

MARCH 2013
VOLUME 20 NUMBER 03

gd

postmortem
the WALKING DEAD

GAME DEVELOPER MAGAZINE

THE LEADING GAME INDUSTRY MAGAZINE



ASOBO

CRAFTING THE FUTURE OF FUN

*Programmers,
Game Designers,
3D Artists,
or Animators.*

APPLY NOW

jobs@asobostudio.com

See website for details : www.asobostudio.com

ASOBO
STUDIO



Postmortem

036 THE WALKING DEAD

Q: When is a zombie game not just a zombie game? A: When it's Game of the Year. Telltale Games CTO Kevin Bruner explains how their craft for narrative set THE WALKING DEAD apart from the rest of the (zombie) pack. *By Kevin Bruner*

Features

013 QUALITY-OF-LIFE SURVEY

Devs: How are you doing? Check out the results from *Game Developer's* first quality-of-life survey. *By Brad Bulkley and Patrick Miller*

021 INTERACTIVE FICTION RENAISSANCE

Text games are back! *Game Developer* talks with five movers and shakers behind the resurgence of interactive fiction. *By Leigh Alexander*

029 THE OLD GUARD

Find out what's next for next-gen engines in this interview with Epic Games CTO Tim Sweeney. *By Brandon Sheffield*

045 FIVE TIPS FOR BETTER PLAYTESTING

Make the most out of your playtesting sessions with these five tips from casual game developer Arkadium. *By Vin St. John*

Departments

002	Game Plan	[Editorial]
004	Heads Up Display	[News]
006	Educated Play	[Education]
009	Good Job	[Career]
010	GDC News	[News]
049	Toolbox	[Review]
055	Inner Product	[Programming]
069	Pixel Pusher	[Art]
078	Design of the Times	[Design]
085	Aural Fixation	[Sound]
086	The Business	[Business]
089	Insert Credit	[Editorial]
096	Arrested Development	[Humor]



WHAT IS A GAME?

HOW I STOPPED WORRYING AND LEARNED TO LOVE THE GAMES I DIDN'T LIKE

Right now, you're reading *Game Developer* magazine. You're probably reading it at the Game Developers Conference, or maybe during your lunch break working at a game development studio—hopefully, you're making games that you'd like to play yourself. Perhaps you've looked at someone's game and thought, "Yuck—who'd ever want to play that?" Maybe you've even followed that up with, "Games just aren't what they used to be."

(MORE THAN JUST) A MISERABLE LITTLE PILE OF SECRETS

Video games are an incredibly diverse medium. When we describe other media (literature, film, music, and so on), we can establish a few basic common assumptions about the experience of consuming those works; music is something we listen to, books are things we read, films are things we watch, and the difference between artists and genres within those media are defined by content (novels and short stories and graphic novels).

You might say video games are something we play, but that's still vague; the word "play" could describe something I do with a soccer ball, or a Dungeons & Dragons crew, or slot machines, or the lottery, or a set of dolls. So, too, with video games: Some are (e)sports, some are rules-driven narrative experiences, some are games of chance, some are works of interactive art, some are toys. Pretty much the only thing one game is guaranteed to have in common with another game is the "video" part, and even that is questionable (see J.S. Joust).


LEARN FROM THY NEIGHBOR I'll admit it right now: When I started working at *Game Developer* this time last year, I was all excited to see what was going on with the core PC/console devs. Indies? Well, they're okay, but not my thing. Mobile? Sure, I guess. Social games? Free-to-play? If I have to. I mean, sure, they're "video games," but they're not really video games, am I right? Well, no, I wasn't—and I missed out because of that mindset.

I wasn't really into social games, but a few months ago I tried the beta of Zynga's THE FRIEND GAME—and had a lot of fun asking and answering questions with Facebook friends that I never would have otherwise asked. I never liked playing games on touchscreens, but now I spend more time playing SUPER HEXAGON on my phone than I spend on Facebook or Twitter. And while I'm not a huge

fan of free-to-play's effect on competitive multiplayer game designs, last year I spent more on free-to-play action games like MECHWARRIOR ONLINE and PLANETSIDE 2 than I did on Steam games, because I was able to play them for a long time without having to shell out \$50 up front.

None of these games replaced the games I love; love for video games is additive, not subtractive. But in dabbling with games across the immensely vast spectrum of video games, I developed my appreciation for how devs had honed so many different skills, whether it's the folks who figured out how to tell a story that works on a TV or a phone (see our postmortem of THE WALKING DEAD on page 36), the artist who puts the pixels in exactly the right place, the programmers who make everything happen, and even the devious geniuses who persuade me to pay for (free) games. Even if you don't enjoy these games, it's worth trying to see what others like about them (see Jason VandenBerghe's Design of the Times column, page 78).

LEAVING YOUR GREEN HILL (COMFORT)

ZONE If there's one thing I've learned in the last year, it's that we're all in this together. When a studio closes or a team gets laid off, we all feel it. When someone blames video games for youth violence, we all feel it—even if you, specifically, don't work on violent games. Instead of moaning about how video games just aren't what they used to be because people are making games that you might not personally like, I urge you to embrace the spirit of game camaraderie. Play a game outside your comfort zone. Talk to a fellow dev who works in a different segment of the industry than you do. If you're at GDC, go to a session that might not be immediately relevant to your day-to-day job. You just might be pleasantly surprised. 

—Patrick Miller
@pattheflip



GAME DEVELOPER
MAGAZINE
WWW.GDMAG.COM

UBM LLC.

303 Second Street, Suite 900, South Tower
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
gamedeveloper@ahldata.com
www.gdmag.com/contactus

EDITORIAL

PUBLISHER

Simon Carless scarless@gdmag.com

EDITOR

Patrick Miller pmiller@gdmag.com

EDITOR EMERITUS

Brandon Sheffield bsheffield@gdmag.com

MANAGER, PRODUCTION

Dan Mallory dmallory@gdmag.com

ART DIRECTOR

Joseph Mitch jmitch@gdmag.com

CONTRIBUTING WRITERS

Alexandra Hall, Brad Bulkley, Leigh Alexander, Brandon Sheffield, Kevin Bruner, Dave Wilkinson, Andrew Maximov, Jason VandenBerghe, Damiam Kastbauer, Kim Pallister, Matthew Wasteland, Magnus Underland

ADVISORY BOARD

Mick West Independent
Brad Bulkley Microsoft
Clinton Keith Independent
Bijan Forutanpour Sony Online Entertainment
Mark DeLoura Independent
Carey Chico Independent
Mike Acton Insomniac
Brenda Romero Loot Drop

ADVERTISING SALES

VICE PRESIDENT, SALES

Aaron Murawski aaron.murawski@ubm.com
t: 415.947.6227

MEDIA ACCOUNT MANAGER

Jennifer Sulik jennifer.sulik@ubm.com
t: 415.947.6227

GLOBAL ACCOUNT MANAGER, RECRUITMENT

Gina Gross gina.gross@ubm.com
t: 415.947.6241

GLOBAL ACCOUNT MANAGER, EDUCATION

Rafael Vallin rafael.vallin@ubm.com
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER

Pete C. Scibilia peter.scibilia@ubm.com
t: 516-562-5134

REPRINTS

WRIGHT'S MEDIA

Jason Pampell jpampell@wrightsmedia.com
t: 877-652-5295

AUDIENCE DEVELOPMENT

AUDIENCE DEVELOPMENT MANAGER

Nancy Grant e: nancy.grant@ubm.com

LIST RENTAL

Peter Candito
Specialist Marketing Services
t: 631-787-3008 x 3020
petercan@SMS-Inc.com
ubm.sms-inc.com



WWW.UBM.COM



It's time
to get
hands
on

fmod®studio

Visit our booth 502 at the GDC expo for a test drive.

<http://www.kickstarter.com/projects/1544851629/throw-trucks-with-your-mind>

THROW TRUCKS WITH YOUR MIND

USING AN EEG FOR IN-GAME TELEKINESIS

Gravity guns and dragonborn shouts just don't have the same panache as true telekinesis. *Game Developer* chatted with Lat Ware, an indie dev working on a game called **THROW TRUCKS WITH YOUR MIND**, about the process of designing a game that uses the brain as an input device.

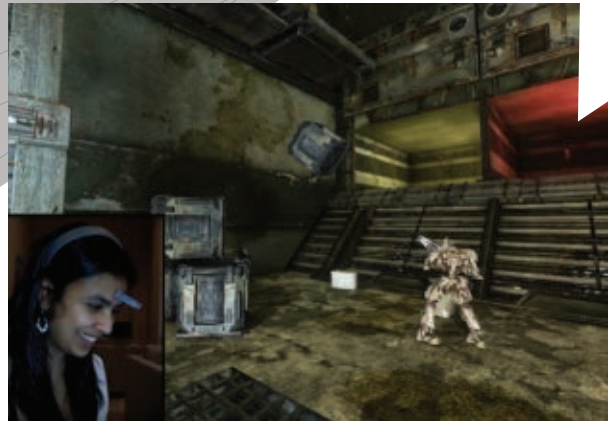
Patrick Miller: So how, exactly, does this work?

Lat Ware: I'm using Neurosky's Mindwave for my electroencephalograph (EEG). They've been getting a lot of attention for the mind-controlled cat ears lately, and I'm using the same hardware. An EEG is a really sensitive voltmeter that is looking at voltage patterns in the surface of the brain. Telling what a person is thinking is an incredibly hard problem to solve, but the headset does tell me how they are thinking; it's doing a discrete Fourier transform and some other signal cleanup algorithms on 1-second blocks of your alpha and beta waves to tell how calm and focused you are.

While I would like to do a purely brain-controlled game, the hardware just isn't there yet. So I am building a game with standard first-person shooter controls (keyboard and mouse) for movement that uses the measurement of how calm and focused you are as the scalars on your Jedi-like super-powers. You select what power you want to use with the number keys, the same way you select weapons in FPS games, and each power is tied to a specific effect: throwing something is powered by how focused you are, pulling something is powered by how calm you are, super-jumping is powered by focus, slow fall is powered by calm. You can only have one power active at a time, and the players don't have guns in this game, so it's up to them to figure out how to use the environment to destroy their enemies.

PM: What's it like working with an EEG as an input device? How's the lag?

LW: The headset itself has at most 1 second of lag. I deliberately introduced some more lag by making your focus and calm values in the game be a rolling average of the last 3-5 seconds, and I'm still playtesting to figure out what is best. The reason I did this was that when I was having people playtest this very early on, I noticed that as soon as people got any negative feedback, they would panic and usually say something to the effect of, "What did I do wrong?" At that moment, they are frantically searching



for what they did wrong, which destroyed their calm and focus, sending them into a death spiral.

By smoothing out the values with a rolling average, I got rid of the death spiral, and I think that is definitely worth the lag that I introduced. You will see the effects of your changes immediately, but not the full magnitude of them. To my amazement, everyone who has tried this has gotten the hang of controlling their calm and focus within 15 minutes. This is the case even among non-gamers. If anything, first-person shooter controls are harder to grasp than the headset—the player has considerable control over the headset, but it is using a muscle that you didn't know you had, so everyone starts from the same point, unless you practice meditation or neurofeedback therapy. Those people always know how to use the headset before they start.

PM: What kind of challenges have you run into with the EEG?

LW: I noticed in playtesting that when two people engaged in a battle of wills, unless someone really fell apart in the battle, both players would get exhausted and nothing would happen. This is because if you are trying to move one of the little crates, you have to be at a minimum of 35% focus. If your opponent is average (50%), then you have to be at 85% focus—which is hard—just to get the crate to move, which isn't enough to turn it into a weapon. So, I put in a mechanic where, if two people oppose each other on an object, they will keep pumping more and more energy into it, so the object will feel a stronger force from every player trying to affect it, and eventually, even if two players are closely matched, one will overpower the other.

Most people, when they first play, are completely exhausted after an hour. I'm completely used to the game, so I can handle it just fine, but I have been working on this for a year, and I did not document how long it took me to adapt.

Also, I really wanted to have Force Lightning in the game, but the only way to do that properly is to have it be powered by anger, and I really don't want to reward players for getting angry. Without Force Lightning, I have a game where you have to be clever and figure out how to use the environment to destroy your enemies. I didn't want to bring in MODERN WARFARE where you look at the person you want to die until he dies. To many games are fighting over that game mechanic and I want to do something new.

-Patrick Miller

PURSuing IMPERFECTION

USING NEW TECH TO SIMULATE OLD VISUALS

Most consumers are always clamoring for higher-definition graphics, but one relatively small group is seeking just the opposite. A dedicated cadre of retro-gamers and emulation enthusiasts is perfecting the art of simulating low-res, standard-definition-style graphics on today's high-definition LCDs.

"I grew up playing 2D games with chunky pixel-art graphics on glowing CRT displays," said Hunter Kaller, an emulation hobbyist who blogs at filthypants.blogspot.com.

rate of 15.75kHz, which today's monitors must convert, or pscale, to their native 31kHz. All that processing introduces a number of artifacts, most notably a lack of the distinctive scanlines typical of a CRT.

Retro videophiles refer to the low-res 15.75kHz video mode old consoles use as "240p"; while the signal is technically interlaced, it only ever draws to the odd or even lines, never alternating. The inactive lines remain dark, which results in scanlines. The scanlines, as well as other traits inherent to CRTs, help blend the raw pixels put out by the consoles, resulting in

illusion. All told, you're looking at a few hundred dollars.

"I don't have room for a CRT and I still spend a lot of time with the older consoles," said Matt Buxton, who writes about retro videophile hardware at VideogamePerfection.com. "I didn't want to buy a nice flat-screen TV and have to compromise picture quality on certain games, so some sort of upscaling solution was clearly needed. There's the convenience factor, too; my girlfriend and I like to play co-op, and we might go from 1080p LEFT 4 DEAD on the PC to 240p SECRET OF MANA on the SNES. It's quite convenient to be able to do that

behavior to approximate the red, green, and blue phosphor components even at current typical resolutions, but hopefully this won't be necessary once 4K displays are widespread," said Kaller.

Some users dial down the CRT shader's barrel distortion, overscan, and other variables to simulate a mythical "perfect" CRT, but others seek even more artifacts. Shay "Blargg" Green created a set of filters that simulates the look of RF, composite, S-video, and RGB video signals. "Part of my nostalgia for playing games was the unique look of each console's composite video, and I was always curious about why," said Green. "NewRisingSun figured out the math and algorithm to simulate composite video and shared it, which motivated me to make it practical in emulators. I had NewRisingSun's algorithm in hand, the skills to do massive optimizations to it, and the vision of a clean real-time NTSC filtering library." As a result, nostalgia purists can enjoy emulated games with authentic-looking dot crawl and similar artifacts. Future historians, too, could get a better idea of what old games looked like without needing vintage hardware.

In a recent forum post, Higan author Byuu laid out his view of the road ahead. "In order to properly mimic a CRT, we need two major things first: much higher contrast (OLED) is the most likely right now, and at least double the DPI. And then once we have that, we'll need someone to come up with the algorithm. But yeah, right now, we can't make a fully convincing replication." The road to perfect CRT simulation may be long, but at least it has a very appealing phosphor glow.

—Alexandra Hall



Left: Typical upscaled emulator output. Right: Emulator output using cgwg's CRT shader.

"When I got into emulation in the late '90s, I was shocked at how different my favorite games looked on high-resolution displays, and was shocked again when I made the switch to modern LCD displays. Nothing ever looked quite right. Scanline filters helped a bit, but when I saw Xythen's mock-ups for his 'phosphor3x' filter concept, something clicked. I thought, 'That's how these games are supposed to look.'"

The problem is that old consoles output interlaced video with a horizontal scan

the soft but attractive—and relatively unpixelated—low-res TV graphics of yesteryear.

It's surprisingly tricky to reproduce this look on modern displays. Gamers who want to do so with their old consoles typically process the video signal with specialized devices. Micromsoft's XRGB line of gamer-oriented upscaler converters changes 15.75kHz signals to 31kHz, optionally adding simulated scanlines. Someone using a more general-purpose upscaler could add an SLG3000 scanline generator to complete the

on the same display with just a few button pushes."

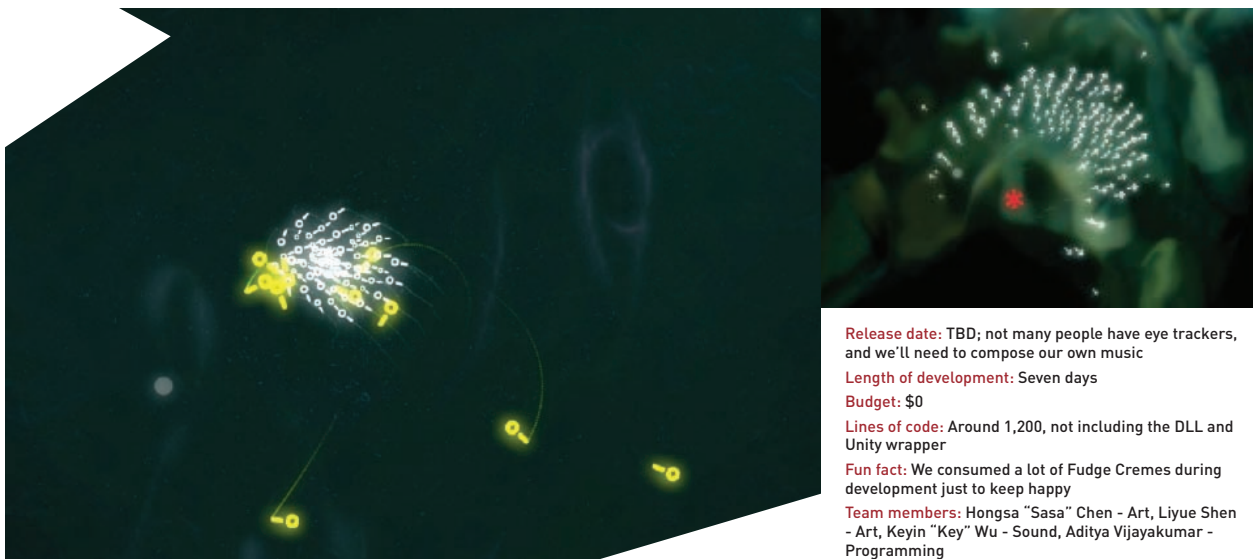
PC-based emulators, meanwhile, have long offered mediocre scanline filters. But recently, cutting-edge emulators like Higan (formerly bsnnes) and RetroArch have begun to harness the untapped power of today's GPUs. In 2010 a programmer called cgwg created the most advanced CRT shader to date, which simulates the behavior of actual CRT phosphors. The result looks agreeably close to a real CRT. "Cgwg's shader cleverly exploits LCD subpixel

Aquario

[HTTP://AQUARIOEYEGAME.WEBS.COM/](http://aquarioeyegame.webs.com/)

[HTTP://WWW.IGF.COM/PHP-BIN/ENTRY2013.PHP?ID=1302](http://www.igf.com/php-bin/entry2013.php?id=1302)

IF ONE-BUTTON GAMES ARE STARTING TO FEEL A LITTLE COMPLEX AND ONEROUS, PERHAPS EYE TRACKING WOULD BE MORE YOUR SPEED. GAME DEVELOPER CHECKED IN WITH ADITYA VIJAYAKUMAR OF CARNEGIE MELLON, WHO IS ONE-FOURTH OF THE STUDENT DESIGN TEAM BEHIND AQUARIO, A FLOW-LIKE GAME OF SWIMMING AND CONSUMPTION CONTROLLED ENTIRELY BY EYE MOVEMENT.



