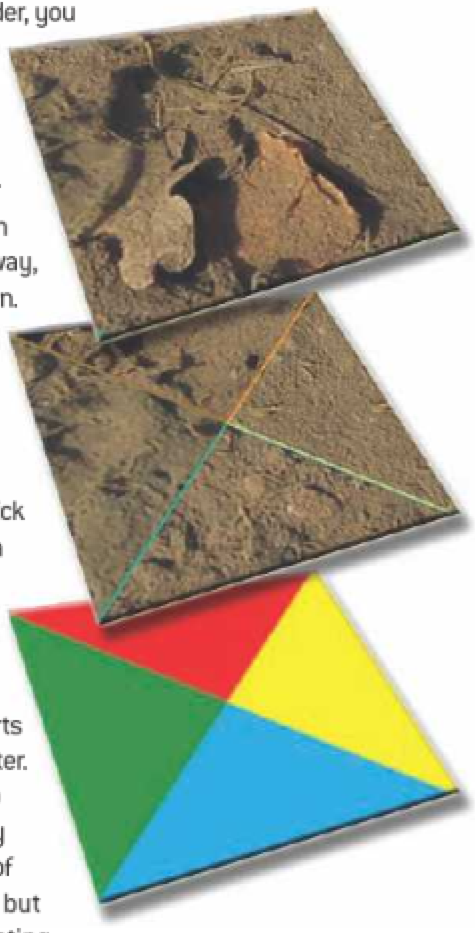


FIGURE 4 When setting up a texture for a Wang-tile-like reshuffling, each color is filled with a different repeating pattern stamp at one-quarter the size of the whole texture. Then the center of each tile can be edited to add visual variety and blend transitions between the stamps.

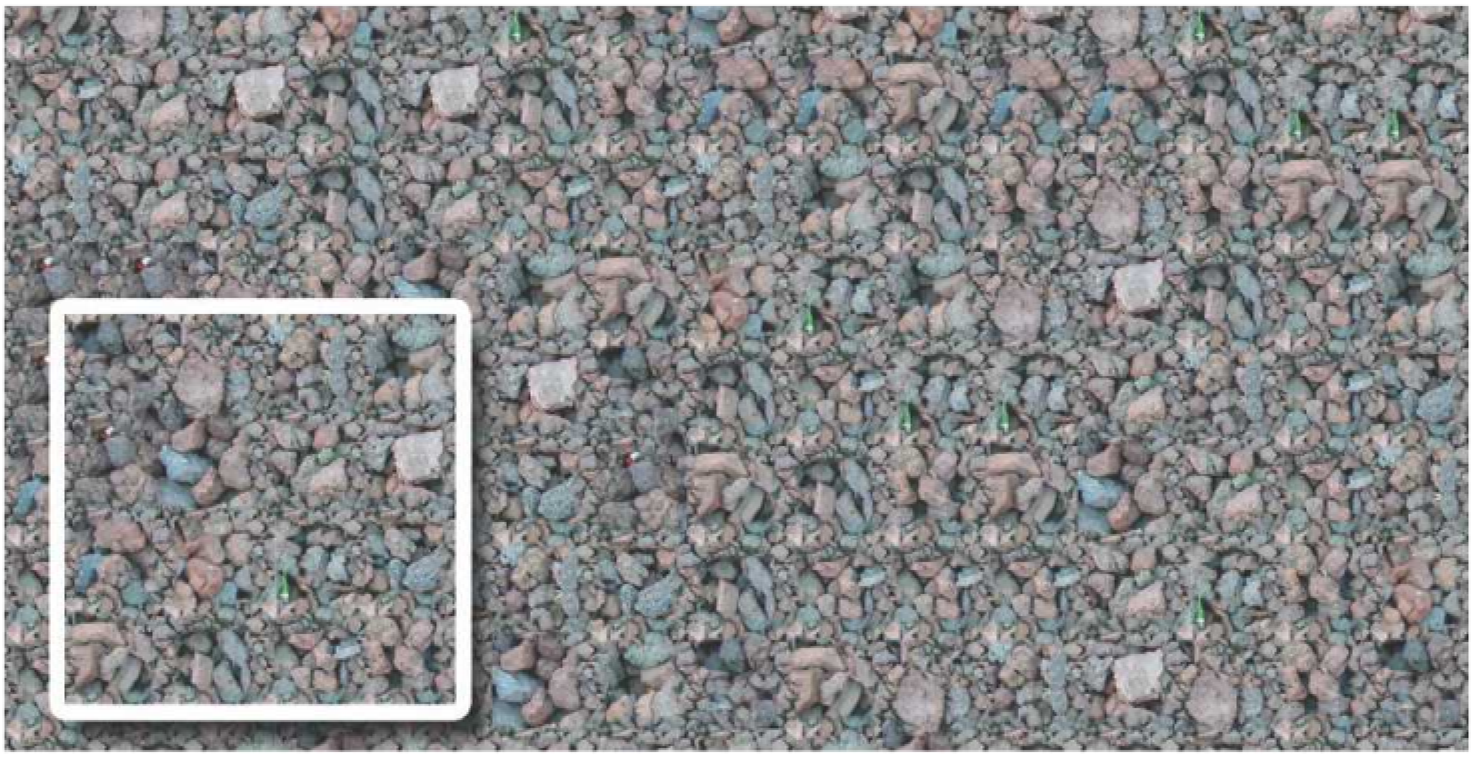


FIGURE 5 In this Wang tiled texture, note that the rhythm is very subdued, but the obvious landmarks are seen multiple times. This combination of repeating pieces without repeating patterns is the signature look of Wang tiling.

shader consists of 16 tiles. The edges of each tile are labeled either red or green. Edges of the same color should contain matching textures, so that, for example, any tile with a red top could be placed under any tile with a red bottom and show no seam. The shader can then randomly create a set of tiles by using the edge-matching rule. At runtime, you'll get an endless, randomized mosaic of your original texture, and if you did a good job with the edge-matching, it should be seamless. (Strictly speaking it's not "endless." The example from *GPU Gems 2* is to generate a set of tiles once and then store it to a lookup texture, which is a lot faster than recreating the position of each tile in the pixel shader every frame. Therefore, it's not technically infinite, just pretty damn big.)

To make this work, you need to pay some attention to how the tiles are laid out. Unlike the offline version, this tactic can't magically create tiles from any old image. Luckily, setting up a Wang tile texture is pretty simple if you use a pattern like the one in Figure 3. As you can see, the texture is divided up into 16 tiles. They're laid out so that each tile blends seamlessly with its neighbors; that's important because it helps the graphics hardware get the right color at lower mip levels.

It looks daunting, but managing the seams is actually easy for any tiled-texture veteran. Make four small tiling textures the old fashioned way, with offset filter and clone stamp in Photoshop. Each should be at one-fourth the size of the original. Next, select each of the colors in the pattern in turn and use Photoshop's "paste into" feature. The result will be a sort of quilted image with correct matches across the edges of all 16 tiles (refer to Figure 4). Of course, the interior of each tile shows seams where the four edge sections come together, so you'll want to add some unique texture to the center of each tile, taking care to not stray across the tile boundaries.

If all goes well, you should be able to get results like those in Figure 5. As you can see, the Wang-ified texture can be extended pretty far before the repetitions really jump out at you.

Make sure the most noticeable feature of each tile is not dead center to avoid a mechanical gridded look. You can also break up the grid by having some of your edge sets include high contrast elements that breach the grid boundaries. And, you can use a larger tile set with more edge colors, which will produce even less regular-looking results.

ASK NOT WHAT WANG CAN DO FOR YOU

As you can see from the results, the technique is cool, but it does have some obvious limitations. It's not magically generating more texels than you had before, it's just shuffling and reshuffling bits of your existing texture to help postpone viewer boredom. It works nicely for images with no overarching structure, visually noisy subjects like grass or masonry. On the other hand, subjects with large sharp features are likely to look stilted.

However, the main advantage and the big drawback are the same. By chopping the image into reusable bits, Wang tiles break down the eye's ability to decipher patterns, which is great for the kind of unobtrusive background textures we often need to tile for video games, but not so hot if you're trying to make a carefully calibrated artistic point.

The fact is all these methods are useful, but none is really emotionally satisfying. For the immediate future (at least until the Larabee versus GPGPU war produces some advances in real-time procedural texturing), we game artists are going to be recycling every little scrap of runtime memory we can.

If you're of a gloomy turn of mind, you might compare us to those Depression-era folks who had to reuse tea bags to make ends meet. Or, you could tell yourself that tiling textures is like being a virtuous, recycling-friendly, sustainability consultant. Either way, you're going to be doing it for a while, so make the most of it.

Jobs on

GAMASUTRA

The Art & Business of Making Games

The Leading Professional Game Source for Job Seekers and Employers

Access the Web's Largest Game Industry Resume Database and Job Board

Take Control of Your Future Today!

Log onto www.gamasutra.com/jobs

THINK SERVICES
A DIVISION OF UNITED BUSINESS MEDIA LLC

DAMION SCHUBERT

FOCUSING YOUR INNOVATION

THERE'S AN URBAN LEGEND WHICH states that in the early days of the space program, NASA determined ball-point pens wouldn't write in space. They pressed their engineers to design a pen that could, and after months of time and millions of dollars in research, they finally did. The Russians, when presented with the same problem, simply had their cosmonauts carry a pencil.

As a veteran in the industry, this story is compelling because it speaks to a problem that plagues game development teams throughout the industry—mismanaged innovation. We have finite resources. There is limited time, a limited budget, and limited engineering cycles—and these resources dwindle as the realities of development occur and unexpected problems arise. Given these challenges, even well-funded teams stumble, resulting in failed innovations and often doomed projects.

FAILURES OF INNOVATION

Failures in innovation can take multiple forms. Most designers worry about innovating too little, which is a good concern to have. Innovation is important to gaming—in many ways, we are a research-and-development industry, and novelty is hugely important to breaking into a core market that is by now deeply jaded. Clones and me-too products are unlikely to score with the public consciousness and break into megahit status, unless they are themselves sequels to megahits.

Most designers know this, but they still fail in the pursuit of innovation, stumbling into all-too-common design pitfalls that plague both shipped products and those that get cancelled before they see the light of day. Common mistakes are innovating too much, mistaking content for innovation, going too weird, failing to execute, and failing to innovate in a direction of value to the market.

TOO MUCH INNOVATION

Students I speak to always seem surprised when I complain of design teams that try to innovate too much. This reaction is natural for young designers—it is their impulse to look at every feature in the game and ask how every possible feature can be improved, if not replaced entirely. How could too much innovation necessarily be bad?

But often, it is—even one truly ambitious new feature can be difficult to build, balance, test and deliver at a high level of quality. Attempting to do too many risky things at once reduces the likelihood that any of them will be executed as well as they could.

In a recent talk, Cliff Bleszinski described how the original GEARS OF WAR feature list was an endless wishlist of cool, but expensive issues. They soon realized that attempting to do too many risky things at once reduces the likelihood that any of them could be executed to their full potential. Focusing and trimming the feature list to highlight cover mechanics and light squad combat allowed them to do both features well—and ultimately earn multiple Game of the Year awards. The customers will likely never know what features didn't make it.

Too much innovation can make the game harder to sell, as it can cloud your marketing message. New features that are unpopular, unfinished, or just not fun can dominate reviews and online chatter about your game, stealing focus from the ambitious game elements that your team actually succeeded in delivering.

FAILURE TO EXECUTE

Creating game systems that are innovative is hard. Areas of true innovation are typically full of engineering risk and design uncertainty. They require tons of iteration and playtesting to get right and are almost impossible to schedule accurately.

Having one truly innovative feature on your schedule makes life difficult—having multiple can make it a nightmare. This is most true when the new features interact with each other directly—the cost of adding these features can increase exponentially as designers, programmers, and quality assurance have to deal with unexpected edge cases related to how these systems mix.

Even one innovative feature can wreck a schedule if it is heavily content-driven. A system in which a player can only use stealth in the dark, for example, requires world builders to create maps and levels to specifications that arise from iteration and playtesting the system. If that system comes online too late, content built early may have to be completely rebuilt or, in worst-case scenarios, be discarded entirely.

One typical result I see when innovation is poorly managed is that when schedule realities hit home, all aspects of the game end up getting shipped underdeveloped. Even in the unlikely case where most of the bugs and edge cases are found and handled, it's unlikely that the innovative feature will be able to get the iteration time and polish needed to truly shine. Then your game winds up in the bargain bins—but if you're (un)lucky, maybe designers at another company will see your idea, figure out how to steal it, and do it right in their own game.

DAMION SCHUBERT is the lead combat designer of 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.

GOING OFF THE DEEP END

Sometimes, designers try to take designs in new, interesting, and altogether weird directions. This can be as simple as changing the core UI paradigm of the genre, such as reversing what mouse buttons do, or changing the genre's standard hotkeys. Or it can be more substantial, for instance adding a new feature to the game that doesn't actually improve how it plays, but just makes it different for the sake of designers calling themselves innovative.

Such changes often present a learning problem for the player—if your genre has a strong market leader, then that genre leader has effectively taught people how to play your game before they even pick up a copy. Games are artificial environments, and players must frequently get over the unfamiliar and uncomfortable before they get what the game is all about. Changing core genre conventions without good cause simply makes your game a more alien experience to fans of the genre.

Anytime a major innovation is proposed for a game design, a lot of questions should be asked: Will the feature actually be seen as better by fans of the genre? Will the feature confuse people expecting the gameplay of the market leader? And will this feature blend gracefully with the game's other features into a coherent whole?