Release date: TBD; not many people have eye trackers, and we'll need to compose our own music

Length of development: Seven days

Budget: \$0

Lines of code: Around 1,200, not including the DLL and Unity wrapper

Fun fact: We consumed a lot of Fudge Cremes during development just to keep happy

Team members: Hongsa "Sasa" Chen - Art, Liyue Shen - Art, Keyin "Key" Wu - Sound, Aditya Vijayakumar - Programming

Alexandra Hall: What came first, the eye-tracking hardware or the idea for Aquario?

Aditya Vijayakumar: We actually decided on the eye-tracking device first. In our Building Virtual Worlds (BVW) class we had to make a fun game within one week, using only the Jam-O-Drum, MaKey MaKey, or the eye tracker for input. We decided to go with the eye tracker since it was a new platform introduced this semester, and we would probably be the first ones to make a game for this system. We came up with different types of gameplay that would work with the eye tracker but decided to finally go with the swarm mechanic.

AH: How accurate is the tracker?

AV: Initially we were skeptical about using the eye tracker, since we were worried about the accuracy and how it would actually work. But we jumped at the opportunity to use a very unique input device, and BVW is the best place to fail. We took one day to make the initial prototype just testing the game with the mouse. The game worked well, and to our amazement, it worked really well when we first tested it on the actual eye tracker. The cursor was spot on. The system calibrates before you launch the game, so the machine adjusts to different people's eyes. The system also takes care of head movements and eye blinks to a certain level, so players can still blink and move their heads to a certain extent.


AH: Do you see much potential for eye tracking in games?

AV: I think there are a lot of possibilities to use the eye tracker as an input device; we had several different types of games within our class itself, all of which made use of the eye tracker with

different types of gameplay. At its current state I don't think it's a commercially viable platform due to its high price tag and lack of portability, but I believe that it will have a lot of gaming potential when it overcomes these problems.

AH: What kind of challenges did you encounter?

AV: Making a game for the eye tracker was more of a design hurdle. Programming was fairly simple since it was similar to programming a mouse, but we found out a few things. The eyes aren't as stable as a mouse; your eyes automatically focus on anything attractive even before you think. For example, if there is a small pink box on a blank screen you automatically focus your eyes toward that pink box. In mouse games, the person playing the game sees the object, thinks what to do next, and then moves the mouse based on what they were thinking. But when making games for the eye tracker you need to keep in mind that the person playing is going to see things before they think and then decide where to look next. The eyes usually act as a way in which your body receives input and you give signals to your hand or other parts of the body to react based on what you saw. But when programming an eye-tracking game we need to keep in mind that the eyes act as both an input and output organ.

After analyzing these things we decided to make a game that is both relaxing as well as attractive with a very simple goal, evolve by increasing the size of your colony. All the player needs to do is look at a point on the screen and the swarm will move toward that point. We made the background a little dull, our swarm white, and consumable organisms attractive colors so they catch the player's attention immediately. 



VFS student work by Benjamin Erdt

MAKE MORE ENEMIES

Game Design at VFS lets you make more enemies, better levels, and tighter industry connections.

In one intense year, you design and develop great games, present them to industry pros, and do it all in Vancouver, Canada, a world hub of game development.

VFS

Find out more.
VFS.COM/ENEMIES

THE ONLY ONE-YEAR PROGRAM
IN PRINCETON REVIEW'S 2012
TOP GAME DESIGN PROGRAMS

KONAMI



KOJIMA PRODUCTIONS

JOIN THE FOX TEAM

THE METAL GEAR SOLID TEAM IS HIRING FOR VARIOUS POSITIONS AT THE NEW LOS ANGELES STUDIO AS WELL AS THE TOKYO HEADQUARTERS. APPLY AT THE GAMASUTRA JOB BOARD OR IN PERSON AT THE GDC 2013 CAREER PAVILION AT

BOOTH CP2308



JAPAN



LOS ANGELES

APPLY FOR TOKYO:
JOB_PRO@KONAMI.COM

PLEASE VISIT OUR WEBSITE AT:
[HTTP://WWW.KONAMI-DIGITAL-ENTERTAINMENT.COM/JOBS.PHP](http://www.konami-digital-entertainment.com/jobs.php)

APPLY FOR LA:
KJPLA@KONAMI.COM



EDWARD BOUND

TRIPLE-A VET EDWARD DOUGLAS TALKS ABOUT FILM IN GAME DEV, CANADA, AND WEARING MANY HELMETS AT HIS NEW STUDIO.

It's not easy to transition from triple-A to scrappy indie, but Edward Douglas's multihatted background in film editing, game cinematics, and creative direction gave him a little bit of an edge. *Game Developer* caught up with Douglas to ask him about the Canadian dev scene, working film experience into the game development process, and his secret to filling several roles at his new studio, Flying Helmet Games.

Patrick Miller: What's your game dev background, and how'd you come to start your own studio?

Edward Douglas: I'm a storyteller, filmmaker, and game designer from Vancouver, Canada. I started in film and television as an editor and an indie cameraman (my first movie was Uwe Boll's *House of the Dead!*), then I joined Electronic Arts's Blackbox for a few rounds of NEED FOR SPEED as cinematics director. Later, I worked at BioWare on MASS EFFECT 2 and new IP dev, and at Ubisoft on H.A.W.X. and the RAINBOW 6 franchise at the creative and project direction level. I was always talking with colleagues about doing a small game of our own, and

the project crystallized for us early in 2012, so we started to put Flying Helmet Games together. I felt like I was in my own *Ocean's Eleven*, recruiting all my former colleagues and friends to help us out.

PM: What's your role at Flying Helmet Games right now?

ED: We have a lot of trouble defining titles at FHG! I call myself creative director, and we call Scott Penner lead developer, because it's better than calling him lead designer/character tech director/programmer/writer! On any given day I may do project management, HR, company strategy, level design, camera and cinematic design, and supervise music. The great thing about starting out in cinematics is that that department touches every other department in a game.

PM: The Canadian dev scene has seen some big changes lately. What's the mood like among devs in Vancouver right now?

ED: When I left Vancouver, the city was booming. When I returned five years later, Propaganda and Blackbox were gone, Radical, Rockstar, and Ubisoft were shutting down, and EA and Microsoft were drastically downsizing. People are worried and looking toward Montreal and even

Toronto, or just leaving the country. But you also have a huge population of experienced developers hungry to make games, so we're seeing more independent studios, like ours, with fantastically experienced developers. I'd say it's a mix of nervousness and excitement.

PM: I hear you have a few folks with a background in film; how are they involved in the dev process?

ED: Coming from a dev and production background, I knew how to make games, but didn't know much about raising funds. One of the first partners we brought on was a film producer, Haydn Wazelle, whom I had worked with in my previous life when I shot his first feature. Publishers are committing less to pitches and risky IP, so his experience with film financing channels has been invaluable. However, there's a big learning curve there for both of us; although there's a lot of similarities to film production, the differences are vast, in part because game business is constantly evolving, while film hit its stride decades ago. One of the things we're doing is treating the production more like a typical film production, where we know there's a defined start and end date for the team. So many studios balloon and lay off in

cycles around their games, and it hits the news and makes the business sound completely chaotic, but this has been standard operating procedure in film forever. We also have been fortunate to get some great team members from the film visual effects side.

PM: Your first project aims to use mobile devices to create a tabletop board game experience. Have you run into any unexpected challenges?

ED: We are making a "tabletop video game," which uses a combination of tablets and smartphones to create a "round the gaming table" social experience. It's not a digital board game so much as a video game where you get the screen down away from your face, so you can look your opponents and teammates in the eye. In-game storytelling can really be the players talking to each other, rather than voiceover or reading dialogue. We can have players leading and instructing each other, as well as players keeping secrets or bartering. We get to allow players to do things that might seem unfair in online games, because we know the players can manage themselves. (If you screw over your friend next to you, watch out because he might be your ride home.)

Who Went Where

- **MIKE BITHELL, LEAD DESIGNER AT BOSSA STUDIOS** and creator of the indie platformer hit THOMAS WAS ALONE, departed from Bossa at the end of January to focus on his own projects. "[So] next Monday I'll start a new Unity file, and see where it takes me," Bithell said.
- **JAY WILSON, LEAD DIRECTOR OF DIABLO III**, is moving on to a new project at Blizzard after seven years helming the iconic franchise. "I've reached a point creatively where I'm looking forward to working on something new," he wrote.
- **BRIAN REYNOLDS, THE CHIEF GAME DESIGNER** at social and mobile game giant Zynga, is leaving the company. The former lead designer of Firaxis's classic CIVILIZATION II is leaving to "pursue other opportunities."
- **JARED SORENSON IS THE NEW VICE PRESIDENT** of gaming at Plyfe, a self-described "digital amplification platform" that gamifies various activities in the web and mobile spaces. Sorenson previously occupied writing and design positions at Turbine, White Wolf, and LucasArts, among others.

New Studios

- When THQ filed for bankruptcy and auctioned its assets, well-regarded **DARKSIDERS STUDIO VIGIL CONSPICUOUSLY FAILED** to find a buyer. Enter Crytek. On January 28 the German developer/publisher founded Crytek USA, the company's first American studio, by hiring 35 developers from the former Vigil studio, including Vigil general manager David Adams. "It would be pretty much safe to say that this team will be working on online games," said Crytek CEO Cevat Yerli, adding that the new team at Crytek USA will focus on "online games and kick-ass triple-A productions. It is going to be quite a significant investment for Crytek over the next five years."
- Four key developers from Sony Liverpool, which finished one final WIPEOUT game **BEFORE BEING SHUTTERED BY SONY LAST AUGUST**, have launched a new studio called Sawfly. The devs are in the midst of taking on contracts from other companies, finishing up their new studio's first game, a "cheeky, irreverent" game for publisher Ripstone, and pitching a new property. Futuristic racers are not on their agenda for now.

DIVERSE FIELD OF FINALISTS GRACES 2013 GAME DEVELOPERS CHOICE AWARDS



THE WALKING DEAD.

Organizers have revealed the finalists for the 13th annual Game Developers Choice Awards, the leading peer-based video game event celebrating the industry's top games and developers. The winners will be honored at the Game Developers Choice Awards ceremony, taking place on Wednesday, March 27, at 6:30 p.m. at the Moscone Convention Center during the 2013 Game Developers Conference in San Francisco.

Independent titles are heavily represented across multiple categories this year, with games from that segment of the industry being nominated alongside some of the year's biggest mainstream releases.

Overall, Thatgamecompany/Sony Computer Entertainment's visually and aurally stunning emotional adventure JOURNEY leads with nods in six categories: Best Design, Best Downloadable Game, Best Audio, Best Visual Arts, Innovation, and Game of the Year. Arkane Studios/Bethesda Softworks's steampunk revenge epic DISHONORED is recognized in four categories: Best Narrative, Best Visual Arts, Best Design, and Game of the Year. Another leading nominee is Telltale Games's THE WALKING DEAD, a postapocalyptic adventure title based on Robert Kirkman's comic book series, honored in three categories for this year's awards: Best Downloadable Game, Best Narrative, and Game of the Year.

The Game Developers Choice Awards also features three Special Award categories, whose recipients were recently announced. The 2013 Lifetime Achievement award is going to Ray Muzyka and Greg Zeschuk, the co-founders of BioWare; the Pioneer award is going to Steve Russell for SPACEWAR!; and the Ambassador award is going to Chris Melissinos, for his work curating the Smithsonian American Art Museum's "The Art of Video Games" exhibition. In addition, the first-ever Audience Award for the Choice Awards—open to all finalists—opened for online voting in mid-February.

For more information about the 13th annual Game Developers Choice Awards, visit its official website. or information about the 2013 GDC, visit its official website. (GDC and the Game Developers Choice Awards are owned and operated by UBM TechWeb, as is *Game Developer*.)

The 13th Annual Game Developers Choice Awards



2013 NOMINEES

GAME OF THE YEAR

Dishonored ARKANE STUDIOS/BETHESDA SOFTWORKS
The Walking Dead TELLTALE GAMES
Mass Effect 3 BIOWARE/ELECTRONIC ARTS
XCOM: Enemy Unknown FIRAXIS GAMES/2K GAMES
Journey THATGAMECOMPANY/SONY COMPUTER ENTERTAINMENT

BEST DEBUT

Humble Hearts DUST: AN ELYSIAN TAIL
Polytron Corporation FEZ
Giant Sparrow THE UNFINISHED SWAN
Subset Games FTL: FASTER THAN LIGHT
Fireproof Games THE ROOM

BEST GAME DESIGN

Dishonored ARKANE STUDIOS/BETHESDA SOFTWORKS
Mark Of The Ninja KLEI ENTERTAINMENT/MICROSOFT STUDIOS
Spelunky DEREK YU/ANDY HULL
Journey THATGAMECOMPANY/SONY COMPUTER ENTERTAINMENT
XCOM: Enemy Unknown FIRAXIS GAMES/2K GAMES

BEST DOWNLOADABLE GAME

The Walking Dead TELLTALE GAMES
Spelunky DEREK YU/ANDY HULL
Trials: Evolution REDLYNX/MICROSOFT STUDIOS
Mark Of The Ninja KLEI ENTERTAINMENT/MICROSOFT STUDIOS
Journey THATGAMECOMPANY/SONY COMPUTER ENTERTAINMENT

BEST TECHNOLOGY

Far Cry 3 UBISOFT MONTREAL/UBISOFT
PlanetSide 2 SONY ONLINE ENTERTAINMENT
Halo 4 343 INDUSTRIES/MICROSOFT STUDIOS
Call of Duty: Black Ops II TREYARCH/ACTIVISION
Assassin's Creed III UBISOFT MONTREAL/UBISOFT

BEST HANDHELD/MOBILE GAME

Gravity Rush SCE JAPAN STUDIO/SONY COMPUTER ENTERTAINMENT
Hero Academy ROBOT ENTERTAINMENT
Sound Shapes QUEASY GAMES/SONY COMPUTER ENTERTAINMENT
The Room FIREPROOF GAMES
Kid Icarus: Uprising SORA/NINTENDO

BEST NARRATIVE

Spec Ops: The Line YAGER ENTERTAINMENT/2K GAMES
Mass Effect 3 BIOWARE/ELECTRONIC ARTS
Dishonored ARKANE STUDIOS/BETHESDA SOFTWORKS
The Walking Dead TELLTALE GAMES
Virtue's Last Reward CHUNSOFT/AKSYS GAMES

BEST VISUAL ARTS

Borderlands 2 GEARBOX SOFTWARE/2K GAMES
Journey THATGAMECOMPANY/SONY COMPUTER ENTERTAINMENT
Far Cry 3 UBISOFT MONTREAL/UBISOFT
Dishonored ARKANE STUDIOS/BETHESDA SOFTWORKS
Halo 4 343 INDUSTRIES/MICROSOFT STUDIOS

BEST AUDIO

Journey THATGAMECOMPANY/SONY COMPUTER ENTERTAINMENT
Hotline Miami DENNATON GAMES/DEVOLVER DIGITAL
Sound Shapes QUEASY GAMES/SONY COMPUTER ENTERTAINMENT
Assassin's Creed III UBISOFT MONTREAL/UBISOFT
Halo 4 343 INDUSTRIES/MICROSOFT STUDIOS

INNOVATION

Mark of the Ninja KLEI ENTERTAINMENT/MICROSOFT STUDIOS
Journey THATGAMECOMPANY/SONY COMPUTER ENTERTAINMENT
FTL: Faster Than Light SUBSET GAMES
The Unfinished Swan GIANT SPARROW/SONY COMPUTER ENTERTAINMENT
ZombiU UBISOFT MONTPELLIER/UBISOFT

A Complete Monetization Platform for Online Games



100+ Payment Choices



Localized Purchase Experience



Flexible Subscriptions



Hybrid Free to Play + Subscription Models



Reporting & Analytics



In-Game Optimized Purchase Experience



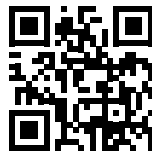
playspan.

Online Games Commerce

Learn more

— GDC 2013 - Stop by the Visa booth #824 in South Hall to learn more about our PlaySpan and V.me by Visa products

— Go to www.playspan.com/gdc2013 for more product information and white paper downloads



Frank N. Magid Associates, Inc.

Come by Visa booth #824 on Wednesday at 3:30 to hear Magid Associates present their report on online game business models.

Or request a copy at www.playspan.com/gdc2013. We will send it out after the event.

PlaySpan is now part of **VISA**



WARGAMING.NET

LET'S BATTLE



WARGAMING.NET IS HIRING FOR MULTIPLE DISCIPLINES!

SAN FRANCISCO
California

CHICAGO
Illinois

HUNT VALLEY
Maryland

US Openings
International Openings

www.wargamingwest.net and www.wargamingamerica.com
www.wargaming.net



gd 013

GAME DEVELOPER QUALITY-OF-LIFE SURVEY

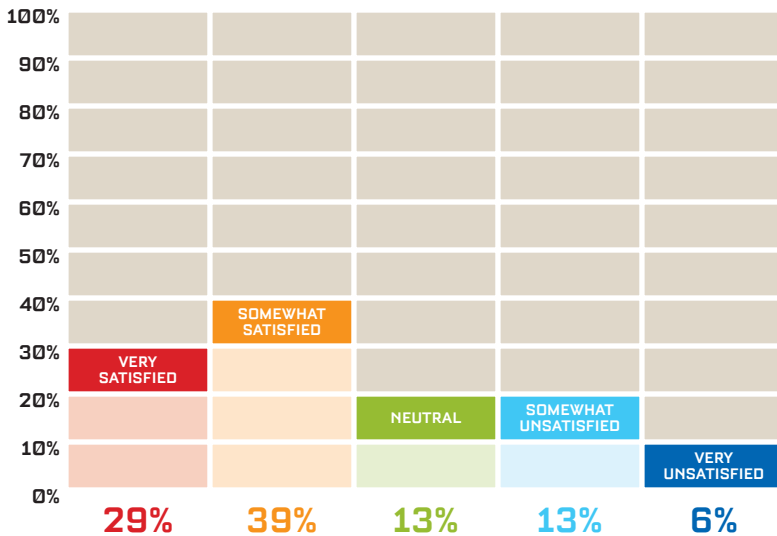
A CHECKUP FOR THE GAME INDUSTRY ¶ "GAME DEVELOPERS: HOW ARE YOU DOING?" THAT'S THE QUESTION WE ASKED APPROXIMATELY 1,000 OF YOU AT THE END OF 2012. WE KNOW THAT BETWEEN THE LONG HOURS, FREQUENT LAYOFFS, AND CRUNCH PHASES, THE GAME INDUSTRY IS A NOTORIOUS GRIND, AND WHILE WE PERFORM A YEARLY SALARY SURVEY EVERY APRIL TO CHECK THE PULSE OF DEVELOPERS' FINANCIAL HEALTH, WE THOUGHT WE'D SUPPLEMENT THAT WITH A QUALITY-OF-LIFE SURVEY TO SEE HOW YOU'RE DOING IN WAYS NOT MEASURED BY DOLLARS AND CENTS. ARE YOU SATISFIED WITH YOUR PAY? ARE YOU CONFIDENT IN YOUR CURRENT PROJECT? DO YOU WANT TO BE IN THIS INDUSTRY FIVE YEARS FROM NOW? READ ON TO FIND OUT HOW YOUR COLLEAGUES RESPONDED.

>>>

BY BRAD BULKLEY AND PATRICK MILLER



JOB & CAREER SATISFACTION



OVERALL, ARE YOU SATISFIED WITH YOUR CURRENT JOB? 29% of developers report feeling very satisfied with their jobs, 39% are somewhat satisfied, 13% are neutral, 13% feel somewhat unsatisfied, and 6% feel very unsatisfied. Job satisfaction rates didn't correlate to developer location.

"OF THE DEVS WHO WANT TO STAY, 90% OF THEM ALSO REPORT POSITIVE JOB SATISFACTION, WHILE ONLY 3% OF THE DEVS THAT WANT TO STAY REPORT NEGATIVE SATISFACTION, WHICH INDICATES THAT DEVS WILL LEAVE IF THEY'RE NOT SATISFIED."

Typical schedules During a typical week, 17% of respondents work less than 40 hours, 58% work between 40-50 hours, 16% work 51-60 hours, 5% work 61-70 hours, 1.5% work 71-80 hours, and 0.75% work over 80 hours. Canadian devs are more likely to work 50 hours or less during regular development (87%), compared to 79% in the U.K. and 72% for the U.S. and Australia.

83% of developers have a flexible schedule, while 17% do not. Job dissatisfaction rates are much higher among devs without a flexible schedule; 36% of those devs report feeling somewhat or very unsatisfied with their jobs, compared to 14.5% of those with flexible hours.

57.8% of developers have the option to work from home, and that correlates with higher job satisfaction: 75% of people who can work from home reported feeling satisfied with their jobs, compared to 61% of people who cannot. Of those satisfied respondents, those who can work from

home were twice as likely to report feeling "very satisfied" with their jobs compared to those who cannot.

Working on weekends and/or holidays appears to be rather common practice; 22% do this regularly, 31% do this only sometimes, 36% only do this rarely, and 11% report never working weekends or holidays. Interestingly enough, working weekends and holidays does not significantly affect job satisfaction levels.

Overall, devs' typical schedules seem to have a mildly negative effect on one's social life and family life; 3% report a very positive impact, 21% a somewhat positive impact, 32% report no impact, 37% a somewhat negative impact, and 7% report a very negative impact.

Compensation and benefits When it comes to compensation, 13% of developers feel they are very well compensated, 35% feel fairly well paid, 25% feel neutral, 19% feel fairly underpaid, and 8% feel very underpaid. Unsurprisingly, feeling

adequately compensated strongly correlates to job satisfaction.

42% of devs receive royalties or sales-based bonuses. Devs who don't receive bonuses or royalties are 20% less likely to report feeling any degree of satisfaction; 61% of devs without royalties or bonuses report feeling somewhat or very satisfied, compared to 81% of those with royalties/bonuses.

Benefit coverage skews positive: 27% of respondents feel very satisfied with their coverage and 29% feel somewhat satisfied, compared to 24% neutral, 10% somewhat unsatisfied, and 10% very unsatisfied. Satisfaction with benefits is directly related to overall job satisfaction, too: 85% of people who are very satisfied with their benefits also report positive job satisfaction, compared to 74% for "somewhat satisfied" on benefits, 64% for "neutral," 47% for "somewhat unsatisfied," and 41% for "very unsatisfied."

Motivation and perceived impact The vast majority of devs are very confident about

DEMOGRAPHICS & METHODOLOGY

In total, we collected 1,051 web survey respondents, referred via a Gamasutra news post, Twitter, and word of mouth, over a period of approximately one month (starting early December 2012 and ending early January 2013). The survey consisted of 40 multiple-choice questions, and participants were free to answer only the questions they deemed relevant to their development background. The demographics of the respondents broke down as follows:

Age: **4%** of respondents are 21 years or younger, **69%** are 22-34 years old, **23%** are 35-44 years, and **4%** are 45-54.

Experience: **9%** of respondents have less than one year of game development experience, **16%** have 1-2 years, **32%** have 3-6 years, **18%** have 7-10 years, **14%** have 11-15 years, and **7%** have 16-20 years.

Management: **46%** of respondents are in a managerial role, and **54%** are not.

Location: More than half of all respondents

are located in North America (approximately **50%** in the United States and **13%** in Canada), followed by roughly **16%** in Europe, with the remainder roughly equally distributed across Asia, Australia, and New Zealand, and Central and South America.

Discipline: **45%** of respondents say their primary dev role is programming, followed by **21%** design, **13%** production, **12%** art, **5%** QA, and **2%** audio. The remainder of the write-in responses mostly consists of

indie developers responsible for several roles. Interestingly, dev discipline isn't strongly correlated to any of the survey's notable findings; we're all in this together.

Studio size and type: **7%** of respondents are individual independent devs, **19%** are teams of 2-5 people, **14%** on 6-10, **18%** on 11-30, **9%** on 31-50, **7%** on 51-80, **5%** on 81-100, **8%** on 101-150, **4.5%** on 151-200, **5.5%** on 201-300, and **3%** on teams of 300+. **36%** of respondents

characterize their studios as "small indie," **25%** as "established indie," **25%** as "publisher-owned," and **14%** as "first-party."

Game platforms: **46%** of respondents work on boxed home console/PC games, **36%** on downloadable games, **20%** on social games, **17%** on browser games, **35%** on mobile (smartphone/tablet), and **10%** on handheld console games. (Respondents were encouraged to check all categories that applied.)

their ability to have a meaningful impact on a project: **40%** rate their ability for impact as very high, **35%** as somewhat high, **15%** as neutral, **6%** as somewhat low, and **4%** as very low. Interestingly enough, devs with three to six years of experience are represented in the "somewhat low" and "very low" category at more than double the rate of any other group, which hints at problems of burnout.

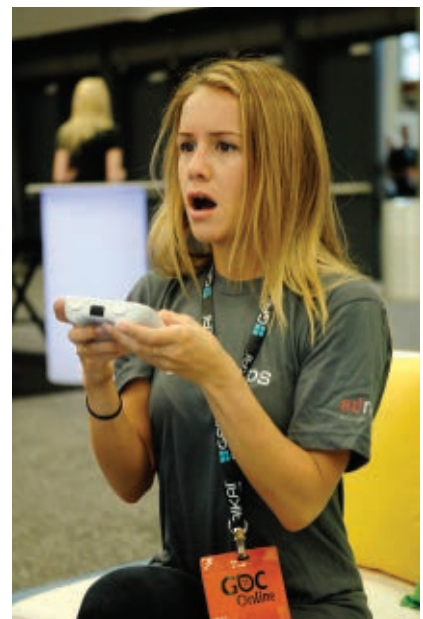
When it comes to evaluating one's prospects for advancement within the company, devs skew somewhat optimistic; **16%** rate their prospects as very high, **26%** as high, **33%** as neutral, **15%** as low, and **11%** as very low. However, respondents' ratings on their prospects decrease significantly after age 34; **47%** of the "very high" and "high" respondents are between 22-34 years old, compared to **29%** for ages 35-44, and **24%** for 45-54, which could possibly reflect a need for devs to keep current on their skill sets and/or devs generally hitting an overall career ceiling around their mid-30s.

Devs are fairly enthusiastic on their current project overall; **30%** report their level of motivation as very high, **34%** as somewhat high, **19%** as neutral, **12%** as somewhat low, and **6%** very low. Motivation correlates strongly with job satisfaction, too; **65%** of people who are very satisfied with their jobs also feel very motivated, and **60%** who are very unsatisfied are also very unmotivated. We're inclined to think that the correlation is a two-way relationship; higher job satisfaction means more motivation, and more enthusiasm for the project itself leads to higher job satisfaction. Also, **70%** of devs report that they enjoy the types of games

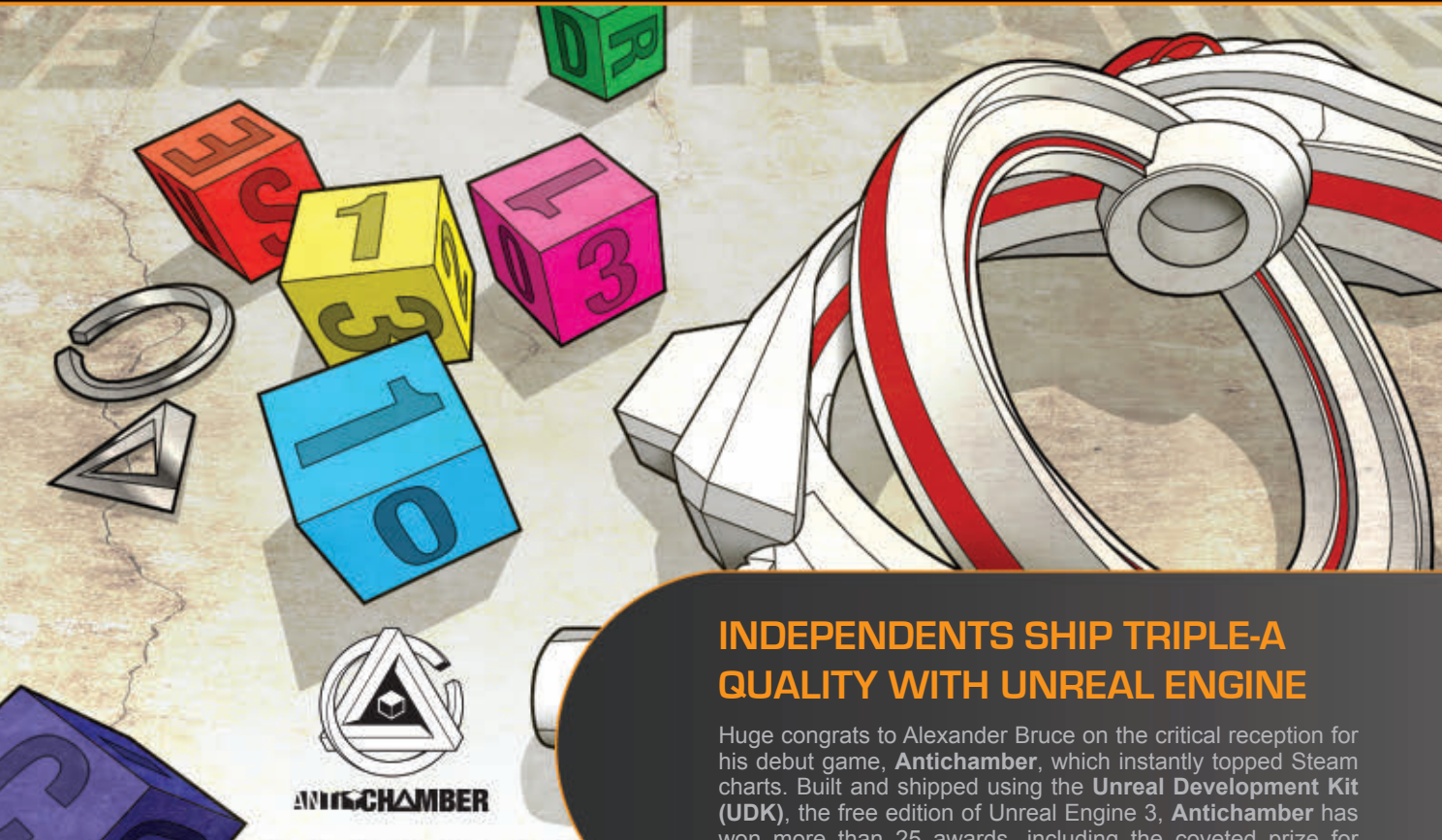
they'd compare to their current project, and of that group, **75%** report positive job satisfaction ratings (compared to **55%** of devs who report positive job satisfaction ratings despite not enjoying the comparable types of games); in other words, it's important to find devs who are already interested in the kind of games your studio is trying to make.

Employer and career satisfaction **23%** of developers expect layoffs after shipping their current project. Layoff expectations connect fairly strongly with job satisfaction rates, too; people who don't expect layoffs are more than twice as likely to be very satisfied with their job. But the fear of layoffs appears to be more prominent than actual layoff rates; for the sake of context, our 2011 Salary Survey respondents reported an actual layoff rate of **13%**.

Devs are largely split over their future at their current company; only **55%** say they want to be working at their current company in five years. Of the devs who want to stay, **90%** of them also report positive job satisfaction, while only **3%** of the devs that want to stay report negative satisfaction, which indicates that devs will leave if they're not satisfied. When it comes to devs' future in the industry, however, they are a little adamant; **89%** report that they want to remain in the game industry in five years. However, the majority of these devs are on the younger end of the spectrum; **92%** of devs under 35 want to stay there, compared to **83%** of devs 35 or older. Also, devs are split on whether to advise a friend or family member to join the industry; **62%** say yes.



UNREAL ENGINE NEWS



INDEPENDENTS SHIP TRIPLE-A QUALITY WITH UNREAL ENGINE

Huge congrats to Alexander Bruce on the critical reception for his debut game, **Antichamber**, which instantly topped Steam charts. Built and shipped using the **Unreal Development Kit (UDK)**, the free edition of Unreal Engine 3, **Antichamber** has won more than 25 awards, including the coveted prize for Technical Excellence at the 2012 Independent Games Festival.



Come talk with Epic at the **Game Developers Conference!** We will be hosting **indie drop-in hours** at our expo suite, **North Hall #BS322**. Please check unrealengine.com or facebook.com/UnrealEngine for further details. Those interested in making an appointment are invited to email licensing@epicgames.com.

GDC 13

MAKE
SOMETHING
UNREAL
LIVE
2013

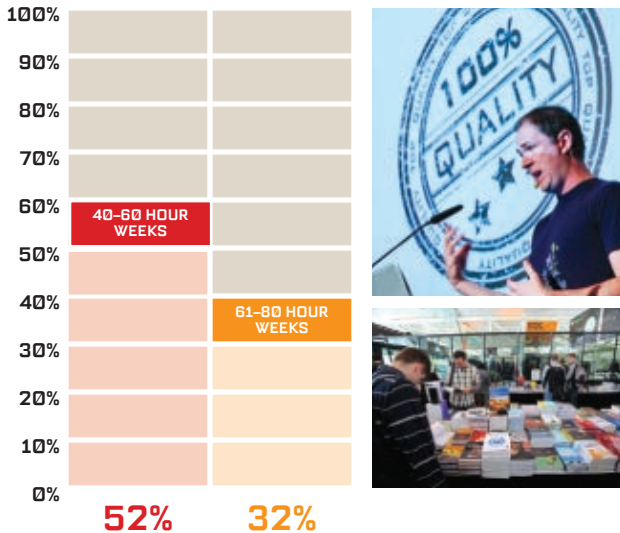
Our game development competition, **Make Something Unreal Live**, will be back at the **Gadget Show Live** consumer show in April 2013. Student teams working with UDK will rapidly build new PC experiences around the theme of "Mendelian inheritance: genetics and genomics." Thanks to our MSUL 2013 partners, the Wellcome Trust and Staffordshire University, as well as mentor studios: Climax Studios, Lucid Games, Ninja Theory and Splash Damage.

Don't forget! **It's always been free to use UDK** until you're ready to deploy a commercial product for PC, Mac or iOS. Even after the one-time \$99 studio fee, Epic doesn't take a royalty until you pocket \$50,000.



Come see Epic at upcoming industry events: **Game Developers Conference** (March 25-29, San Francisco, CA), **Gadget Show Live** (April 2-7, Birmingham, UK), **East Coast Game Conference** (April 24-25, Raleigh, NC) Email licensing@epicgames.com for appointments and sign up for our newsletter at unrealengine.com.

CRUNCH TIME



Crunch intensity and duration Crunch times vary rather wildly, according to the survey respondents; **7%** report working crunch schedules less than 40 hours/week, **25%** work 40–50 hours, **27%** work 51–60 hours, **20%** work 61–70 hours, **12%** work 71–80 hours, and **10%** work 80+ hours. These schedules rarely last more than four months; **29%** report crunch cycles that last less than a month, **30%** 1–2 months, **23%** 3–4 months, **7%** 5–6 months, **3%** 7–8 months, **2%** 11–12 months, and 3% more than a year. (One wonders at what point a yearlong crunch cycle is simply considered a typical work week.) We found that simply having crunch cycles was enough to dent reported job satisfaction, though the duration doesn't seem to affect that factor. Interestingly enough, **38%** of devs who regularly work less than 40 hours and **32%** of devs who regularly work 41–50 hour weeks do not see their hours increase during a crunch cycle, compared to **25%** for devs with 51–60 hour weeks and **7%** for 61–70 hour weeks. Essentially, the longer your regular working schedules are, the more likely you are to work even longer hours during crunch, not less—something to keep in mind next time you're asked to work longer hours during normal dev cycles in order to avoid crunch later on. Also, crunch cycles happen for all types of games at about the same rates; it doesn't matter whether you're making console games or social games, you're still equally likely to end up in crunch. Location doesn't correlate strongly with crunch duration or intensity.

Crunch impact Asked to measure the impact crunch cycles have on their social and family life, **1%** of devs respond that it has a very positive impact, **4%** report a somewhat positive impact, **17%** see no impact, **50%** see a somewhat negative impact, and **28%** see a very negative impact. In general, devs start reporting a negative impact on their social/family lives when crunch schedules exceed 50-hour weeks. Crunch cycles also have very widespread effects on devs' physical health; **9%** report a large impact, **33%** report a moderate impact, **40%** report minimal impact, and only **18%** report no impact; certainly something worth considering, especially in light of how important benefits packages are for job satisfaction.