One mistake common to MMOs in particular is innovation that is more interesting to academics and designers than it is to the players. I know of a team that wanted to allow players to set other players' houses on fire, in the name of creating "interesting social dynamics." Design innovations should always be player-centric, with a real focus on what impact they will have to all of your target markets. A new feature that pleases your hardcore but alienates any newcomers, for example, is probably best left on the cutting room floor.

Ralph Waldo Emerson once said, "Build a better mousetrap, and the world will beat a path to your door." The trick is defining "better"—what makes a better mousetrap depends on what the rodent-hunting customer finds valuable. It's the designer's role to identify the top problems the customer has and focus his energies on finding a design response to those issues. At the end of the day, it's not successfully innovative unless it's better.

MISTAKING CONTENT FOR INNOVATION

Laundry lists of content have long been seductive bullet points for the back of the box of any game, perhaps because it's so easy—if the market leader has 10 races, surely 12 or 15 races will be better. This mentality permeates almost every genre. RPGs are the worst offenders—with race, class, creature, spell, quest, and zone counts being common—but they are far from alone in this regard. RTS games like to boast about unit counts and the number of factions. Fighting games love to sell numbers of combatants and arenas. For a while, FPSs would sell their weapon counts, or would add dual-fire modes to effectively double the weapon count.

However, more isn't necessarily better. The market leaders in almost every genre still have a relatively small, constrained set of races, classes, combatants and/or weapons. Quality is the key—the more classes and races you have, the more combinations you have to balance. Further, this problem compounds itself in player vs. player arenas—each character you add to a fighting game increases the variables to balance against exponentially.

At the end of the day, more of the same isn't true innovation—at least not on its own. In all the above examples, players are eventually going to choose a favorite class, a favorite army, or a favorite gun, and tailor their own game experience in that direction. Designers are almost always better off ensuring that each individual choice the players could make is as rich and interesting as possible. Doing less also ensures that those choices which are innovative can be properly balanced and allowed to shine.

FOCUSING YOUR INNOVATION

All the above should not be mistaken for a plea to innovate less—the industry needs better, smarter games to move forward, and you will need new cutting-edge features in your products in order to be competitive. Games that fail to innovate are, in general, destined for the bargain bin and general ignominy. Designers have to take chances, but you can't take on too much risk in your project. Games need highly-targeted innovation, ensuring that you are focusing your energy on features that really matter.

Innovate, but identify features that are high in demand within your core market. Try new things, but prototype and iterate on them early, especially if they have high content demands. Don't innovate too much. But do ensure the features you choose to innovate on are well-executed and polished to a glossy sheen.

A lot of designers think that gamers want you to show them something new. They don't—they want you to show them something better. Good designers understand that innovation is inherently risky—ideas are cheap, executing on them is hard, and managing that risk is often the difference between a project succeeding and it collapsing under its own weight. It may not feel natural for designers to focus on one or two key gameplay features with home run potential and discard the rest, but designers that can do so almost always make better games.

"Anytime a new, major innovation is proposed for a game design, a lot of questions should be asked."

learn
network
inspire

GDC Canada

Vancouver, BC

May 12–13, 2009

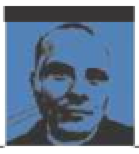
Game Developers Conference® Canada

Vancouver Convention & Exhibition Centre

THINK SERVICES
A DIVISION OF UNITED BUSINESS MEDIA LLC

REBOOT

GDC-Canada.com



JESSE HARLIN

GENERATION XX

FOR YEARS, ONE OF THE BIGGEST TOPICS in the game industry has been female players. At first the discussion centered on how best to attract women to video games in general. Now it seems to mostly be about how game genres, markets, and companies are evolving to meet the interests and needs of female gamers. But take a quick look around your own audio department and something becomes immediately apparent: There are still very few women in game audio.

This month I spoke with sound designers Pamela Aronoff (CRYSIS, DEVIL MAY CRY series), Fryda Wolff (EVERQUEST I & II), and Kristen Quebe (FABLE 2, F.E.A.R.) about their experiences in the industry, their advice to young female sound designers, and their thoughts as to why there aren't more women in game audio.

BUILDING BLOCKS, ROAD BLOCKS

All three sound designers found their ways into the industry with familiar stories. Aronoff was studying music in college when she became fascinated by her recording gear and the "socialization of media" that games represented. For Quebe, "Games came before audio. I practically grew up in an arcade playing games like ELEVATOR ACTION, TEENAGE MUTANT NINJA TURTLES ARCADE, and my all-time favorite, GALAGA."

Quebe's projects in college often focused on interactive audio environments, which caught her instructors' attention and led to an internship at a game company. "At that point my jaw dropped. 'You mean you can make sounds for video games for a living?' The option to take two of my greatest passions and turn them into a career felt like a dream come true," Quebe says.

Wolff similarly turned her passion for gaming into an in-road to game audio. Originally interested in college sound design programs, Wolff parlayed a position as a game master at Sony Online Entertainment into a voice implementation position on EVERQUEST II. "Except that after EQII shipped," she explains, "I didn't have much to do. I begged one of the resident lead sound designers to give me something to do—me, the brat with no prior experience. Thankfully, he had this let's-see-what-she-can-do attitude. He kept throwing tasks at me, and I learned my way around editing and mixing software on the fly."

TECHNO-FEM-PHOBIA

Unfortunately, sexism isn't an unheard of issue. "I have a real passion for audio," says Aronoff, "and I think because of that, initially I was blind to the institutional or cultural sexism. When I got my first job I think if anything it helped that I was female. I think they thought, 'Audio girl. Cool!' Over time, I've definitely seen my share of issues on the topic [of gender imbalance]. I hate it when I get around guys while there is a tech problem and they exclude me from the trouble-shooting conversation. I've been in audio since I was a teen. I have mad trouble-shooting skills!"

Wolff mentions this "boys' club" mentality, as well. "If there's an opening for a project, a longtime audio pro is more likely to call his buddy from 15 years back than take a chance on a younger female who is newer to the audio community. It's difficult to get one's foot in the door, particularly as a woman, when there are so very few audio ladies who 'go way back.'"

Wolff continues, "I think the environment the old boys' club creates is subconsciously hostile toward women who will literally never be one of the guys, regardless of their talent or experience. That just means women will have to work a little bit harder and be a little bit tougher to prove themselves deserving of being part of the club."

For Quebe, of the four and half years she has been in the industry, she says she has only met four other women total who work in audio production in video game development studios. "I think that is changing though," she adds, "as we branch out and make games that have a larger appeal to women."

GETTING THERE

Craning one's neck to see more change is common among audio designers. "The industry has been making huge strides," says Quebe. "Women have forums and groups to talk to other women in the industry. With that said, games have to have a broader appeal to women. I think more women would enter into this field if they had more exposure to it. If we can get more women interested in gaming, then that can lead to more women interested in careers within the game industry."

Aronoff taught audio at Loyola Marymount University for a few years. "When I asked what people wanted to do when they graduated, the girls in the class said either management or agent. When I asked why, they said they were intimidated. I think girls get more easily psyched out and have a harder time believing they have what it takes to make it in the audio tank."