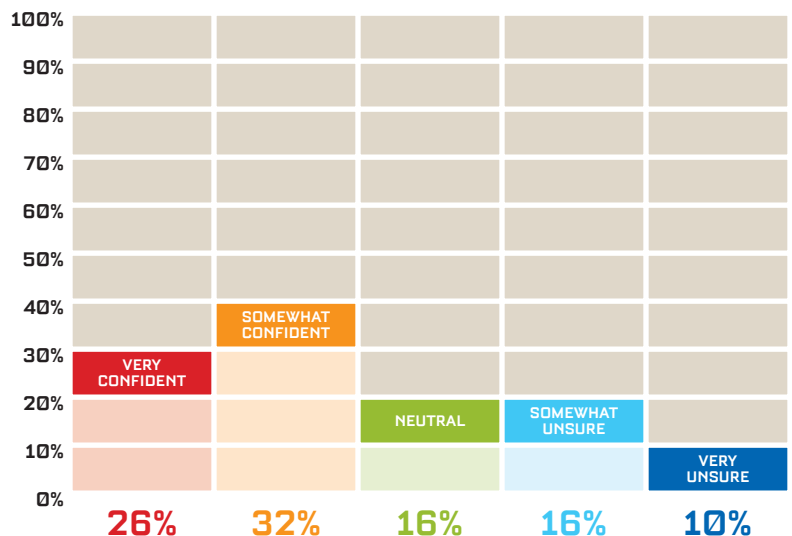
DURING THE MOST INTENSE STAGES OF DEVELOPMENT, HOW MANY HOURS DID YOU WORK DURING A TYPICAL WEEK?

Typical crunch schedules vary rather widely, with just about **52%** of devs falling between 40- to 60-hour weeks and about **32%** putting in 61-80 hours or more.

MANAGEMENT

Confidence in management Developers skew somewhat confident in their current project's management, with **26%** reporting they are very confident, **32%** somewhat confident, **16%** neutral, **16%** somewhat unsure, and **10%** very unsure. (Considering just under half the respondents to the survey identify themselves as part of the management team, we thought we'd point out that respondents in managerial roles are **15%** more likely to report confidence in management.) A whopping **91%** of respondents who are very confident in management also report positive job satisfaction, so it's clearly a very important factor for retention and morale.

Management's satisfaction Managers are twice as likely to be satisfied with their jobs, more likely to be allowed to work from home (**56%**, compared to **29%** of non-managers), more confident the product will be good (**75%** compared to **61%** of non-managers), and half as likely to report a very negative impact on their family and social life during normal dev cycles. **25%** of managers spend over 12 hours per day at home, compared to **14%** of non-managers. Overall, it sounds pretty good to be in management, though they are more likely to regularly work weekends and holidays (**59%** compared to **41%** for non-managers). Only **30%** of managers never work weekends or holidays, so if you want those managerial perks, you'll have to pay for it.

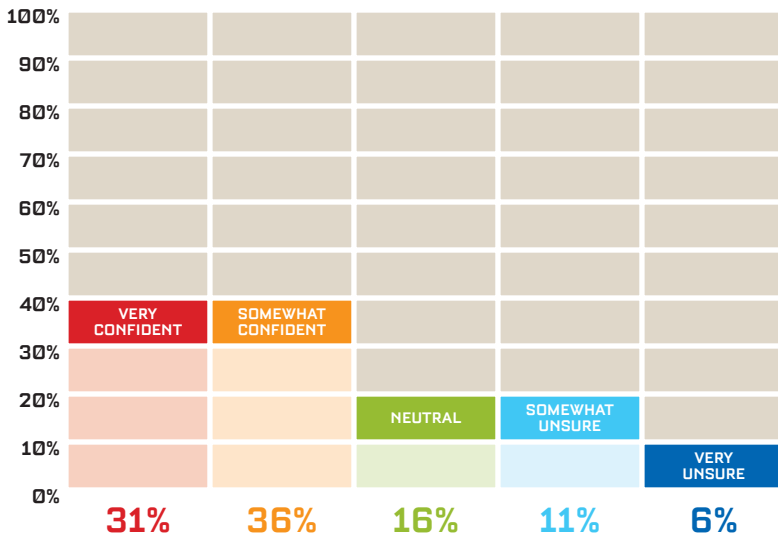


ARE YOU CONFIDENT IN YOUR MANAGEMENT'S ABILITY TO LEAD THE CURRENT PROJECT?

Overall, devs report confidence in their current project's management.



PRODUCT PERFORMANCE & QUALITY



ARE YOU CONFIDENT THAT YOUR CURRENT PROJECT WILL BE GOOD?

Developers are largely confident in the quality of their current project.

Game quality Developers are largely optimistic about the quality of their current project; **31%** report they are very confident and **36%** are somewhat confident, compared to **16%** neutral, **11%** somewhat unsure, and **6%** very unsure. **88%** of developers that report being confident in their game's quality also report positive job satisfaction.

Critical and market success When we ask developers about their perception of their last project's success in the market, they are a bit less optimistic; **25%** say they considered it to be very successful, **31%** somewhat successful, **23%** neutral, **13%** somewhat unsuccessful, and **8%** very unsuccessful. They did better with the critics, however; **28%** report that their last project was very well received, **37%** fairly well, **25%** neutral, **8%** fairly poorly, and **2%** very poorly. We were somewhat surprised to see that the estimations of critical success and market success were relatively close to each other. This could be because critics are accurately reporting product quality (and evaluating games with standards similar to those the public uses to make purchasing decisions), or because their reviews have a strong effect on sales, or possibly a combination of the two.

Connecting workload to success

Developers on 51- to 60-hour work weeks are the most likely to report their last project as a market success (**64%**), followed by devs on 40- to 50-hour weeks (60%), 61-70 (**50%**), 71-80 (**43%**), and less than 40 hours/week (**38%**). **70%** of devs who never work weekends or holidays report having successful projects, compared to only **43%** who worked weekends/holidays at any frequency. Also, about **60%** of motivated teams had successful projects, compared to **40%** for unmotivated teams.

Late to the party Game development is a chronically late business. Only **49%** of developers say their last project shipped on time, while **33%** made it less than six months late, **11%** between six months to a year late, and **8%** shipped over a year late.

Survey takeaways Overall, these survey results point to a consistent pattern: Poor product quality and performance is connected to low motivation, morale, and excessively long hours. From our perspective, these statistics stress the importance of effectively managing a project's scope and workload throughout development; long, intense crunch cycles appear to be symptoms of flawed project scoping, planning, and management.



TAKING THE INDIE DEV PULSE

With so many experienced developers deciding to start their own studios after one too many layoff cycles, we thought we'd ask: How are the indies doing?

Indie devs have half the market success rate of other devs. **34%** of indies (both individual developers and small independent studios) have successful projects, compared to **70%** for publisher-owned studios and **65%** for first-party studios.

Indies are far more likely to work less than full-time. **28%** of small indies work less than 40 hours per week, compared to **6%** of first-party devs, **10%** for publisher-owned devs, and **15%** for established indies.

Small indies are having the best of times and the worst of times. On one hand, small indie developers are far more likely to be able to work from home (**81%**, followed by **56%** from first-party devs), they're the most confident in their current project's quality (**36%** of "very confident" responses were from small indies, followed by **30%** from first-party devs), and they report that their job has the least negative impact and greatest positive impacts on their family and social life than any other dev studio type.

On the other hand, they're more likely to regularly work weekends or holidays (**36%** of devs who regularly work weekends/holidays are small indies, followed by first-party devs at **19%**), and they report the highest rate of dissatisfaction with benefits and compensation. Also, small indies have the lowest reported rate of shipping on time (**39%**); publisher-owned studios ship on time **59%**, and both first-party studios and established indies ship on time **49%** of the time.



VOX POPULI



In addition to the survey questions, we left an open comment space for the respondents to comment on the industry (or the survey) however they liked. Here are some of the responses.

" Console game development has always been great. But the social/web space I now work in sucks—I only do it for the money :-{ "

" I've basically stepped out of mainstream game production into indie games and education. I've taken a pay cut but I work at home and really enjoy the people that I choose to work with. The projects are rewarding and I'm learning new things. I believe that education is a great way to stay in touch with the new generation of people entering the industry and a perfect way to keep in touch with the wonder of working within an incredible industry."

" I'm not sure this survey fits self-employed indie devs. I'm not sure I'll make it as an indie dev but after half-a-dozen work-induced mental breakdowns at a triple-A developer before being made redundant and left unfit for full-time/proper work I don't have much choice anymore. I'll probably be dead in 18 months. Thanks industry. Thanks a bunch."

" Been wanting to get into the industry since early high school and it did not disappoint. I love this industry."

" I co-own and manage production for a studio that does not have ongoing forced overtime. We successfully deliver projects on time and on budget, so it can absolutely be done without the workplace hostility, harassment by management, and lack of basic project management skills I've seen at previous studios."

" My current title is game designer. I got into this after years of art and animation work. I'm a pretty creative person. Recently I've been tasked with gathering data, analyzing the data, creating graphs, reports, scheduling tasks, and tracking work. I have no fucking clue what I'm doing. Somehow my

job description and task are not in sync, and the work I'm doing is well outside of my skill set. Yay for my job."

" When I look around the office and notice that there are no older people working at the company, it's easy to understand why. The pace at which we work is going to burn you out until you either have a heart attack or leave."

" Let's stop the crunch and the abuses."

" My company hasn't had a real crunch in two years, a testament to better working conditions through good management."

" I would attribute unreasonably long work hours, over many years, to the recent onset of multiple, serious health problems for me. This includes incredibly painful repetitive-stress injury to both my hands, as well as back and neck problems that will require surgery."

" Got bought by a large publisher. The Eye of Sauron has moved and now we have producers everywhere making us quantify everything. I'm very concerned that this will stifle creativity and push 'polish' out so far it gets cut."

" While my work demands aren't high, the product is served to a very base audience who doesn't expect anything. A large part of my office's work is in free online gambling. It's very frustrating providing a product to a user who is solely interested in winning money, and has no interest in the content you're trying to provide."

" This is a hard job."

" I would like to see improved maternity benefits for women in the game industry. It would be a good way to reach out to the female minority."

" It would have been nice to have an industry mentor growing up." gd

Open Source / HTML5 / WebGL / Physics simulation



enchant.js is a time-saving and completely open source engine for creating games in **HTML5**.

With the number of both official and user-created plugins ever increasing, enchant.js currently supports **WebGL**, drawing in both Canvas and DOM, **2D and 3D physics**, event-driven processing, and more.

Find out just how easy HTML5 game authoring can be at

enchantjs.com

The Interactive
Fiction
Renaissance

Game Developer talks to the vanguard of
the text-game revival...

By Leigh Alexander

Copyright (c) 1981, 1982, 1983, United
Business Media. All rights reserved.
Revision 88 / Serial number 840726

Page 21

Text games have come a long way from
ZORK. Thanks to new tools, new authors,
and ubiquitous mobile devices enabling
new players, the interactive fiction
genre is enjoying a revival of sorts.

Game Developer spoke with:

- >Inklewriter dev Jon Ingold
- >longtime author Andrew Plotkin
- >indie dev and advocate Anna Anthropy
- >Failbetter Games CEO Alexis Kennedy
- >interactive-fiction pioneer Emily
Short...

about how (and why) the IF scene is
expanding.

>

[MORE]



[JON INGOLD] longtime text-game author (FAIL-SAFE, ALL ROADS), now spearheads interactive fiction innovations at Cambridge, UK-based Inkle. Notably, Inkle's new choice-oriented IF tool, Inklewriter (inklestudios.com/inklewriter), is one of the more prominent new tools designed for the kind of accessibility needed to democratize a once-niche art form. Follow him on Twitter via @joningold.

>ingold

[LEIGH ALEXANDER] *Why do you think there seems to be so much new interest in making and playing text games?*

[JON INGOLD] I think it's pretty unsurprising, given the amount of writing and reading we're all doing on the Internet these days. That's why Inklewriter is pitched the way it is: clean, simple, Twitter-like for sharing, Tumblr-like for creation. Inklewriter lends itself to making chatty, conversational pieces, and we've seen a lot of that—people using interactivity not to make a game, but to play out an argument they might have otherwise been written up on a blog [like Emily Gera's recent thing on Kotaku comments: writer.inklestudios.com/stories/bpfb].

I think people get puzzled by the difference between interactivity and games: Games are hard things to make, with fiddly rules and balancing, and most games you invent tend to fall apart because there's an easy way to win, or not enough choice. Inklewriter doesn't really support game-making, exactly: You can't make any rules. But it's uniquely good at exploring cause and effect—which is to say, telling stories. And everybody loves stories.

[LEIGH ALEXANDER] *How does the mobile and tablet space contribute?*

[JON INGOLD] I think the key thing is that mobiles and tablets mean we're all using computers more often, and more casually. Remember when it used to be rude to check your texts during a meal? Now it's normal to tweet, and not just amongst the computer-savvy crowd. Portable computing means we're all chilling out around technology a bit more, and trying things we maybe otherwise wouldn't have tried.

I saw a real example of this over the holiday season, when my mother sat down and read one of my interactive stories. She's never been able to before, because anything done on a computer is terrifying to her. But on an iPad, she felt totally safe. So there's tablet computing, expanding the size of the audience by one, at least.

[LEIGH ALEXANDER] *The accessibility of creation tools like Inklewriter helps democratize the craft of interactive fiction. What challenges does the tools space need to overcome to keep reaching more people?*

[JON INGOLD] I think the biggest challenge with developing a tool is resisting the urge to get all baroque on its ass. There's always that extra niche feature that would be so cool if it were there—but every feature you add to a tool changes the way the tool presents itself to new users, and changes a user's perception of what the tool is for. So add five cool niche features and your tool might start to look like it's for making fiddly, avant-garde things only. On the flip side, make it too simple and straightforward-looking, and no one will imagine it's capable of anything more.

So as tool creators we have to keep returning to our users and saying, what are these people like? What do these people care about, and what don't they care about? What message do we want to send them about what they should be doing?

[LEIGH ALEXANDER] *In terms of Inklewriter's potential, what are some things you hope to see people start doing with it?*

[JON INGOLD] The question floored me for a moment, and then I realized that we think about Inklewriter's potential more in

terms of who the people are than what the people do. Interactive stories have been boxed in, forever really, by the constraints that the form puts on who can do the actual writing, but I think Inklewriter can change that, at least a little, and let completely untechnical people come in and write something excellent. For us, the goal is about getting writers with unusual, rich, and diverse perspectives to invite us into their worlds.

[LEIGH ALEXANDER] *Why is it a good time for people to develop or renew an interest in text games?*

[JON INGOLD] When I started writing IF, a few hundred people on the Internet cared: A few hundred would play your game, and would discuss the ideas of game, puzzle, and story design, and maybe 10 of those were clever—or loud—enough to set the prevailing wind.

Now, if you write a piece of IF, you can get thousands of readers. You can get all sorts of feedback and discussion. You can choose between five or six ways of writing stories, all with different affordances and paradigms, and have big arguments over which is best. You don't need to learn too much that's technical [except for writing, I suppose].

But more than that: There's an optimism and a curiosity around interactive text. When I wrote my first game, I'd try to explain the merits of interactive stories to people and heads would shake. Now, they turn.

>plotkin

[LEIGH ALEXANDER] *You've been making acclaimed text games for as long as I can remember, but from where I sit there's an explosion of interest in making and playing them that seems new. What factors do you think create this resurgence?*

[ANDREW PLOTKIN] We have a big recent interest in "indie games" and "art games"—which can each mean several things, but text games and narrative-focused games play well under either banner.

Within that, or maybe next to it, we have a lot of designers trying small experimental games. Text is great for solo work; it's great for rapid production of tiny games. If you're working in a well-understood interface model, there is probably an off-the-shelf tool for you—as you note—so you can skip building a framework and go straight into your content. That's very attractive, and game designers are realizing it.

We have a gigantic wave of nostalgia for anything 15 or 20 years old. [Seriously, the last three iOS games I installed were KARATEKA, RIVEN, and LOST TREASURES OF INFOCOM. Okay, three of the last four, anyhow.]

Also, there's just momentum in tools and communities. If a bunch of people start trying a particular kind of game, it gets attention, and more people start both playing and creating in that genre. This has been building in slow motion in the IF world for several years—Inform 7 was a big boost—but it applies equally, and I think more rapidly, in other kinds of choice-based and text-based games.

[LEIGH ALEXANDER] *You were able to fund Hadean Lands via Kickstarter, are a believer in open-source tools, and will launch on iPhone; meanwhile crowdfunding, openness, and mobile opportunities are some of the most relevant trends to indie game creation in general right now. What should other creators of interactive text learn from you?*



[ANDREW (ZARF) PLOTKIN] is among the most beloved and longest-serving authors in the IF community, creator of popular titles like SPIDER AND WEB and SHADE, among numerous others, in addition to his many contributions to the community's tools and infrastructure. In 2010 he made headlines when he raised over \$31,000 via Kickstarter for the creation of his next game, HADEAN LANDS, an impressive demonstration of the strength of the IF community and of the gratitude for his work.

[ANDREW PLOTKIN] Oh, geez. I don't know if I can answer that. None of those trends are simple answers, and I don't know if I've found the right path through any of them.

[LEIGH ALEXANDER] *Was your fundraising lightning in a bottle, or do you see a wider commercial opportunity for creators of interactive text on mobile?*

[ANDREW PLOTKIN] My Kickstarter project was definitely a thing of its moment—in relation to Kickstarter's history and mine. It got attention for being notably successful, but the stakes for "notably successful Kickstarter" have moved way, way up. And I deliberately offered a wider range of work than just "IF on mobile."

Really, Kickstarter successes don't signpost commercial opportunities—commercial successes do that. It's the new games and the interest in new games which should be drawing everybody's attention.

[LEIGH ALEXANDER] *There are an increasing number of uniquely accessible tools arriving to help new developers make choice-oriented or hypertext-style games. This brings more creators to the medium, but also seems to suggest a shift away from the traditional text parser and its associated strengths and challenges. What are your thoughts on that?*

[ANDREW PLOTKIN] My thoughts are ambivalent, as you might expect! On the one hand, people tend to lump the games together. More interest in any of these forms is more interest in all of them, and that's good for me.

But on the other hand, people tend to lump the games together—and parser IF does have its own strengths! You can do a lot of things with a menu system, but you can't graft it onto a game like ZORK—or SHADE, or SPIDER AND WEB—and expect it to play out the same way. You need to design your game to fit the model. So I have to worry about whether players are going to wind up just not very interested in the games I want to make.

The sensible answer is "Make the games first, then decide." I realize this. But I indulge in a little worry anyhow.

[LEIGH ALEXANDER] *Would you agree that IF is less "niche" than it was 10, even five years ago? How has the community changed?*

[ANDREW PLOTKIN] IF is still "niche," but niche-ness is much less niche these days! Niche is practically mainstream. Or at least, people are much more willing to poke their noses in.

>anthropy

[LEIGH ALEXANDER] *You've become a big advocate for Twine because of its accessibility. How do tools like that help you in your mission to help motivate new voices in game development?*

[ANNA ANTHROPY] Twine works for three big reasons: It's free, it's not programming, and finished stories are webpages you can plunk onto the Internet. That solves the three big historical barriers to nonprofessionals who want to get involved in game-making: the cost, the skill barrier, and distribution. All of these are huge deterrents. In our society it's middle-class men who are given the most opportunities in tech fields—who can afford to go to college, who aren't weeded out by a famously sexist culture—and ultimately those are the people who end up best-equipped to deal with the barriers to game-making in the traditional way.

[LEIGH ALEXANDER] *What does it mean that there's now a community for individually voiced outsider art?*

[ANNA ANTHROPY] It means amazing art all the time. This morning I played a game about wandering a surreal psychosexual dream world while taking a nap next to a hot older man on an airplane. There's a game like this every week. Last week it was someone's interactive memorial for his brother who had just died, and a game about someone's experience as a bisexual woman being shamed by an online lesbian community. It's hard for even me to keep up with. I've thought about retiring my blog; the videogame community I've always wanted is blossoming around me, and it looks so different from the mainstream. Here's a face of videogames whose architects are women and queer people, speaking in a thousand voices.

[LEIGH ALEXANDER] *How have you seen the audience and opportunity for interactive text evolve in recent years?*

[ANNA ANTHROPY] A few years ago "interactive fiction" was an insular group of [highly literate] nerds sitting around and making games about each other. That interactive fiction scene was very inward-looking: It was all about parser-based stories—you type what you want to do, the game responds—which meant that huge barriers to accessibility still existed, both for creators (making a game for a parser is programming) and for players (the language that the game understands is hidden, and has to be learned, presumably from other players).

Interactive fiction now, with hypertext at its center, is outward-looking and outward-expanding. Hypertext is immediately accessible to people who haven't spent the time to learn the vocabulary of the games status quo.

[LEIGH ALEXANDER] *Do you think increasing interest in making text games (presuming you agree such a thing exists!) reflects more audience appetite for storytelling and more sophisticated themes?*

[ANNA ANTHROPY] Hypertext retains the purposeful, deliberate ambiguity that makes text games a place suited toward exploring themes like social interaction, identity, sex, feelings—all the stuff mainstream games seem so poorly equipped to tell us about. Twine's explosion was a sure sign that people have been wanting to find ways to interrogate these themes through games, but they weren't able to find a means.

[LEIGH ALEXANDER] *What do text games do that other games can't, and what do you think traditional developers should learn from the current IF community?*

[ANNA ANTHROPY] How about: Don't be such fucking cowards. While mainstream games like SPEC OPS: THE LINE and HOTLINE MIAMI are tiptoeing up to the idea that maybe violence is something we should be worrying about while continuing to let the player inhabit the role of an armed dude acting out fantasies of violence, Twine games are talking about identity, alienation, abuse, sexuality, dysphoria, sexual assault, depression, self-discovery, loss, and D/s dynamics in the cyber-future. Look at these games and be ashamed of how small you've allowed your world to become.

G A M E | D E S T I N A T I O N :

B L A C K B E R R Y | 1 0

It's where your
game belongs.

BlackBerry® 10 offers a powerful and easy platform for game development. It's integrated with major development tools and leading game engines, including Unity, Marmalade and Shiva 3D. Plus, the leading BlackBerry 10 hardware produces a visually stunning and incredibly immersive gaming experience that really lets your masterpiece shine.

Learn more

developer.blackberry.com/games

 **BlackBerry**®

© 2013 Research In Motion Limited. All rights reserved. BlackBerry®, RIM®, Research In Motion® and related trademarks, names and logos are the property of Research In Motion Limited and are registered and/or used in the U.S. and countries around the world. Used under license from Research In Motion Limited. All other marks are the property of their respective owners.

Images courtesy of SHADOWGUN, by MADFINGER





[ANNA ANTHROPY] Indie powerhouse Anna Anthropy's recently published book, *Rise of the Videogame Zinesters*, is something of a manifesto for game creation as individual self-expression, and the accessibility and flexibility of choice-based text-game creation tool Twine makes her a big fan. Since she started encouraging friends and allies in the indie community to try making Twine games, a vibrant homebrew community has sprung up—notably, there's a passionate and

[LEIGH ALEXANDER] *Who are some IF makers you are excited about these days?*
[ANNA ANTHROPY] porpentine ~ merritt kopas ~ maddox pratt ~ kim moss ~ j chastain ~ madamluna ~ lydia neon ~ kitty horrorshow ~ christine love ~ jonas kyratzes
[i] Memorial: <http://www.theautumnalcity.org/Memorial.html>
[e] CYBERQUEEN: <http://aliendovecote.com/uploads/twine/LD25/CYBERQUEEN.html>
[s] Reset: <http://www.lifeinneon.com/games/reset.html>

>kennedy

[LEIGH ALEXANDER] *Why do you think there seems to be so much new interest in making and playing text games, and how does the mobile and tablet space contribute?*

[ALEXIS KENNEDY] Technologically, there have been attempts in the past to extend literature—hypertext, interactive fiction—but the gap between the page and the screen was just too wide for mass adoption. Now readers are accustomed to text on a screen, thanks to mobile devices and e-readers, and to the degree of interactivity that comes with it. Even if you're just reading a Kindle book, you can share or search for phrases from right there in the interface, and that comes to seem natural. So it's easier to extend or colonize the borders of fiction.

Culturally, it's the mainstreaming of geek. The success of big fantasy and SF franchises, and the tsunami of casual gaming, means that acceptance of game-like activities is filtering out through the demographics.

[LEIGH ALEXANDER] *The accessibility of creation tools like StoryNexus helps democratize the craft of interactive fiction. What challenges need to be overcome in the tools space to keep reaching more people?*

[ALEXIS KENNEDY] The big barriers are "how the hell do I get started?" and "what in the name of God have I done?" Choose Your Own Adventure-esque branching path narrative is great for the first, but deadly for the second: It's easy to get started, but it's also very easy to write yourself into a thousand-branch hole. Programming languages (e.g., ChoiceScript or Inform) are the opposite—a big initial learning curve, but a saner experience once you get started. Tools like StoryNexus find a safe route between the two.

A secondary issue—but one that's important as the space grows commercially—is distribution. You need some sort of mediating technology to read an interactive story. The web is good for this, but finding a way to earn revenue from interactive writing can be hard, compared to the well-established channels for content publishing elsewhere. StoryNexus is trying to bridge this gap, too.

[LEIGH ALEXANDER] *It seems the text space these days is geared largely at moving away from the parser—and the accessibility barriers associated with it—and toward interactive reading and choice-based interfaces. What opportunities might this present for designers and game developers?*

[ALEXIS KENNEDY] The traditional text-game approach is

a question-and-answer dialogue—"OK, what do you do now?"—and largely a synchronous one. We haven't seen much of what happens with asynchronous and out-of-band gaming. On the StoryNexus front, this could mean characters who occasionally email you and ask for a response, or a virtual life where you experience weekly events, or a serialized story. Above all, it means digestible games that can be consumed in small chunks—that occupy the same niche as web comics, perhaps. In turn, this means wider audiences and more room for experimentation.

[LEIGH ALEXANDER] *It seems you've focused for some time on the social experience around interactive storytelling. Who do you see as the audience for gaming in this way, and what does the community element add to what a lot of people view as a solitary pursuit?*

[ALEXIS KENNEDY] The immediate act of reading is a solitary one, but the context of reading is a shared one, especially when we're reading about imagined worlds. Doyle, Tolkien, Pratchett, Rowling—all of these attracted passionate fans who wanted to revel in the shared world together, solving the mysteries, arguing over the characters, or just being in the space. That's what community gives a text game—other people to energize and validate your own experience.

[LEIGH ALEXANDER] *Why is it a good time for people to develop or renew an interest in text games?*

[ALEXIS KENNEDY] The technological and cultural changes I mentioned above have led to an upsurge in text games—particularly in creative toolsets, which in turn means the range of titles is expanding beyond SF, fantasy, and horror. It's the indie-gaming revolution in miniature.

But in some ways, literature is more like gaming than most other creative forms. Through an accident of technology, games have a strong family resemblance to film, but film is a much more passive medium than literature. The plot in a book doesn't advance until you turn the page. Books and games are both demanding, participatory forms.

>short

[LEIGH ALEXANDER] *What does the IF space have to teach other developers?*

[EMILY SHORT] A vocabulary of interactive narrative. Authors in this space have spent a lot of time, experimentation, and virtual ink on the topic of what player participation does to a story. We have a lot of examples, and a lot of conversations about, games that convey their meaning through mechanics; environmental storytelling, and stories that are partly about how the reader chooses to discover them; complicity, ethically challenging choices, expressive choices, choices that turn out to be meaningless in retrospect, choices that don't change the events of the story but revolutionize the way you understand that story. Being steeped in all those techniques is a great craft advantage, whether you're writing something text-based or not.

This is not to say that there's no sophisticated thinking about narrative in triple-A games—of course there is. But IF has been a very productive venue for experimentation. I recommend the Failbetter blog and the Inkle studio blog, as well as the IF Theory Reader, as sources.

[LEIGH ALEXANDER] *Have you seen the opportunity and*

OUR [ANIMATORS]
LEAP FIRST

HEROES
WANTED



CONNOR
18TH CENTURY'S AMERICA
PROFESSIONAL ASSASSIN



JONATHAN COOPER
UBISOFT MONTREAL
ANIMATION DIRECTOR



UBISOFT®

JOBS.UBISOFT.COM



[ALEXIS KENNEDY] Failbetter Games is founded on a legacy of passion for interactive stories; the studio is best known for its massive choice-driven online role-playing story *FALLEN LONDON*, but it also plays host to the StoryNexus platform, a browser-based story-game creation tool that even enables writers to monetize their work and build community around it. CEO Alexis Kennedy has long been attracted to the junction of game design and writing.

audience for interactive text evolving in recent years?

[EMILY SHORT] Absolutely. We're seeing traditional publishers becoming interested in interactive eBooks and interactive narrative that goes beyond just adding some footnotes or multimedia features to a traditional text, transmedia projects that incorporate several different kinds of production and might include an interactive text component, and Twine and other text games produced by indie communities who never considered themselves part of the "interactive fiction" community.

Several things have happened: One, the barrier to entry of writing some kind of interactive story is as low as it's ever been, and it's easier than ever to make those stories available to readers. It sounds ridiculous now, but 10 years ago we used to have despairing conversations about how we'd never reach a bigger audience because it was economically infeasible to put interactive fiction in a box at a store.

Two, IF has benefited a bit from the rising visibility of indie games in general, which means we have more contact with adjacent but not identical genres and it's easier to get people who might not be longtime text-adventure devotees to play text work. And that also makes a difference to what IF authors think of writing, not for technological reasons but for cultural ones.

[LEIGH ALEXANDER] *The accessibility of new creation tools helps democratize the craft of interactive fiction. What challenges need to be overcome in the tools space to keep reaching more people?*

[EMILY SHORT] Communication about what tools already exist, and better development into spaces that are genuinely unexplored, instead of recreating the same old thing. I regularly get email and messages from people saying something like "Oh, hey, I'm making a CYOA tool. Wouldn't it be great to be able to write your own choice-based games?" and I feel a little bit guilty writing back with a list of all the tools that I know are already in that space. (And I'm sure I don't know every single thing in the field.)

Polish and style. A creative tool is this incredibly intimate thing. It becomes an extension of the creator, almost an extra limb. As important as any technical capacity is how much the tool appeals to the user, how naturally it fits. Not every interactive narrative tool is going to appeal to every user, which is a strong reason to have a rich ecology of tools. But a lot of creators are put off by form-factor issues that the tool creators might not have considered at all.

Community. Any kind of sophisticated tool needs a support community, people to give advice to novices and help them over the hills.

Good examples. Any new interactive storytelling platform or tool badly needs at least one cool, compelling work to help new users understand what that platform is capable of. Launch without that, and it's a lot harder for people to understand why they should care

or what the affordances of the tool will be.

Publicity. Some of these tools are marketed as game-creation tools exclusively, even though they'd be interesting to people who don't think of themselves as traditional gamers at all, much less traditional game designers. There are lots and lots of applications for interactive narrative—educational, literary, journalistic, or nonfictional—and continued growth requires that we reach across cultural divides.

[LEIGH ALEXANDER] *Do you think other companies will start consulting or looking to hire text and conversation game writers?*

[EMILY SHORT] That's already happening. I get asked on a fairly regular basis for referrals to people with interactive text experience.

[LEIGH ALEXANDER] *What are your thoughts on the current state of the IF community?*

[EMILY SHORT] The biggest point is that there is no longer "an" IF community. In the early 2000s, that phrase mostly referred to the set of folks writing and playing parser-based text adventures and talking about them on Usenet. Now there are a lot more folks involved, and they're not all talking through the same venues. There are Twine authors and ChoiceScript authors who are coming from a different background and social community than a lot of the Inform and TADS, Hugo, ADRIFT, or Quest authors.

Another point is that IF doesn't all look the same any more. I used to hear a lot of complaints about how IF in, say, 2008 looked the same as it did in 1982—blocky text in a little console window—and that conveyed all kinds of negative things about production quality. No matter how much narrative sophistication or improved programming there might be under the hood, that little window of blocky text was suggesting to potential players that they were still looking at ZORK.

Now, though, we're seeing IF that looks like *Guided Youth*, like *Living Will*, like *howling dogs*, like maybe make some change, or *Ex Nihilo*, inkle's *Frankenstein* novel, or StoryNexus's *Zero Summer*. Some of those are beautiful, some are provocatively frenetic or disturbing, but they're not identical.

[i] *Guided Youth*: <http://ifdb.tads.org/viewgame?id=1dytdvbtbgxuwx0h>
[m] *Living Will*: <http://ifdb.tads.org/viewgame?id=cwblhjdjih48v94>
[d] *howling dogs*: <http://ifdb.tads.org/viewgame?id=mxj7xp4nffia9rbj>
[e] *maybe make some change*: <http://change.textories.com/>
[u] *Ex Nihilo*: <http://ifdb.tads.org/viewgame?id=m5vqmisz4y38o5tz>



[EMILY SHORT] The work of renowned IF pioneer Emily Short has tended to focus on plausible interaction with artificial characters; her company, Little Text People, was acquired in recent years by Linden Lab in part because of her leading work in the field of social simulation.

CREATE YOUR FUTURE.

Sheridan's innovative, intensive digital media programs offer a great opportunity to gain the skills needed for careers in many exciting, highly competitive industries.

Sheridan offers programs taught by passionate, industry-savvy faculty in first-rate facilities, all backed by Sheridan's world-class reputation in arts education.

- Bachelor of Game Design
- Bachelor of Interaction Design
- Game Level Design (post-grad program)
- Game Development – Advanced Programming (post-grad program)



Sheridan

Visit us at booth #321 at the Game Developers Conference in San Francisco, March 25 – 29, 2013.

**APPLY TODAY AND TAKE
THE FIRST STEP TOWARDS
YOUR FUTURE!**

sheridancollege.ca



Tim Golem



Kristian Howald



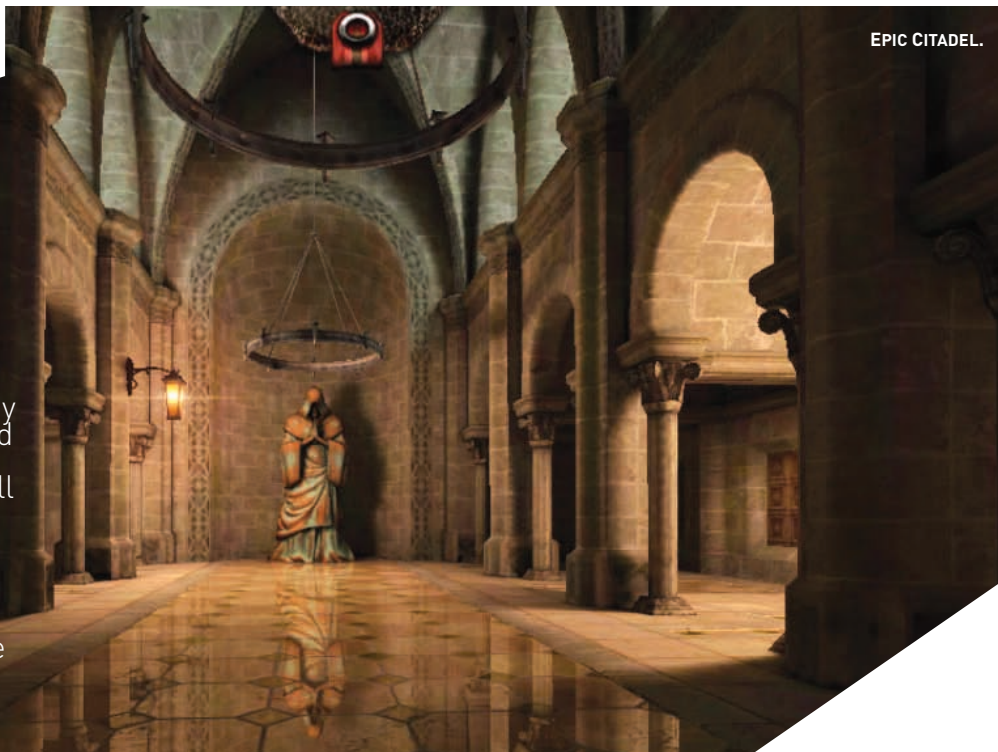
Lexi Young



Xiaoyu Wang



“We came into 3D game development really seat-of-the-pants. When id Software created DOOM, I looked at that and said, ‘Oh my god; they’ve invented reality. I’m giving up as a programmer. I’ll never be able to do that.’ But, over the next few years, as they started to build QUAKE, I started to think, ‘Hmm, maybe I can figure out this texture-mapping stuff.’”



EPIC CITADEL.

[BRANDON SHEFFIELD] *Let’s start off with a big sandbox question. What do you see as the next big thing the industry needs to tackle graphics-wise and computationally in games?*

[TIM SWEENEY] Oh, wow. The industry’s advancing on a bunch of fronts simultaneously. One is just advancing the state of the art of lighting technology. We’ve really added to that with sparse voxel octree global illumination stuff—basically, technology for real-time indirect lighting and glossy reflections.

That’s really cool stuff, but it’s also very expensive and only suited for the highest-end GPUs available, so having a family of solutions for that that scales

all the way down to iPhone with static lighting is a big priority for Epic—the ability to go to a game that scales in a dramatic factor from low-end to high-end. We’re really concerned—I think it’s our number-one priority, really—with productivity throughout the whole game development pipeline, because we’re looking at companies like Activision spending \$100 million developing each new version of CALL OF DUTY, and that’s insane!

We can’t afford that sort of budget, so we have to create games with fewer resources. Any way we can tweak the artwork pipeline and the game scripting pipeline to be able to build core games more quickly

with less overhead is better. We put a lot of effort into visual scripting technology to greatly improve the workflow.

[BRANDON SHEFFIELD] *It does look a lot more intuitive for a less tech-savvy person.*

[TIM SWEENEY] With Unreal Engine 4, we really want to be able to build an entire small game on the scale of ANGRY BIRDS without any programming whatsoever, just mapping user input into the actions using a visual toolkit. This technology will be really valuable.

We’re also expanding the visual toolkit for everything: for building materials, for building animations, for managing content when we have a huge amount of game assets. We’re just greatly simplifying the interface so that it’s basically as easy to use as Unity.