When it comes to advice for young women entering the field, the audio designers I spoke with agreed that a wealth of good information is out there, mostly online. "The internet has become an unrivaled resource," says Wolff. "Snoop the networks of working audio pros. See who they know, find out what they do, ask them questions, and learn whether it's right for you."

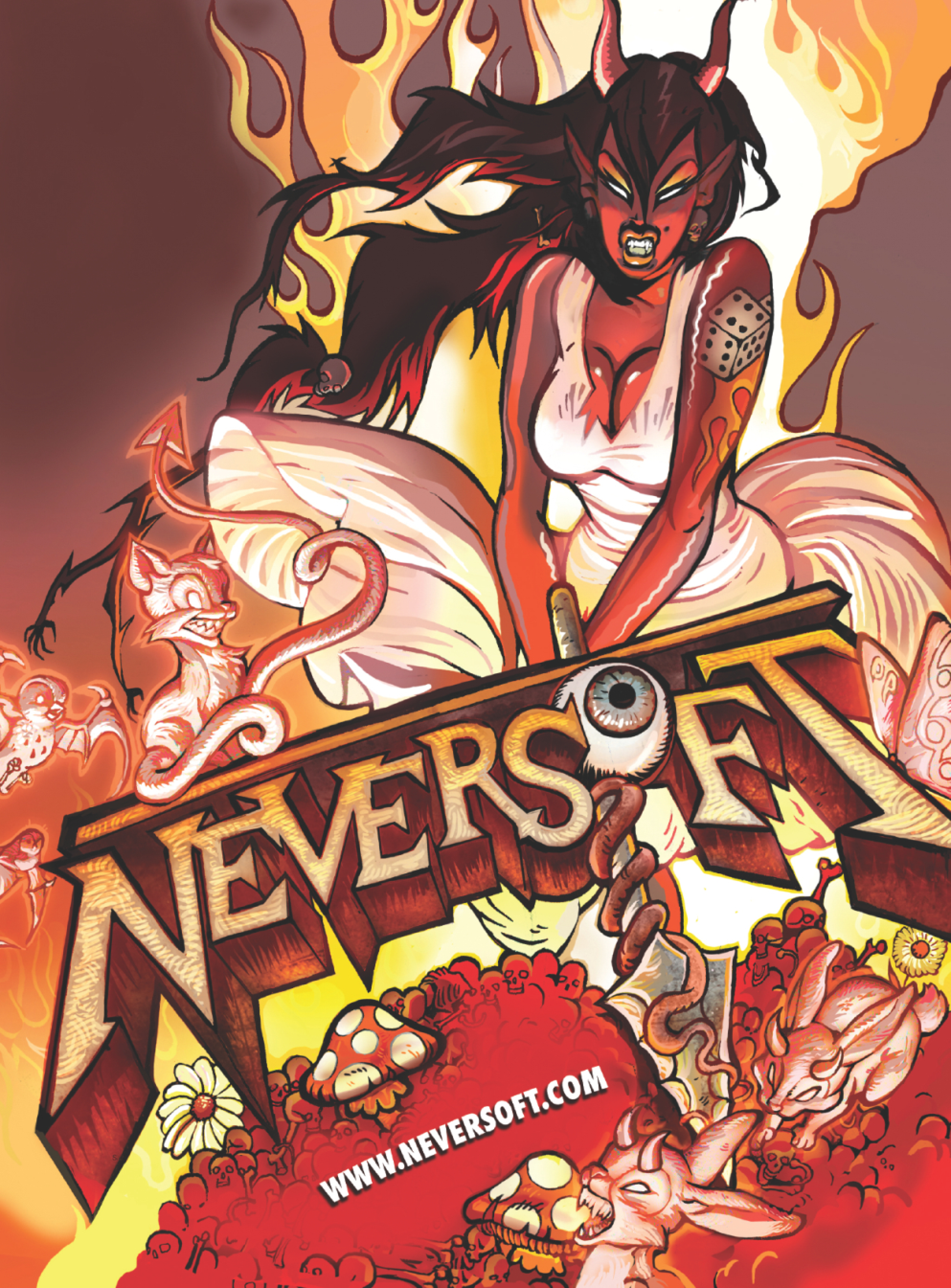
"It's hard to get your foot in the door," Quebe says, "but it is possible with enough dedication and hard work. Get involved with some of the women in game groups. These can provide great insight from women in the industry as well as provide a strong support system." ⌘

RESOURCES

WIGI
www.womeningamesinternational.org

IGDA WOMEN'S SIG
www.igda.org/women

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.


NEVERSOFT
WWW.NEVERSOFT.COM

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

NOW HIRING

WWW.LITH.COM

MONOLITH™

ACTIVISION®

GREAT GAMES
START WITH
GREAT PEOPLE

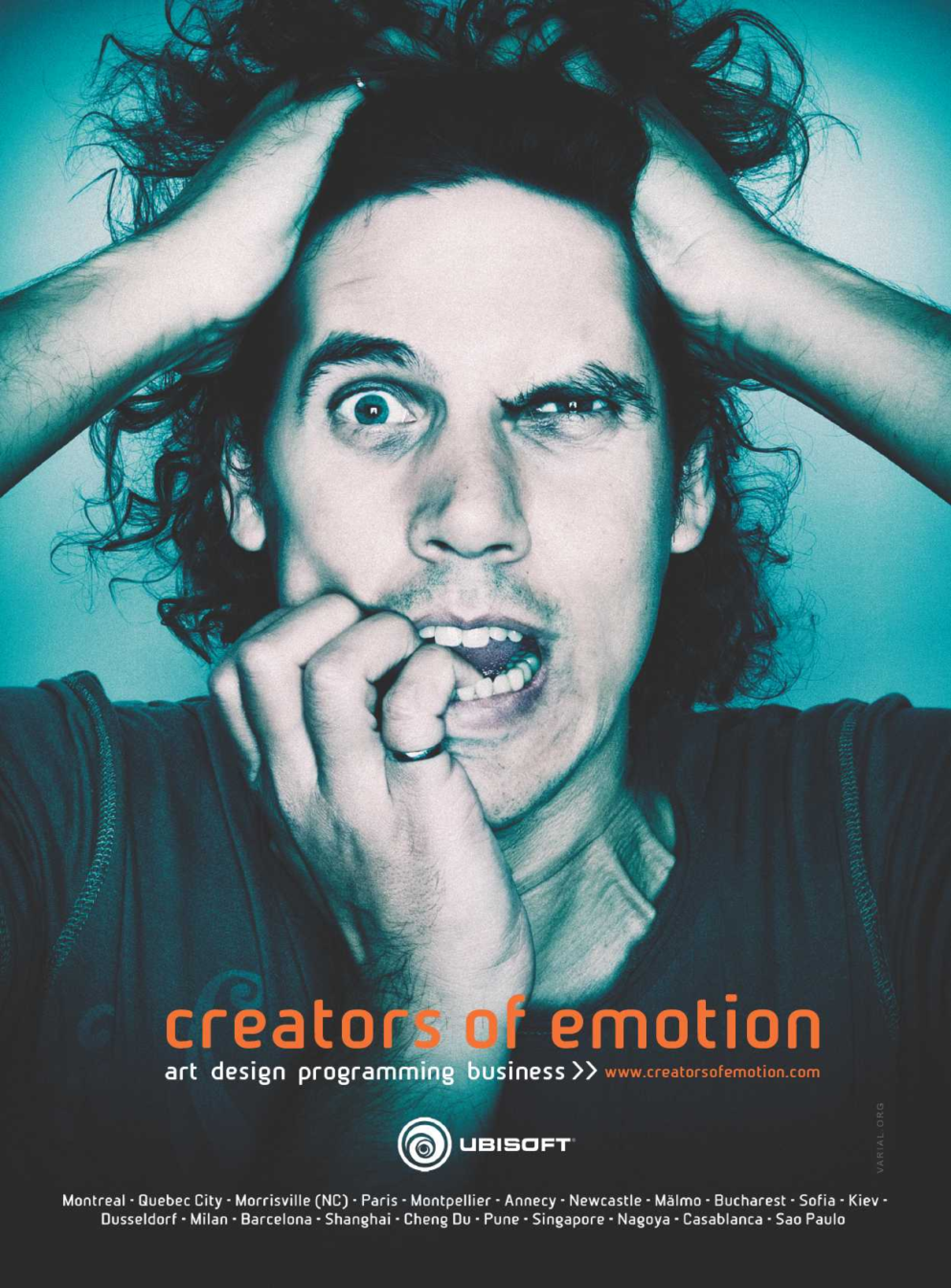
ARE YOU READY TO EXPLORE THE MOST SOUGHT AFTER JOBS IN ENTERTAINMENT?