On one hand, you have the Unreal Engine having by far the largest and most complete feature set of any engine, but also with Unreal Engine 3 it was a big, complicated user interface. With Unreal Engine 4, the effort is to expose at the base level everything in a very simple, easy-to-use, and discoverable way and to build

complexity on it so that the user can learn as they go without being terrified by it in the form of a huge, complicated user interface. That’s a problem all applications have to deal with nowadays. If you look at an iPad app that does 90% of what the world needs really easily, versus a Windows version of the app for something like Photoshop—I spent an hour and couldn’t even figure out how to draw a picture in Photoshop; it’s that bad. (laughs) There is a lot to do there and a lot to learn from.

[BRANDON SHEFFIELD] *Going back to voxels, I’ve always been kind of fascinated by them because they’re less expensive with higher fidelity potential, but you can’t texture them and you can’t really animate them well. Do you ever foresee a future in which that might be possible? It would be a total industry shift away from triangles, but...*

[TIM SWEENEY] It’s clear now that voxels play a big role in the future.

[BRANDON SHEFFIELD] *Certainly for lighting, right?*

[TIM SWEENEY] Well, that’s just one way we’ve been finding to use them effectively. [John] Carmack did a big write-up



FORTNITE.

about voxels and the virtues of sparser representations of the world. It seemed crazy to me at the time, but now it's becoming clearer that he had a lot of far-ranging insight there.

The thing about voxels is they are a very simple, highly structured way of sorting data that's easy to traverse. Whereas any other form of data, like a character stored as a skinned skeletal mesh, any time you

want to traverse it, you have to do a gigantic amount of processing work to

transform it into the right space. You need to figure out what falls where and rasterize it or whatever, but voxels are just efficiently traversable. I feel like there's a big gap.

The data representation you want to use for rendering your scene differs greatly from the representation you want to use for manipulating your scene and basically moving objects around and choosing how they interact. It's not clear which representation wins. In the sparse voxel-oriented approach, one neat thing is we can update it dynamically; as objects move around, we can just incrementally change those parts of the voxel octree that are relevant. You can figure out the extreme edge case of the algorithm, so that instead of the voxel octree being fixed with its orientation to space, all you have to do is align it to screen and arrange it objectively so you're seeing a 2D view of this voxel octree, whereas the other dimension is your z dimension.

You basically have this projective voxel octree, stick everything in the entire scene into it, every frame, and then that unifies all of these screen space techniques like screen space ambient occlusion with the large-scale world effects that we're using with the sparse voxel octrees. Imagine rasterizing your entire scene directly into a representation like that—using that for real-time lighting and shadowing and then rendering that result out the frame buffer. It's hard to say whether that has merit;



GEARS OF WAR 3.

that's an algorithm where you need 20 or 30 teraflops as opposed to one or two.

[BRANDON SHEFFIELD] *It seems like we've gone so far in the polygon direction that it would take a significant spend and a lot of research to try to push in a different direction.*

[TIM SWEENEY] Yeah. Polygons are nice for representing your scene that the idea that you can go with this smooth, seamless mesh that has nice properties like being planar... I have a hard time seeing the world moving away from that. And if you think about something like a skeletal animated character, trying to represent data like that in a voxel representation would be

hopelessly inefficient. Say you have two fingers and want to animate them independently, but they're close enough that they probably share some voxels in common; you want a polygon representation so these objects can be independent and move independently with no relationship between them. So I think polygons ultimately will always be our working scheme representation, but our rendering scheme representation is where you're seeing a lot of this innovation.

You can really look back and say, a-ha! The industry started inventing these techniques five or six years ago with screen space techniques like ambient occlusion, and now we're

starting to realize that, instead of just doing that in screen space, you can also do it in a voxel representation of the world. They have in common this very regular structure that's easy to traverse. I think we're going to see enormous innovation in these areas over the next 10 years because, with DirectX 11, you now have all this power of general-purpose vector computing hardware, and you can use a few teraflops of performance in traversing an arbitrary data structure. In the previous generation, we had to rethink everything in terms of pixel shaders and vertex shaders, and that rules out most of these techniques. So the next few years are





Prepare for the adventure of a lifetime!

See your game ideas come to life in Berlin.



We're currently looking to fill these positions:

- Product Lead**
- Game Analyst**
- Mobile Engineer**

Check out all our open positions at wooga.com/jobs

GPU TECHNOLOGY CONFERENCE

MARCH 18-21, 2012
SAN JOSE
CONVENTION CENTER

SUPERCHARGE YOUR GAME DEVELOPMENT SKILLS

NVIDIA's GPU Technology Conference has expanded to include cutting edge mobile and game development sessions. Spend time with Valve, Gearbox, and NVIDIA's game development engineers in a friendly, fun and collaborative environment. Come learn first-hand about the benefits of a partnership with NVIDIA and walk away with the latest development techniques!

Register at www.nvidia.com/gtc. Use 10% discount promo code **GM10GDM**.

Thank you to our sponsors:





going to be very interesting. I suspect that a lot of developers will look at this from different perspectives and come up with entirely new techniques that we're not anticipating now.

[BRANDON SHEFFIELD] *What do you think about Carmack's solution to textures with id Tech 5, where every texture has to be specifically drawn?*

[TIM SWEENEY] It's a neat idea that's kind of the extreme Carmack way of doing things, the brute-force, completely general solution to the problem. Some artist is going to want to customize any particular part of an object, and that makes it really easy to go in and paint over and customize any part of a scene to an arbitrary degree. That is certainly the general solution, and it's a very brute-force algorithm that gives you regular performance regardless of scene complexity, and gives you regular performance for streaming, which is really important.

I personally tend to be on the other end of the spectrum, which is that, rather than going

at everything brute force, we want procedural authoring techniques to enable content developers to build small enough content and then amplify it by instancing, reusing, scaling, rotating, and customizing objects with minimal amounts of work. In GEARS OF WAR, you see that the same mesh was used maybe 20 times in different places. I think those techniques tend to be the best solution to the overall game development productivity problem because we want to go and create games with minimal effort. Although they're not as flexible, they do give you the greater amplification of both data and development effort, and, to me, that is always going to be a really desirable property.

You can envision the two ideas being combined. The thing that's awesome about Carmack's megatexture approach is that it gives you the freedom to customize everything. The drawback to it is it gives you the cost of everything being customized, even if it's not, because you have to store everything uniquely. What I would like to do is basically represent

the world in some sort of hierarchical, wavelet-inspired way so that data layers that are zero or haven't been changed relative to their base aren't stored at all, and therefore don't incur any cost.

The idea is that you want to build a cool door façade and use it in 10 different places, and in three of those places you go in and do a custom thing on top of it. You're storing the data there, but you're only incurring the cost for the areas where it's actually being customized. Similarly, the idea with wavelets is you can paint at multiple resolutions and you can do perturbed geometry at multiple resolutions, so you might have a beautiful car mesh that's used in a bunch of places; then, in some places, you have it wrecked because you destroyed the high-level vertices or the lower-level detail versions of it, but the rest of the content is still there. So, with a little bit of data, you've made a really interesting procedural modification to an object that already exists. I think nuanced techniques like that will win in the long run because they give you the full customizability of the Carmack approach without the full cost of using it everywhere.

[BRANDON SHEFFIELD] *I feel like there aren't really a lot of companies that are taking such a long-term view of things—thinking five, 10, or 20 years ahead technology-wise, especially in games and also in terms of business models. Epic is thinking about that with Unreal Engine 4, but also with INFINITY BLADE being the company's most to-scale profitable game. How have you been able to take this view, where a lot of others aren't thinking past what they're going to put out next year?*

[TIM SWEENEY] Well, being both an engine developer and a game developer forces us to think further ahead. Our big success with the engine came in the third generation, and that was because everybody else had just been finishing their previous-generation games, while we'd been building Unreal Engine 3 three years prior to Microsoft or Sony actually choosing their hardware for

the generation. So we had this enormous investment that we'd already built up, and as soon as the industry moved over from that we were ready for it and everybody else was years behind. That gave us a huge advantage.

We've been doing that informally all along—really trying to think ahead—but that experience made it an explicit and clear part of our strategy. We need to stay ahead of the rest of the industry so we can be there with tools and technology when people need them. People don't realize they need them until they need them. So we must realize they'll need them three years before they need them, so that we can actually build them. That's been the challenge.

We have great relationships with the hardware companies—Intel, Nvidia, and Apple. With the pure hardware manufacturers like Intel and Nvidia, we can talk about our roadmap with them and they can talk about their roadmap with us three, four, or five years out, and we can really line everything up. It's what's necessary. We have to start thinking of technology developers like Epic, Crytek, and Unity a lot like the hardware industry thinks of itself. When Intel ships a CPU design like the new Sandy Bridge or whatever, it first envisioned that architecture seven years ago. They had to, because that's their development cycle and they have to go through this long series of processes to develop that from scratch to ship. We need to have a lot more engine development work in the pipeline. UE4 was in development simultaneously with UE3 for three years or more, plus a longer research cycle than that in advance. It's just necessary for survival and continuing to lead the industry forward.

[BRANDON SHEFFIELD] *So it was a conscious choice to be ahead of the game with Unreal Engine 3, but earlier, when you realized people needed your technology, was that a tipping point and a change in Epic's mindset? Were you initially envisioning it as this service, essentially, that was going to be sold to people, or was it like, "No, we need to build this game, and this is what we need*



to build it; so we're going to go that direction."

[TIM SWEENEY] We came into 3D game development really seat-of-the-pants. When id Software created DOOM, I looked at that and said, "Oh my god; they've invented reality. I'm giving up as a programmer. I'll never be able to do that." But, over the next few years, as they started to build QUAKE, I started to think, "Hmm, maybe I can figure out this texture-mapping stuff."

With the first generation of Unreal Engine, we went in not really intending to build an engine so much as build a game, and the engine was a byproduct of that effort. Then we were a couple of years into development when a couple of developers caught us up and



BORDERLANDS 2.

said they wanted to license our engine, and we were like, "Engine? What engine? Well, I guess we have an engine." The whole engine business at Epic was a completely customer-driven idea. As it's evolved, it's become a much more serious effort. More than 40 people are contributing code to Unreal Engine 4. That's a huge effort. It's a team worldwide who works for customers providing support and developing features in Japan and Korea and China and Europe.

We're creating a real significant global business, working closely with all of the hardware companies to determine the roadmap as much as we can. Roadmaps then line up with working with customers to work out various conflicting requirements between different markets and desires. It's a very serious, real business now, completely different than it was a few generations ago. I wrote a quarter-million lines of code on Unreal Engine 1—I wrote about 80% of the code myself. What could I do now being one person out of 40?

[BRANDON SHEFFIELD] That makes me curious—how much day-to-day coding do you actually get to do?

[TIM SWEENEY] I spend at least a few hours a day, but right now I'm not critical path on anything like I was on Unreal Engine 1. My schedule is too unpredictable to contribute to that, but I really try to stay on top of it and talk with all the key guys who are architecting the major systems.

[BRANDON SHEFFIELD] It seems like in some companies—this is especially a Japanese problem—people get pushed up and out of doing stuff and into having meetings about doing stuff instead. It's good you've avoided that.

[TIM SWEENEY] Yeah, we've really put a lot of effort into making sure our key folks at Epic are able to do what they are best at. There are some world-class programmers at Epic who are never going to be leads because they are far more valuable at inventing new ideas than coordinating the efforts of the people who do that. There's a very different set of talents required for leadership versus more individual contribution. It's very important that you recognize the distinction between the two and realize what each person is really best at.

[BRANDON SHEFFIELD] Of course, it's also important to compensate accordingly if you really need someone in that non-leadership position.

[TIM SWEENEY] Some of Epic's most valuable people aren't in leadership roles.

[BRANDON SHEFFIELD] These days, it feels like there are not very many people trying to push graphic fidelity forward. There are a lot of people who are more concerned with business models and things than they are with graphical fidelity and stuff. Do you feel some kind of pressure to push graphics in the next phase of game evolution? There's you, there's Crytek, DICE, possibly id... Who else is going to fight for graphics over convenience?

[TIM SWEENEY] Well sure, if you look at EA's DICE studio with BATTLEFIELD and Activision with CALL OF DUTY, they're

certainly making major investments in graphical quality. I think that's a general goal of the major Western developers—at least developers of major shooter franchises—to really push the graphical line.

It's an interesting distinction; when you talk to Asian developers, the overall focus is more on maximizing the customer experience than on maximizing the graphics. A lot of the companies out here are decades ahead of us in that area; every day they look at the stats of what users are doing, whether they're getting stuck, what things they're buying, what things they're not enjoying. They gather massive amounts of data and use it to tweak the games constantly and make it better on a daily basis. I think both of those methods have merit, and the ideal would be to do both of them.

I think that's going to be the interesting thing that happens when you see Western companies trying to move their big game franchises into a free-to-play model worldwide and coming into contact with the Asian companies who are moving their free-to-play games to the West; you get this big clash of production values versus customer experience optimization. That's going to push everybody to improve significantly. That's going to be quite an arms race because it means we need to learn different ways of making our games. We can't come up with this grand vision for GEARS OF WAR, spend three years building it, and then see if customers like it. I'm exaggerating; we actually put a lot of effort into playtesting and getting customer feedback up front, but it's nothing like the scale of what happens in a game maintained by Tencent, for example.

[BRANDON SHEFFIELD] I feel like, over the last five years, many companies have dropped the graphical fidelity and stopped trying to push graphics and have left it to the realm of blockbuster guys. Riot can make League of Legends look good enough, then have [such] a fantastic user experience that it doesn't matter. So I wonder if you consider yourselves guardians of graphics technology for the future, keeping

graphics moving forward because you're trying to push the console makers, to some extent, through the chipsets they may have?

[TIM SWEENEY] Well, Epic's engine programmers and our artists really take it as a matter of pride that we want to have the best-looking stuff available, bar none, on every platform. If we're building a high-end PC game or a next-generation console game, we want to have the best graphics quality possible with however many teraflops are available. If we're building an iOS game, we want that to be the prettiest iOS game. With every generation, the number of things you need to do right to succeed with your game increases. It can't just be a beautiful game; it also has to be a super fun game. It has to have great multiplayer. It has to have great sound and great controls. Now we're adding all of the user experience maximization on top of that. It's just getting more and more challenging to build a game, and we need to respond to that by growing in team size and really staying on top of all that the industry is trying to do. In the future, we can't just give up pushing graphics. That's not and never has been an option.

[BRANDON SHEFFIELD] If Epic were the last company—if Unreal Engine 4 or 5 or whatever were the last high-graphic push in games—would you continue pushing forward graphically if there were no competition?

[TIM SWEENEY] Sure. We always want to outdo ourselves regardless of where the competition is. The main goal is not to increase graphical quality by just throwing more money at the problem, but to do it intelligently by building better schools and technology that make it possible to do that efficiently. We've been very much focused on not competing by brute force all along. **i**

Brandon Sheffield is director of Oakland, California-based Necrosoft Games, and editor emeritus of Game Developer magazine. He has worked on over a dozen titles, and is currently developing two small-team games for PlayStation Mobile. Follow him on Twitter via @necrosofty.



Chania Crete Greece
14 - 17 May



Take a step back from the trenches of development and explore topics that a looming crunch time often precludes. At the conference Foundations of Digital Games conference, for the 8th year in a row leading researchers and developers from around the world will be presenting their latest work.

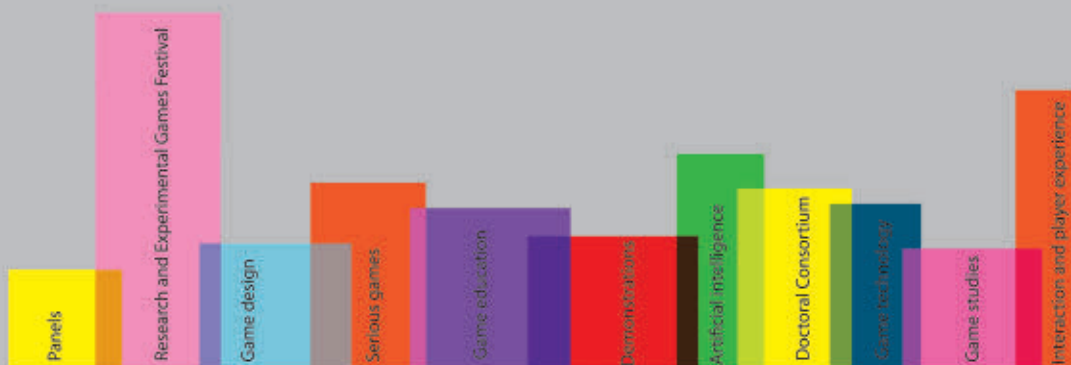
Topics range from:

- adaptive games - procedural content generation - player psychology - game analytics
- player experience - game culture - games for change - game philosophy

Contribute to this forum focused at advancing the study of digital games both as cultural and technological artifact.

Join us in Chania, Greece for FDG 2013!

Tracks



FDG 2013 is a conference of the Society for the Advancement of the Science of Digital Games



BY KEVIN BRUNER

Telltale is a bit different from your typical game company. We bring some of the best stories from TV and film and make episodic "story" games from them. We do our own licensing, design, development, publishing, and marketing, all under the same roof. We also tend to make product decisions together as a studio. We are independent, but bigger than "indie." All of this brings a very unique set of challenges with it, which makes this postmortem a bit different as well. At Telltale, working with external partners and figuring out what "playing a story" means is just as important as slinging DMA packets around and writing code that controls what each CPU core is doing.

We want to make games that let people immerse themselves within an actual role in stories like they never have before. We've had a fair bit of success in attracting some well-known licenses, but our roots are in graphic novels, reaching all the way back to Sam & Max and Bone. We were thrilled at the opportunity to tackle something as rich as The Walking Dead and work with Robert Kirkman. By now, our series has won more than 80 Game of the Year awards and has been widely recognized as a groundbreaking achievement in the industry, but it certainly wasn't an overnight success! >>>

THE WALKING DEAD



WHAT WENT RIGHT

PARTNERING WITH SKYBOUND In early 2011 we reached out to The Walking Dead creator Robert Kirkman and his production company Skybound to discuss adapting his award-winning comics into a Telltale series. We were not going to pitch a game that was about killing zombies. Instead, we were going to pitch an episodic “story game” that focused on the thing we felt made his books so compelling: normal people adapting to impossible situations. The game we pitched was focused on dialogue and character development, and was to be incredibly cinematic and seriously emotional. We wanted people to connect with the characters and role-play like they never had before. There were certainly much more obvious takes on a zombie genre game, which made us a little nervous. Our pitch was probably not what they were expecting. Luckily, they were familiar with some of our previous work, so they took the meeting.

When we met with Robert and the other folks at Skybound, they were very impressed by our vision of what a WALKING DEAD game could be. They understood how such an approach, if successful, would capture a much deeper sense of what made the comics so successful. Most importantly, they saw a special opportunity and were willing to take this risk with us.

In the end, Robert Kirkman and everyone else at Skybound were (and are!) amazing partners. Even as the television show was becoming a runaway hit, they always made time to ensure we had the feedback and support we needed. One of the best things Robert did early on was introduce us to his friend Gary Whitta (screenwriter of Book of Eli and After Earth). Robert trusts Gary, and Gary really understands what makes THE WALKING DEAD unique and special. Consequentially, Gary joined the team relatively early as our story consultant. As a gamer himself (he was editor-in-chief of PC Gamer for many years), his integration was easy and his impact immediate, making valuable contributions to the season story and design. Gary helped push us further toward the darker and more dramatically difficult aspects of THE WALKING DEAD. When considering some of the trickier or more heart-wrenching parts of the story, such as letting the player choose to kill Duck or not, Gary would often be the one evangelizing the “you have to let

the player go there” angle. Eventually, he dove in even deeper and wrote the script for the fourth episode.

LEARNING FROM PAST PROJECTS Many game companies “reset” at the end of a project. They choose to start a new engine, or pursue a new game mechanic or control system, and choose to tackle completely new problems. At Telltale, we intentionally evolve our processes and technology without resetting them. We carefully take what has worked in the past and add layers to it instead of reinventing it. The same tools used to create THE WALKING DEAD can still load and play the entire Telltale catalog.

Telltale has always aspired to make our unique experiences truly compete with more traditional genres such as FPSs and RPGs. Over the years, with each series, we’ve challenged ourselves to make “playing stories” more compelling and push the envelope as to what it means to “play a story.” In addition to the fundamental writing and design tasks, each series has tried to tackle specific aspects of this larger agenda. We’ve had some great successes and a lot of lessons learned along the way.

Very early on with SAM & MAX, we were simply figuring out the formidable challenges involved in designing, producing, and publishing digital adventure games on an episodic schedule. Later, with a series like WALLACE & GROMIT, we challenged ourselves to move past traditional “point and click” and embrace the console controller. STRONG BAD’S COOL GAME FOR ATTRACTIVE PEOPLE was the first regularly scheduled episodic game ever simultaneously released on consoles and PCs. POKER NIGHT AT THE INVENTORY had characters dynamically telling jokes and stories while reacting to the player and the cards. It also introduced our first online features with its unlockable TEAM FORTRESS 2 items. Back to the Future enabled us to examine how to make our games less obtuse and more accessible, reaching wider audiences and more platforms. With JURASSIC PARK, we experimented with alternate control schemes and playable characters, and tried making fast-paced action sequences. Most importantly, we learned ways to keep the pacing of the story moving along while staying interactive and keeping the player engaged. Along the way, each series allowed us to refine our episodic production pipeline and tools, and become a much more significant (and successful!) self-publisher.



We've never been content to be a boutique game developer/publisher. In innumerable ways, THE WALKING DEAD is the result of eight years of continuous exploration and evolution. This process not only informed us while creating THE WALKING DEAD, it also gave us the confidence to take the even more ambitious risks you see in the finished game, such as...

THE "NO GOOD CHOICE" GAME

MECHANIC Going in, we knew that in THE WALKING DEAD you'd be confronted with awful choices, not much time to consider those choices, and then have to deal with the fallout of those choices. We did spend some time exploring integrating choice into more traditional game mechanics, but things didn't start to really gel until we decided to go "all in" on choice. As a studio, we wholly embraced the idea that the game would allow narrative opportunities if they were plausible to the player. However, it wasn't a simulation: We would still be handcrafting the "spine" of the story, and if a narrative option seemed plausible within that context, we committed ourselves to letting the player explore it.

A great example of this is a critical moment in the third episode. One member

of your party, a woman named Lilly, suddenly and deliberately shoots another member of your party, ostensibly to ensure the ultimate safety of everyone. The initial idea for this section of the game was that zombies would then immediately drive the group back into their RV, which would force the player to deal with Lilly's actions in those close, inescapable quarters. As the details of that interaction came into focus, it became clear how strongly we felt about the idea of abandoning her at that moment. This was a moment we simply had to allow the story to play out both ways (she stays or she goes). We felt that empowering the player in critical moments like this was far more satisfying than giving a blanket good or evil ranking.

We also discovered that smaller, more intimate details often worked much better for us than giant, sweeping, branching opportunities. For instance, it's often more interesting to let the player slight or insult NPCs than it is to let the player outright steal all their belongings or completely betray them.

Sometimes there were great initial story ideas, but they included some plausible choices that didn't lead to interesting places. For example, in episode two, we really wanted different characters (not just Mark) to end up as the St. Johns' victim

depending on how you played, but when we explored allowing him to play a more significant role, and even perhaps survive, he just wasn't adding anything good to the story. He became a sort of vestigial story problem in an otherwise really good part of the game. In these cases, we had to punt. Often though, we would keep working until we had moments where all possibilities were compelling and producible. It was most exciting when we created moments where all possibilities felt equally compelling and intriguing. These made the choice taken and the choice not taken just as important to the player.

Other games have certainly explored choice and branching narrative before, though most of those games have used player choice as a subset of a larger gameplay mechanic. They typically had rather binary player choices such as good or evil, light side or dark side, etc. which in turn drove various AI and NPC systems. We felt strongly that a statistics-driven AI and NPC system would not be able to deliver the kind of experience we envisioned. Instead, our system tracks every choice a player makes at a very detailed level and then makes that information available to designers. They use that information to create narrative-driven logic that controls the content and subsequent choices that are offered to the player. Our version of choice isn't emergent from a system, but a carefully crafted bespoke experience driven by narrative possibility. This is what we mean by "tailored narrative."

EPISODE DIRECTORS MAKE EVERYONE A STORYTELLER

Beginning with JURASSIC PARK we introduced the idea of individual episode directors. An episode director is akin to the film director of a movie. They work very closely with the writers, designers, and production departments to craft a vision of how the episode will unfold for the player. The director chooses the combination of narrative tools that are used to create each moment in the finished game. Their job is to leverage lighting,



Caption



color, music, sound effects, composition, pacing, and contrast to make the best interactive experience they can, and then they have to communicate and drive that vision through to the completion of the episode. Having someone dedicated to interpreting the script and design of each episode in this way encourages everyone to approach their work from a narrative context first. It transforms everyone in the studio into a storyteller.

In this model, sound designers aren't simply creating a room ambience anymore; they are contributing to a beat in a story in conjunction with their director. A modeler is creating environments for specific moments instead of generic simulations. Lighting setups change over the course of the episode based on narrative structure instead of the simple ticking of a clock. The artistic process isn't driven by a simulation anymore, and instead serves the narrative context the player will experience.

The episode director isn't a position at Telltale; it's a role that is filled based on each episode's unique circumstance, and a director needs to have passion for the license, the characters, the script, and the interactivity. We have opened this role to anyone in the studio, from any discipline, so long as they have a clear vision for what their episode will be, and the support and confidence of the team to execute it. So far, we've had directors come from writing, design, art, programming, and cinematics. (We haven't yet had any directors from marketing or finance, but the door is open!)

BRINGING PLAYERS TOGETHER There are deliberately no right or wrong choices in *THE WALKING DEAD*. We didn't set out to make a game you could win or lose. We wanted to expose difficult and interesting possibilities to players and reward them whatever they chose, while still leaving the choices not taken as a lingering possibility. The path not chosen should feel as important as the one they did choose. For instance, there is an opportunity to steal food from a stranger's car in the second episode. Some might choose to steal the food in order to survive; others might view that as a loss of one's humanity.

We wanted to enable the player to see where they stood compared to the rest of the world. So, at the end of each episode

of *THE WALKING DEAD*, we presented a statistics screen that shows five critical decisions the player made, and how their choices compared to other players. Exposing this to players transformed a single-player experience into a platform for social commentary, and encouraged players to discuss their motivations and experiences with each other. It also encouraged players to reconsider their choices and try replaying the episode from a different perspective. Although *THE WALKING DEAD* isn't a multiplayer game in the traditional sense, it certainly inspires a lot of conversations between friends and communities.

In addition to the five stats displayed at the end of each episode, the game also tracked dozens of other decisions players made. We were able to use this to help us understand what types of decisions were most interesting to players and reinforce those moments in subsequent episodes. In fact, at the end of episode four, we added an additional statistics screen showing what combination of survivors were joining you for episode five, and how your group compared to everyone else playing.

Creating a compelling narrative "possibility space" for a player is significantly more rewarding when players are able to see the bounds of that space and compare their place in that space with their family and friends.

WHAT WENT WRONG

CASTING IS HARD Casting has always been tricky for us, and as we keep ratcheting up the drama in our games, the bar for our actors keeps rising as well. *THE WALKING DEAD* had particularly tricky casting challenges. Our leading characters were a southern black professor and an eight-year-old girl, in addition to a large and diverse supporting cast. They certainly were not your typical video game fodder.

We had so many auditions that were more caricature than character. Of the actors who were able to walk the line between too generic and overly stereotyped, there were few voices that matched the way we imagined our characters. Of the few actors left that we

WALKING

did like, many of them were unavailable, too far away, or over our budget.

We initially cast the voice of the main character, Lee, from this compromised field, and proceeded to record the first episode. When we got the first group of lines back, we knew we were in trouble. Lee just didn't have the range and personality that the game demanded, and now we had a large production team waiting instead of implementing. We needed a new Lee, and fast.

Luckily, one of our cinematic artists suggested an actor he was a fan of: the eminently talented Dave Fennoy. We tracked him down, and he was able to quickly get us an audition. Once we heard him, we knew he was what we needed. Dave is not only a fantastic voice talent, but an amazing actor who created a Lee that was much more nuanced and dynamic than we had hoped. Dave gave us confidence and allowed us to push the character of Lee even further.

Clementine, the eight-year-old girl who is with you throughout the game, was also a casting challenge. She is not a precocious stereotyped kid, but a smart young lady who is desperately trying to make sense of a world turned upside down. Finding any adult actress to voice a young girl is challenging, but finding one that could portray a complex character like Clementine was especially difficult. We went through another arduous casting process before finding the amazing Melissa Hutchinson.

The Stranger, the ultimate villain at the end of the game, was also particularly difficult to cast. We needed the character to have a kind of quiet insanity. Once again, we cast and fully recorded what turned out to be the wrong actor. Literally the night before the schedule required we start recording, we found Anthony Lam—the creepy, quiet insane voice we were looking for. Fortunately, he was able to record immediately the following day.

Those lines were being fed into the game as fast as they were recorded.

We need to get better at casting. As our games are becoming increasingly more sophisticated and dramatic, they are requiring more diverse and skilled actors. Fortunately, for *THE WALKING DEAD* we were mostly able to find the great actors we required in time, but just barely.

The challenge of defining and communicating the essence of our characters in ways that provide talented actors the tools they need in a game context has proven to be a daunting and difficult task. Our actors rarely get to perform with one another—we usually record their parts individually—which makes it hard to offer them context or insights into their character's motivation. Also, we're finding that directing the performances during the actual recording sessions is itself a specialized skill set that we need to improve.

REMAKING EPISODE TWO While we were focusing intently on episode one, episode two was quietly coming to life in the background on its own.

When we went "all in" on choice, it changed how episode one became fun and engaging. As these things were changing, we were iterating at what seemed like light speed, tweaking interfaces and adding features like "choice notifications," on-screen inventory, panic meters, and more. We were intensely critical of everything, and we were finding fantastic new ways to make choice compelling.

When we finally got episode one to a good place, episode two had already been written, recorded, and was partially into production. When we were able to compare episode one and two, they felt radically different. Episode one had all this really cool new stuff that wasn't being exploited

in episode two, and episode two felt kind of "old Telltale" by comparison. So we decided we just couldn't move forward with the version of episode two we had.

We gathered a large group of writers, designers, producers, and directors offsite at a local hotel conference room and spent two days hammering out a new direction for the episode that exploited and even advanced everything that was feeling so great about episode one. Though some large elements such as locations, principal characters, and story beats remained, nearly all of the dialogue needed to be rewritten and then rerecorded. We were super enthused by the new design, but we were already behind schedule, leading to...

MASSIVE SCHEDULE COMPRESSION

With a monthly episodic game company, the schedule is king and constantly looms over everything we do. Just imagine if you tuned in for the last episode of *The Sopranos* and there was a screen saying "Sorry, we didn't get it done. Come back later!" All the recasting and retooling that happened early in the series took a lot more time than we had anticipated. By the time episode one really came together, our series launch date was upon us, and as they say, the show must go on! Episode one launched with a much better reception than we could have dreamed.

Unfortunately, now we were further behind schedule than we had ever been before. To make matters worse, we had a huge rewrite of episode two to complete before we could restart production on it. While the new episode two script was completed at breakneck speed and recorded as quickly as possible, we considered what could be done to get back on track. We ended up basically putting nearly everyone in the studio onto the project.





We have always been a multi-project studio, so we've never really had everyone working on the same game at the same time, and figuring out how to best divide production responsibilities among such a large group was tricky. We broke the action sections of the game (such as the zombie attacks) out into their own strike teams, while a different group owned the dialogue-driven parts. All in all, the studio quickly turned out some amazing content, but we were still weeks behind on episode two, and our schedule was still getting stretched out as far as it possibly could.

In the end, episode two ended up in customer's hands nine weeks after episode one was released. Though this was an amazingly short time given the circumstances, customers were understandably anxious, and didn't hesitate to let us know it. Our forums and Facebook pages were swarmed with people demanding their episode, and we didn't do an adequate job of communicating what was happening.

By keeping the whole studio focused on THE WALKING DEAD, we were able to make up most of the time. We wrapped the series just a few weeks short of its originally planned completion date.

BACKEND SERVERS AND DLC THE WALKING DEAD was our first game in which each episode was delivered as DLC instead of individual executables. This required a new "in-app" store, new content licensing code, and new download managers. That's a lot of "new," and furthermore, each platform had its own unique DLC quirks and complexities. iOS even required us to implement our own content delivery network! DLC also presented an entirely new QA process from simply staging content on partner networks, installing and uninstalling individual episodes, and controlling the licensing status of content on all our test accounts. We had a lot to learn about DLC.

In addition to DLC, the game was also the first to send telemetry back to the Telltale servers so we could present the choice statistics at the end of the episode. We created a new REST API specifically to handle this data, which required new code both on the web servers and in the game engine itself.

Both of these changes were substantial, and they ended up being a lot of work to bite off with an important title like The Walking Dead, but we felt that the game would suffer significantly without these features.

When we launched, the game was incredibly well received, and our servers were quickly overwhelmed with the volume of hits they were receiving. When the servers stopped responding to the game, it would cause noticeable delays at the Press Start screen and at some save points during the episode. For the first few days, we actually ended up throwing incoming data away and serving a stock reply while we frantically brought more servers online.

On Xbox and PlayStation, the system handled the download of new episodes for us, but on iOS, the application also had to download the content for the user. This required a download manager, which turned out to have many more problems in the real world than we anticipated. Mobile devices tend to have significantly more network connection problems than PCs or consoles, and we just weren't able to simulate the wide array of real-world circumstances we would encounter. Unfortunately this meant that too many iOS players had to suffer with us as we pored through our server logs, download code, and test cases to finally get the download manager as robust as it needed to be.


BUGS, BUGS, BUGS Between the compressed schedule, the new DLC model, and the work required to launch on many platforms, we ended up with a lot of bugs.

The worst of these was related to our save game "rewind" system, which allowed the user to rewind the story an earlier point in the game. Since most of the episodes didn't exist when the system was implemented, we didn't have the test cases required to ferret out the edge cases that occurred in the real world. Unfortunately, that meant that some players lost their save data and had to replay the game from the beginning, which is about the worst thing that can happen to a player in a choice-based episodic game. Another bug caused the "generate random choices" dialogue to appear when in fact no random choices needed to be generated, which just made the players who were anxious about losing their save data even more worried.

In addition, each platform had its individual list of bugs, as well as the usual permutations of hardware-specific PC bugs, which made it very difficult for us to communicate solutions to the player. We found that many angry customers had to navigate a complex maze of information to find their solution. Even the best-intentioned users often ended up advising other users with solutions that worked in their context, but might be disastrous in another context. For example, some users would "tweak" their data to recover a save game, and other users might try a similar strategy, but end up destroying their save data. Obviously the root of the problem is the buggy system itself, as users should never have needed to attempt any of these things in the first place.

Though we've aggressively patched these issues and resolved most of them, you can't really unring a bell. There are still small content bugs left in the game, and players are too often left with an impression that the game is buggy. All we can do at this point is continue to patch, and take what we've learned from this experience to ensure bugs like this don't happen in the future.

THE WALKING DEVS For the last eight years, Telltale has been manically committed to the idea that interacting with characters and stories could be some of the best game experiences ever. We're convinced that stories are not just for cutscenes—stories can be the game themselves. THE WALKING DEAD is our example of the state of the art, but we're confident that we can make playing a story even more compelling. We've also been equally committed to episodic games, smaller games, and cheaper games. And now we've built a studio that has matured to the point of being able to create a Game of the Year while staying true to these goals.

Fortunately for us, it looks like we'll have an opportunity to continue exploring the space. Our game based on DC Comics's Fables is well under way, THE WALKING DEAD has a second season in the works, and many new properties are seeking out Telltale in hope of working with us to make a game. Stay tuned! 

Kevin Bruner is the CTO and co-founder of Telltale Games. He has been passionate about story in games since playing Infocom text adventures as a kid. He has been working in games for nearly 20 years. When not making games, you can find him playing hockey (badly).

Check out Microsoft career opportunities at booth 2116

We're looking for top talent to help us change the face of entertainment on a global stage. You'll have the opportunity to create unexpected, immersive, and truly extraordinary ways to experience games, movies, music, TV, and more across devices. Whether you're a designer, an engineer, or another kind of superstar, with leading resources like Xbox and Kinect you'll have endless possibilities for re-inventing the world of entertainment.

www.microsoftentertainmentjobs.com

#JumpInGDC



OUR [PROGRAMMERS]
ARE INSANELY GOOD

HEROES
WANTED

RAPHAEL PARENT
UBISOFT MONTREAL
LEAD GAMEPLAY PROGRAMMER



VAAS MONTENEGRO
TROPICAL PARADISE
SOCIOPATH



UBISOFT®
JOBS.UBISOFT.COM

{5}

tips for better playtesting

CASUAL DEVELOPER ARKADIUM'S BEST PRACTICES FOR PLAYTESTS

{P}

Playtesting is one of the most important elements in the game development process. When you see someone from outside the studio actually sit down and play your game, you're able to better understand how accessible, usable, and appealing your game is. These are not things you want to evaluate after your game is released to the public. Therefore, data collected from playtesting can be an invaluable tool for mitigating risk and giving your game its best chance at success.

However, there are many ways that playtesting can go wrong, so it's important to establish a process that you're confident in, and to improve it over time. At Arkadium, our playtesting process is a collaboration between game design and marketing, and typically involves inviting players to our office. We believe that the best practices we've identified over time may be helpful to your team as well, so we've boiled them down to these five tips for better playtesting.

{1}

RECRUIT YOUR TARGET PLAYER

When designing your playtest, you should be keenly familiar with the type of player your game is for. While your game may appeal to a larger audience, it's a mistake to "design for all." We find it helpful to identify personas for our target audience—for example, "Jane, 25-34 years old, owns an iPhone"—and try to recruit players who closely match that description.