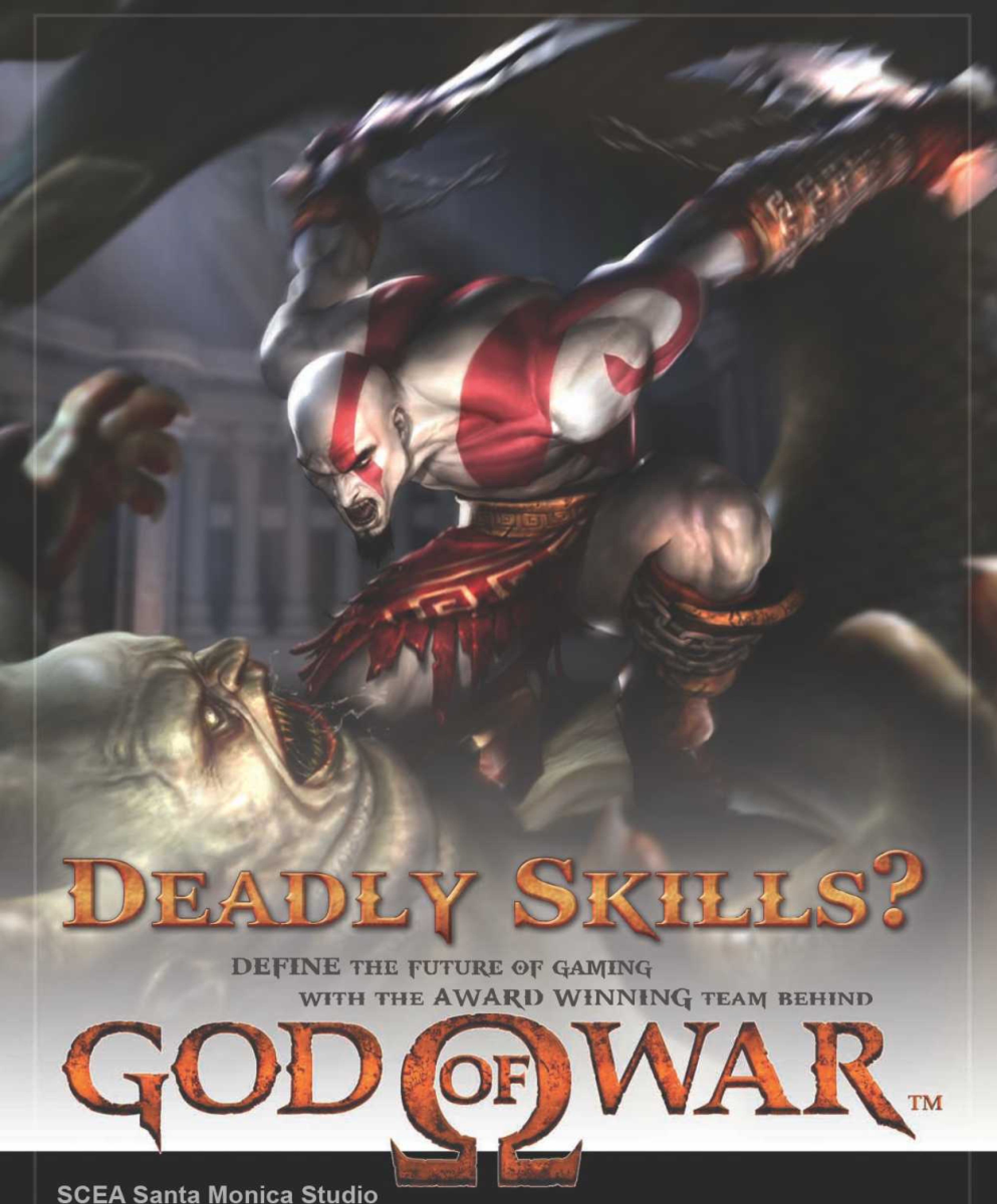
COME VISIT US AT GDC BOOTH #510

ACTIVISION MINNEAPOLIS · SHABA GAMES · BIZARRE CREATIONS · RAVEN · BEENOX · UNDERGROUND · Vicarious Visions · RADICAL ENTERTAINMENT · FreeStyleGames

redoctane · treyarch · Luxoflux · NEVERSOFT · infinity ward · Toys For BOB · DEMONWARE · HIGH MOON STUDIOS · BUDCAT

Visit our site: activision.com


creators of emotion
art design programming business >> www.creatorsofemotion.com
UBISOFT
VARIAL.ORG
Montreal · Quebec City · Morrisville (NC) · Paris · Montpellier · Annecy · Newcastle · Mälmo · Bucharest · Sofia · Kiev ·
Dusseldorf · Milan · Barcelona · Shanghai · Cheng Du · Pune · Singapore · Nagoya · Casablanca · Sao Paulo


DEADLY SKILLS?
DEFINE THE FUTURE OF GAMING
WITH THE AWARD WINNING TEAM BEHIND
GOD OF WAR™

SCEA Santa Monica Studio

is looking for: programmers . designers . artists . animators . producers

Apply online: http://www.us.playstation.com/Jobs

SONY COMPUTER ENTERTAINMENT

PLAYSTATION 3

ACCESS IT IN REAL TIME

gamedeveloper DIGITAL EDITION

POSTMORTEM SQUARE ENIX'S THE WORLD ENDS WITH YOU
gamedeveloper
OCTOBER 2008
THE LEADING GAME INDUSTRY MAGAZINE
SENSATIONAL SHOOTING BIOMETRICS FOR PLAYER REACTION IN SHOOTERS
TOOL BOX USING DOXYGEN FOR CODE DOCUMENTATION
DESIGN OF THE TIMES CONSTRUCTING MEANINGFUL CONSEQUENCES IN GAMES
TOP TWENTY PUBLISHERS

Game Developer magazine brings you postmortems, product reviews, and strategies for creating innovative, successful games from industry leaders and game development experts.

www.GDMAG.com

THINK SERVICES

BLIZZARD®
ENTERTAINMENT

IS HIRING

We are actively recruiting across all disciplines for the following locations:

Irvine, California | Austin, Texas | Velizy, France | Cork, Ireland

Seoul, South Korea | Shanghai, China | Taipei, Taiwan

www.blizzard.com/jobs

Visit us in the GDC Career Pavilion at Booth #410.

DigiPen
INSTITUTE OF TECHNOLOGY

DigiPen Institute of Technology's Tradition

As the world's first college to offer a Bachelor's degree program dedicated to game development, DigiPen has been a top choice for students pursuing a career in the interactive media industries. Redmond, WA is one of the largest gaming and software communities in the nation. Home to 70 different gaming companies (including Microsoft and Nintendo), studying in Redmond offers students looking towards a career in the game industry a unique advantage not available elsewhere.

Founded in 1988, DigiPen Institute of Technology continues to be a leader in game development education; our students have placed more games in the Independent Games Festival than students from any other school. Games such as Narbacular Drop (the inspiration for Valve's Portal), Synaesthete, and Empyreal Nocturne are only some examples that prove our students truly understand the complex relationship between user and experience.

At DigiPen, an innovative and pioneering combination of solid foundational skills and theoretical coursework with practical hands-on project experience is incorporated into every semester. With highly academic degree programs taught by a world-class faculty, DigiPen's comprehensive curriculum goes far beyond just teaching art production software or specific programming languages. Through internships and teaching opportunities, our students directly experience what it takes to become successful and valuable members of the interactive media industries.

DigiPen Offers the Following Degree Programs:

- Bachelor of Science in Real-Time Interactive Simulation
- Bachelor of Science in Computer Engineering
- Bachelor of Fine Arts in Production Animation
- Master of Science in Computer Science

Fall 2008: DigiPen Launches Two Game Design Degrees
The designers and creators of electronic and digital entertainment fill a unique role that requires a knowledge of art, technology, innovation, storytelling, history, psychology, and many other disciplines. Based on feedback from industry leaders, DigiPen has developed two new Bachelor degrees to address the needs of the industry:

- Bachelor of Science in Game Design
- Bachelor of Arts in Game Design

A great game designer is sensitive to the player's experience and understands what it takes to create an innovative and engaging experience. Game design students at DigiPen will become experiential experts and develop skills that go far beyond straight game mechanics or story telling. Students will prototype and present numerous projects to DigiPen faculty and personnel from local businesses. Through exposure to challenging problems that practitioners in the field regularly encounter, graduates of these programs will be highly capable of designing quality games for the industry.

DigiPen Opens New Campus in Singapore
Singapore, the prosperous island city-state and growing technical center between Malaysia and Indonesia, is now home to DigiPen Institute of Technology's first international branch campus. Established in cooperation with Singapore's Economic Development Board, the new campus will offer the following degree programs:

- Bachelor of Science in Real-Time Interactive Simulation
- Bachelor of Fine Arts in Production Animation

Both programs will be identical in content and rigor to their US counterparts. Graduates from DigiPen Singapore can expect to contribute to, as well as benefit from, the rapidly developing digital interactive media industries in Singapore.

DigiPen Institute Of Technology
5001 150th Ave NE
Redmond, WA 98052
Phone: (425) 558.0299
Toll-free: (866) 478.5236
Email: admissions@digipen.edu

What employers are saying about DigiPen Graduates

"DigiPen is, bar none, the first choice for Airtight Games in recruiting out of school talent. The training they receive is rigorous and fully applicable to the industry they are entering."
Matt Brunner, Art Director, Airtight Games

"The DigiPen curriculum is well designed in addressing specific needs of game development companies. ArenaNet has had a high success rate in hiring DigiPen graduates. Some of our most talented and promising programmers have graduated from DigiPen."
Steve Shepard, HR Manager, ArenaNet

THINK SERVICES

GAME CAREER SEMINAR SERIES

PROGRAMMED BY THE GAME DEVELOPERS CONFERENCE
IN ASSOCIATION WITH GAMECAREERGUIDE.COM

BREAK INTO THE GAME INDUSTRY TODAY!

Don’t Miss the Opportunity to Network and Learn From Game Industry Professionals.

APPEARING AT:

GDC 09

March 27th, 2009

GDC Canada

May 13th, 2009

WHO KNOWS WHERE WE MIGHT BE NEXT - STAY TUNED.

www.gamecareerseminar.com

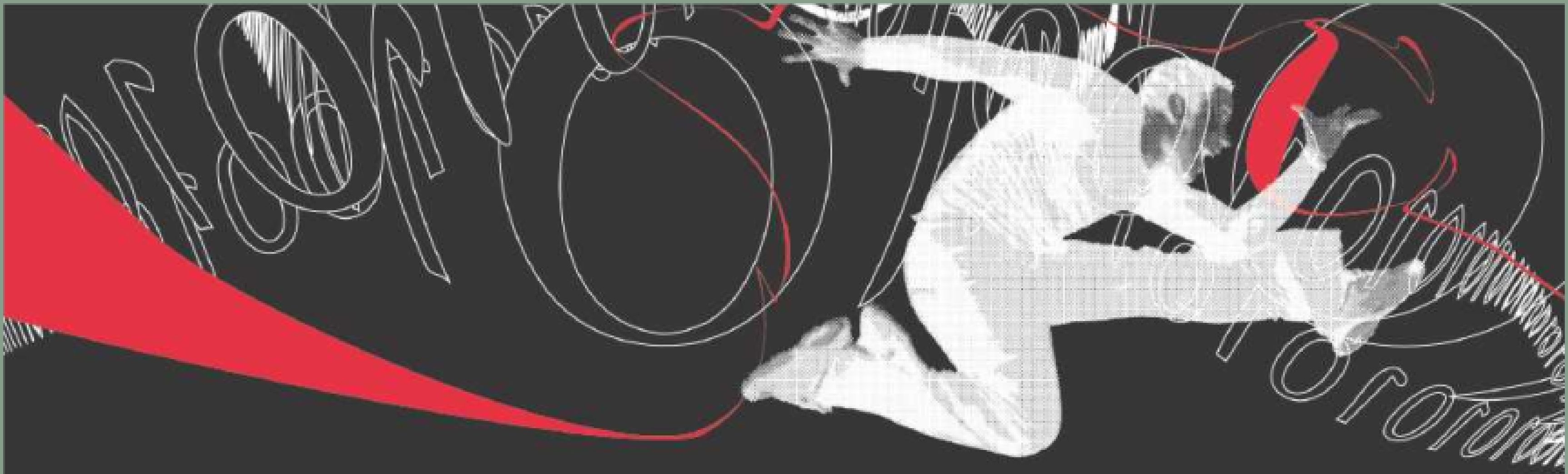
[GEEKED AT BIRTH.]

LEARN:

ADVANCING COMPUTER SCIENCE
ARTIFICIAL LIFE PROGRAMMING
DIGITAL MEDIA
DIGITAL VIDEO
GAME ART AND ANIMATION
GAME DESIGN
GAME PROGRAMMING
NETWORK ENGINEERING
NETWORK SECURITY
ROBOTICS AND EMBEDDED SYSTEMS
TECHNOLOGY FORENSICS
TECHNOLOGY MANAGEMENT
VIRTUAL MODELING AND DESIGN
WEB AND SOCIAL MEDIA TECHNOLOGIES

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk. Can you walk the walk? Here's a chance to prove it. Please geek responsibly. **www.uat.edu** > 877.UAT.GEEK
877.828.4335



BECOME A MASTER OF DIGITAL MEDIA APPLY NOW FOR SEPTEMBER 2009

We offer a 20-month master's degree in entertainment technology and digital media. This powerhouse graduate program combines industry-facing curriculum, real world projects and a 4-month internship in Vancouver, Canada: videogame capital of the world.

Learn more. Contact Alison_robb@gnwc.ca, drop by our booth (#326) at the San Francisco GDC Career Pavilion or visit **mdm.gnwc.ca**

CENTRE FOR **DIGITAL MEDIA**

Masters of Digital Media Program
@ The Great Northern Way

UBC SFU EMILY CARR UNIVERSITY BCIT

a collaborative university campus environment

Create the Game

Gaming Degree Programs for the Next Generation

Game Art
Bachelor's Degree Program

Game Development
Bachelor's Degree Program

Game Design
Master's Degree Program

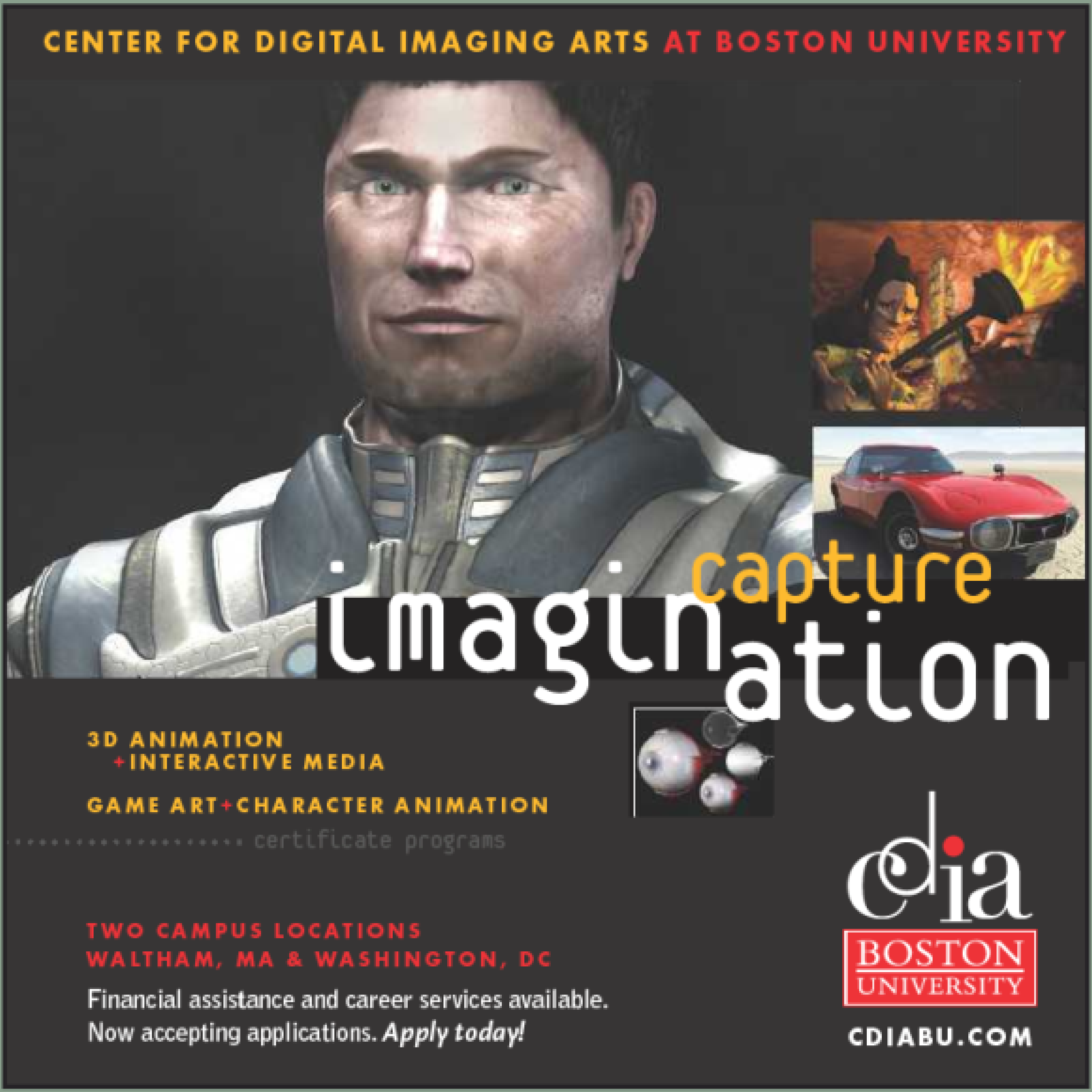
ANIMATION | DESIGN | ENTERTAINMENT BUSINESS | FILM | RECORDING ARTS | SHOW PRODUCTION | VIDEO GAMES | WEB

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

800.226.7625 • 3300 University Boulevard • Winter Park, FL 32792
Financial aid available to those who qualify • Career development assistance • Accredited University, ACCSCT

fullsail.edu *Online Programs Available*

FULL SAIL UNIVERSITY.


CENTER FOR DIGITAL IMAGING ARTS AT BOSTON UNIVERSITY
capture imagination
3D ANIMATION +INTERACTIVE MEDIA
GAME ART+CHARACTER ANIMATION
certificate programs
TWO CAMPUS LOCATIONS WALTHAM, MA & WASHINGTON, DC
Financial assistance and career services available.
Now accepting applications. Apply today!
cdia BOSTON UNIVERSITY
CDIABU.COM

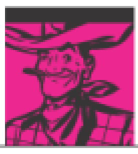
THE LEADING GAME INDUSTRY MAGAZINE
gamedeveloper
Game Developer magazine brings you postmortems, product reviews, and strategies for creating innovative, successful games from industry leaders and game development experts.
WWW.GDMAG.COM
THINK SERVICES

GET EDUCATED

ADVERTISER INDEX

COMPANY NAME	PAGE
Activision	54
Blizzard Entertainment	58
BRADIC—Louisiana State University	23
Center for Digital Imaging	63
DigiPen	59
Epic Games	11
Eyetronics	C2
Full Sail Real World Education	62
Game Connection	24
Georgia Dept. of Economic Development	3
Hansoft	6
Havok	C3
Masters of Digital Media	62
Meaning Makers	30
Monolith Productions	53
Neversoft Entertainment	52
Pocket Soft	39
Rad Game Tools	C4
Sony Computer Entertainment	56
TechExcel	21
Trinigy	13
University of Advancing Technology	61
Ubisoft	55
Unity Technologies	17
Vicious Cycle Software	27

Game Developer (ISSN 1073-922X) is published monthly by United Business Media LLC, 600 Harrison St., 6th Fl., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as United Business Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 4601 W. 6th St. Suite B, Lawrence, KS 66049. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence. All content, copyright *Game Developer* magazine/United Business Media LLC, unless otherwise indicated. Don't steal any of it.



MATTHEW WASTELAND

ARRESTED DEVELOPMENT

HARD NEWS

All the biggest headlines from game development

Game Journalist Totally Hung Out with Yuji Naka

SAN FRANCISCO, CA – Area game journalist Benjamin Day recently hung out with Yuji Naka, one of Japan's top game designers, known for his involvement in seminal Sega titles such as SONIC THE HEDGEHOG.

"Naka-san is totally awesome," Day enthused. "He's super cool and I'm really glad I got the chance to meet him."

At the most recent Tokyo Game Show, Day was lucky enough to be in a small group with some of the other guys from his work and some real-life Japanese people he knows, who said, "Hey, do you want to meet Yuji Naka?" And Day was like, "Of course I do!"

Day and his friends were led to an amazing shabu-shabu place nearby, which is not on any tour books or anything, you pretty much have to be Japanese and know the area to be aware it even exists. The food there was just absolutely incredible, and you could never get it in America ever, not even if you had a Japanese girlfriend, which by the way would be totally sweet.

After the delicious meal, Yuji Naka made his appearance.

"I asked him a ton of stuff like, 'What was Blast Processing anyway, and who came up with the idea for that term?' And Naka-san was like, 'Oh, I don't even remember, it was probably Tom Kalinske.' We all laughed, it was really funny," described Day.

Afterward, the group headed to a kind of traditional Japanese bar known as an *izakaya* and totally just hung out, chatting about games, life in Japan, and the future.

"He asked me what I wanted to see from him next, now that he's not at Sega anymore, and I told him, duh, make a real proper sequel to NIGHTS! Not like that recent one for the Wii but a true follow-up, worthy of the original," Day reported.

Day has since resumed his study of Japanese by watching anime, and now knows the proper translation of "scary," "cute," and "sister."

Sound Designer in Marijuana-Use Shocker

SANTA MONICA, CA – A sound designer was found smoking marijuana back near the storage area of a local video game studio, according to several reports.

The man was not identified, but witnesses described him as having several days' worth of scraggly beard growth. He was heard muttering to himself about "stereo fold-down from a five-dot-one mix."

Local residents expressed surprise at the sighting. "As if there weren't enough uncertainty in these times," said one. "Now there's talk of a sound guy getting stoned! How much more can our world be turned upside down? What's next, the Q/A department?"

Local Producer Still Inept at Bug-Tracking System

BOSTON, MA – "Hey, what do I click to close a bug again?" called out Robert Cardinal, 39, to anyone who might hear him, on a recent Thursday afternoon in the office.

"Oh, crap," he continued, "I think I just closed the wrong thing. Actually, did I delete it? I don't understand all these little icons. Christine? Christine, are you there?"

Production Coordinator Christine Vogel, 28, entered the office and calmly explained the basics of operating the bug database, as fresh-faced as the first time she ever did it.

"That's the 'close' button," she said, pointing to an icon that resembled a green check mark. "Actually, I know that bug isn't closed at all—Frank just told me he hasn't looked at the caching code in over a week. You should keep that bug open."

Mark Weston, the studio head who hired Cardinal, says that he was primarily looking for people skills when conducting the search for a new producer.

"Rob's good at a lot of things," said Weston. "Maybe the nitty-gritty of operating the database isn't one of those, but what I really needed was a hard-hitting straight shooter who can tell it like it is."

Weston explained that one of a producer's jobs is to fully understand the status of the project at all times, and to be able to communicate it when necessary. "Who could I rely on but Rob for that kind of thing?" Weston posed. Rumors of Miss Vogel fiddling with her resume during after hours crunch are as-yet unconfirmed.

New UI Code Created For All (non-German) Languages

WARWICK, UK – Ian R. Hoffman, a programmer at Flinty Bruiser Entertainment, recently finished work on a complete user interface system for his company's next game that perfectly sizes text boxes to match average English word length.

The code, which Hoffman has dubbed BruiserUI, can handle both shell screens and in-game HUD elements. It includes support for multiple platforms and is fully double-byte enabled in order to support Asian languages.

"I made extra sure to check the longest words we could possibly use in our game against all of the text boxes, and it all lines up perfectly," Hoffman said. "Everything is readable and, best of all, there's no wasted space on the screen."

When confronted with the possibility of German translations being longer than their English equivalents, Hoffman responded, "Eh? I'm sure it'll be fine."

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

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

BINK

BINK HAS SHIPPED IN 2,900 GAMES FOR A REASON.

IT'S THE STANDARD FOR VIDEO IN GAMES - UP TO 4 TIMES FASTER AND USING AS MUCH AS 18 MB LESS MEMORY THAN OTHER CODECS.

IT HAS A BUILT-IN AUDIO CODEC WITH MULTITRACK CAPABILITY, SUPPORTS EVERY PLATFORM, AND CAN HANDLE AN ARBITRARY NUMBER OF VIDEO CHANNELS, LETTING YOU ADD Z-DEPTH, NORMALS, OR UV PER-PIXEL DATA, AND MORE!

FLAWLESS HD VIDEO WITH 5.1 AUDIO ON WINDOWS, XBOX AND PS3!

RAD GAME TOOLS

Now playing on the Wii, PSP, PS3, and Xbox 360

425-893-4300 www.radgametools.com

GRANNY 3D BINK VIDEO PIXOMATIC MILES