SOME QUESTIONS WE ALWAYS CONSIDER WHEN RECRUITING ARE:

- }} How old is our intended audience?
- }} Is the intended audience predominantly male, female, or split?
- }} What device or platform is our game targeting? (Don't recruit players if they've never played a game on that platform before.)
- }} Does your game assume any prior knowledge from the player?

The last point is often the most important. If you're developing a sequel to a popular first-person shooter (FPS), you might assume that most of your players have played the first game, or perhaps a similar

FPS. In this case, it would make sense to recruit playtesters who are already familiar with the basics of FPS games, and solicit feedback on the elements that are unique to your game. If you're developing for a casual audience, however, you may actually want the opposite. We often find that it's imperative to test our game with players who have never played a game's predecessors, especially when testing the game's tutorial or a game that's meant for more casual players.

Every time we have a playtest, we strive to have some new playtesters who have never tried out our game before. This guarantees we have a fresh set of eyes providing feedback every time, even if we also invite returning players to playtest something new. We also have a policy against recruiting friends of our employees for playtesting. While their feedback can be very valuable in a pinch, we find that when they know someone personally who has worked on the game, people have a tendency to want to love it more than they would otherwise, which can drastically skew the data.

Of course, it can be difficult to find and recruit playtesters who closely match your target persona. There are likely online communities—some related to your game, for instance—that are full of players who would love a chance to visit your office and provide you with feedback. However, while your biggest fans will be eager to help, they tend to be more informed and more engaged with your games than the average player. If you only collect feedback from your biggest fans, you're much less likely to hear about the problems that new players face on Level 1, and much more likely to hear about how great Level 10 is. We supplement our pool of playtesters



by recruiting on sites like Craigslist, FindFocusGroups.com, and Meetup.com.

{2}

TEST YOUR TEST BEFORE YOU TEST!

Before your testers arrive, set aside some time for everyone on your team involved in the playtest to do a run-through. Testing your playtest will ensure things run as smoothly as possible, by ironing out the kinks ahead of time and getting the whole team on the same page.

Test your players' experience from the moment they enter your door. Visiting playtesters should be greeted by someone when they enter, or at least find a sign-in sheet. Every detail matters to create a comfortable experience. You may have a state-of-the-art usability lab and great refreshments, but it won't matter if the building's front door is locked! Is all of your technology working? Check your speakers, mouse, keyboard, and Internet. You don't want to have to call IT during a playtest because you forgot to install a plug-in that your game requires to run.

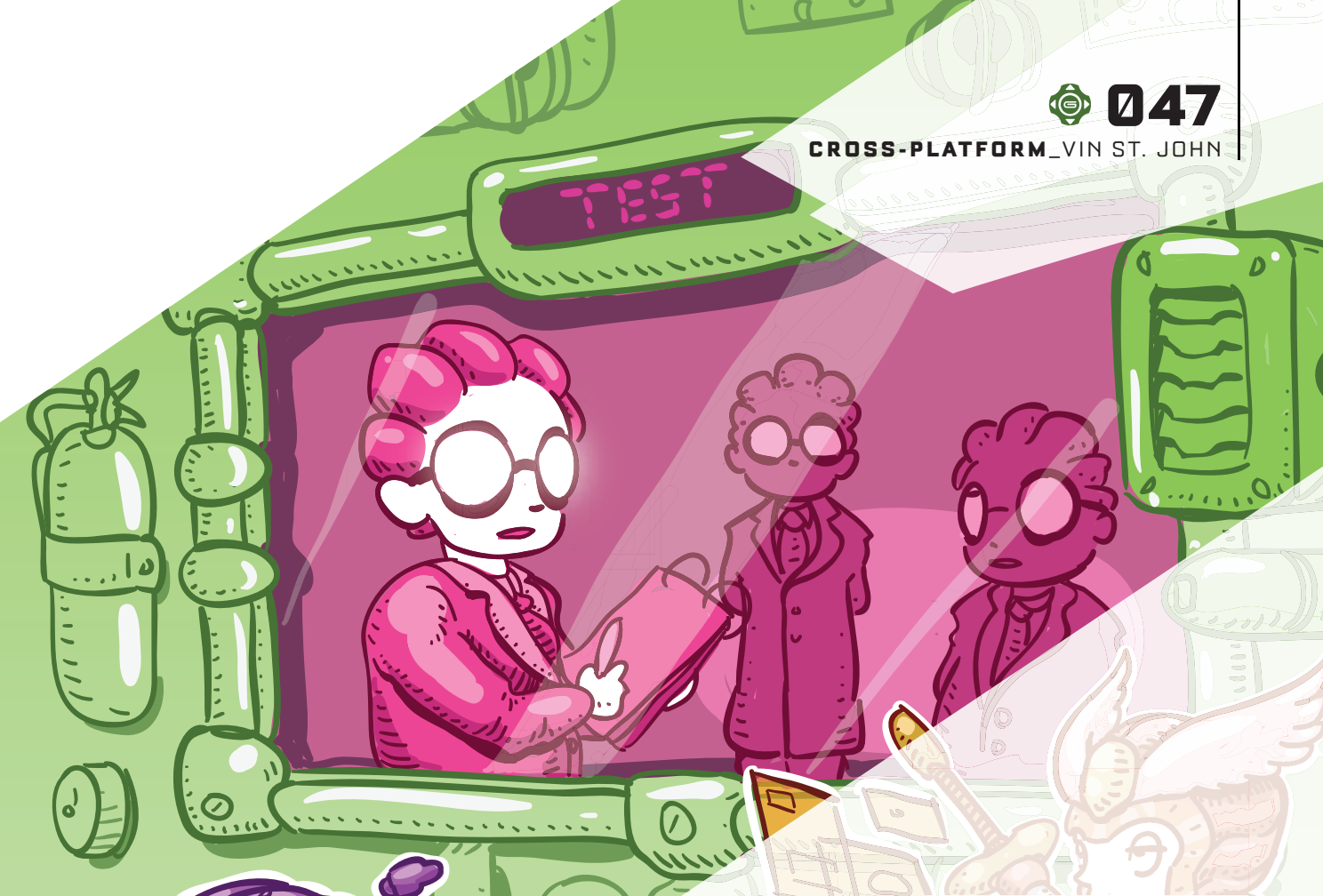
Brief all playtesting helpers on your process. At Arkadium, the whole team can get involved in playtesting. Everybody involved in the process is briefed that day on the goals of this playtest, what we hope to learn, and the process we're using

{3}

TAKE THE PRESSURE OFF The concept of playtesting is still new to some game developers; for the average human being, it is completely foreign. Your playtesters may be



Arkadium Games.



unsure of what to expect, and that uncertainty can have a negative effect on your test results. Make your playtesters comfortable so they can focus on playing your game and you'll get much better feedback.

Be sure to tell your moderators to segregate themselves from the game during the test. We assure players, "We're testing the game, not you," and that we are hoping to find areas where they struggle or get confused, because we want to improve the game. We don't want our players to be afraid of giving honest (negative) feedback, so we always stress that we are observers, not designers, and are here to listen to any and all criticisms.

Our observers sit back and let the playtester play the game with very little interference. However, we find it helpful to encourage the playtester to think out loud during their experience, to help us understand their point of view.

Recording the test for the rest of the game's team to watch later is also helpful. Because most people don't play video games in front of a note taker and a video camera, we find it's important to put a player's mind at ease about these elements and explain to them why they are helpful. Players usually enter our office not knowing what to expect, but leave understanding what a playtest is and why we conduct them.

{4}

ROCK THE SURVEY You'll almost always want to ask your players some questions

after they're done playing. Try to predict these questions ahead of time and put them in survey form so that you collect the same set of data for every playtester. If you have any spontaneous follow-up questions, you can ask them when they're done taking the survey.

A carefully worded, highly focused survey can make sharing your playtest results with the team much easier. Focus on questions that directly address the goals of your playtest. Leading questions like "Was the tutorial confusing?" are much less helpful than questions that test a player's knowledge, like "Please describe what the green button does in this game."

If your playtesters are newcomers to your game or genre, they are probably unfamiliar with many of the terms and conventions that your team may take for granted—plan accordingly by using plain, descriptive terms in your questions whenever possible. Our surveys always include some basic player profile questions like "What games have you been playing in the last month?" to give us context. In addition, every multiple-choice question includes an optional space for written explanation.

{5}

ANALYZE IN AGGREGATE After a playtest, you're going to have survey results and notes from many different playtesters and observers. Compile that data quickly and share it with your team in aggregate, without offering analysis or

drawing conclusions. Analyze the results as a group, and start on the aggregate data—your game designer may have seen one player who thought the game was too easy, but is that what everyone else saw? If all the other players said the game was difficult, then you know that "too easy" is not a trend. If you like you can still return to that playtester's feedback afterward and address it as a special case. Analyzing your data in aggregate first will guarantee that your entire team benefits from the full playtest.

Once you've got your data compiled, don't just forget about it! Identify the issues that your playtest has brought to light and prioritize your next steps. The data you collect during your playtest—which can include anything from the player's emotional state, to the number of failed attempts to click a button—should always help you draw conclusions and come away with action items for your team. If your data is inconclusive, consider revisiting the structure of your playtest and survey for next time.

These are the best practices we've developed at Arkadium's New York headquarters, and they have helped us implement recurring playtesting in a consistent and reliable way. Of course, we're always improving our process, and what works well for us might not suit the needs of your studio. If you find any of these tips helpful, or have a different way of doing things, let us know. 🎮

Vin St. John is the marketing manager at Arkadium.

The next generation of Havok Physics
is here...



Check out Havok's new tech at GDC

Faster run-time performance. Reduced memory footprint. Built for next-gen platforms.

Schedule your meeting now: www.havok.com/physics

Havok's leading middleware technologies include:

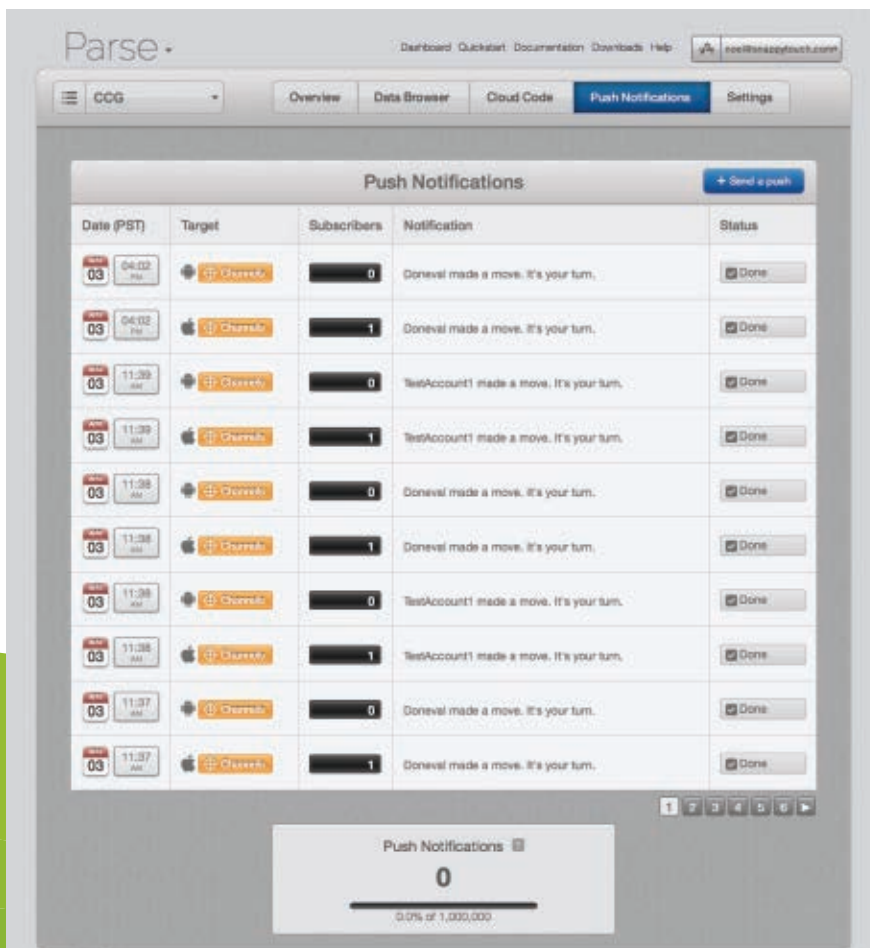
Havok Vision Engine • Havok AI • Havok Animation • Havok Behavior • Havok Cloth • Havok Destruction • Havok Physics • Havok Script



PARSE

Like many game developers out there, I went for many years without ever touching the dreaded server. I wrote shaders, parallelized code on SPUs, and optimized cache misses, but I was lost as soon as the conversation turned to relational databases. But as the times changed and I started making games on iOS, I was eventually forced to dive into the server backend waters. I'm still not an expert, but I learned enough along the way to write simple server backends for my games with MySQL and PHP.

Unfortunately, there's a big difference between writing simple server backends and creating a robust, scalable backend with advanced features that could support a top-selling iOS app. That's where Parse comes in: It allows developers to create a scalable server backend with very little server experience. Parse is so simple, I could have used it even before I knew what a SELECT command did.



INTRO TO PARSE Parse is a scalable server backend intended for mobile platforms, although it also supports desktop and web platforms. In order to interact with the server, Parse exposes a REST (Representational State Transfer) API, which means you can do everything just by doing HTTP requests. But since sending raw HTTP requests is cumbersome and ugly, they provide native API wrappers for a lot of platforms (and different languages). That means that you can write client code by simply calling functions in your native language, and those functions will get translated to HTTP requests and sent to the server, which parses the response and returns the data to your code. Parse also provides a top-notch web-based dashboard to manage all your apps, change settings, view and edit data, users, and so on. I can't stress enough how well done this dashboard is, and how easily Parse lets you manage and debug your applications.

The Parse backend offers four major feature categories: Data, Social, Push, and Cloud Code.

Parse Data is the core of Parse; it lets you store data in a big database, access it, and modify it from any device. You can do the kind of queries you've come to expect from a relational database, and you can even browse and query it through the web dashboard. The client API takes care of presenting the fields in each table as a dictionary, which makes working with the database much

simpler than through SQL statements. On iOS, all you would have to do to update (or add) a new record to the GameState table is:

```
PFObject* gameState = [PFObject objectWithClassName:@"GameState"];
[gameState setObject:[NSNumber numberWithInt:score] forKey:scoreKey];
[gameState setObject:player.name forKey:playerNameKey];
[gameState setObject:[NSNumber numberWithInt:player.actionPoints] forKey:apKey];
[gameState save];
```

Their API documentation is very comprehensive and well done. Between their samples and reference, it takes no time to figure out how to use all the different functions.

Parse Social is mostly a fancy name for user management; you can create users, verify their login, and manage their login info (send confirmation emails, links for forgotten passwords, and all that boring bookkeeping that you would have to implement otherwise). Anyone who's had to implement that from scratch will appreciate having it ready out of the box. You can also let your users log in using their Facebook or Twitter information.

Logging in an existing user is as simple as:

```
[PFUser loginWithUsernameInBackground:m_username.text password:m_pw.text target:self selector:@selector(handleUserLogin:error:)];
```

Parse Push is used for delivering push notifications—handy for games that use a turn-based multiplayer mode, an in-game messaging or item-trading system, or anything else for which players would expect to receive a notification. Writing the

notification server code yourself isn't too difficult, but it's not trivial either. You can start with something like EasyAPNS, which is open source, but you would still have to hook it to your game code, and it only works for Apple push notifications. So having Parse Push from the start would save a lot of time, and because it's not a platform-specific solution, your notifications can reach users in different platforms.

Cloud Code is a slightly different feature from the others; it lets you actually run code on Parse's own servers. This is crucial for a lot of games where simply storing and restoring data isn't enough. Most games

will at least want to do some kind of validation to make sure the state they're receiving from the client is valid. The main downside of Cloud Code is that, for now, it only runs JavaScript code, which is not everybody's language of choice.

PLAYING WITH PARSE I've only used Parse for a couple of prototypes for turn-based multiplayer games with push notifications, so I can't comment firsthand on how well it scales,

but an impressive number of apps and games use their service, so it's good to know that there are full products out there now using it.

From a development point of view, implementing Parse was as easy as they make it sound. In a few hours I had user creation, login, messaging, and push notifications working. That's hard to beat, especially when the alternatives to Parse would be rolling your own from scratch (which, if you want to make it really scalable, requires some serious server tech know-how), or using Amazon Web Services or Google Apps. If you use the latter you would get scalability, but you would still have to write a lot of the server code.


PARSING PRICING Parse has three tiers: Basic, Pro, and Enterprise. Basic is completely free as long as you don't go over some limits, so it's great for development and prototyping; Pro costs \$199/month and has somewhat higher limits; and Enterprise prices aren't quoted, so you'll need to contact them to find something that fits your needs.

This is the main (and unfortunately huge) downside of Parse: Pricing seems reasonable for paid, small, "boutique" games and apps, but costs can get completely out of hand very quickly for free games with lots of users. A lot of mobile games these days are free with in-app purchases, so they rely on having lots of users (and a relatively small amount of revenue per user). These are the games that need highly scalable backends, but this is where Parse's pricing becomes prohibitive.

For example, the free tier is limited to 1 million API requests per month. It may sound like a lot initially, but it's extremely limited. That's only about 33K API requests per day. If each user does 50 API requests (and when counting every login and action in the game, that's a pretty conservative estimate), that means you can only have about 600 daily active users. Beyond the 1 million API requests, you're going to be paying 7 cents per API request (yes, you read that right). So each user beyond the 600 free ones is going to cost you \$3.50 per day in API requests alone!

The Pro tier isn't much better: 15 million API requests per month is about 10K users, which is very low for most moderately successful free games. Beyond that limit, you'll be paying 5 cents per API request, which is extremely expensive. Push notifications are equally limited and can also rack up huge bills, in addition to the charges for API calls. All of that is on top of the \$199/month you're paying for the Pro tier.

Another drawback of Parse is that prices can change at any time, like they did last year, taking it from a very reasonable price even for high-volume applications to its current rates. I would hate to build my successful game around it, just to find out a couple of months after launch that rates have doubled. At that point you're also tied to their API and data model, so migrating to a different service could be quite costly. I would feel better about signing up with them if they guaranteed to maintain or reduce your rates after you sign up.

NOT FOR PARSIMONIOUS DEVS Parse is absolutely fantastic for prototyping, because you get so much from the start and you can safely use their Basic tier for free. It will also help you to structure your code in a way that works well with server requests. Later in development, you can decide whether to leave it in, or replace it with your own custom server code. Paid games with a very small number of users can also benefit from their Basic and Pro tiers. If your game's player base is in the medium-to-large range, however, you're probably better off looking for another provider or doing it yourself. 

Noel Llopis is a game industry veteran turned indie-game developer. He avoids violence in his games and instead relies on creativity and sharing. His latest games include CASEY'S CONTRACTIONS and FLOWER GARDEN.

Parse
<https://www.parse.com>

CLIENT APIS

iOS and OS X (ObjC), Android (Java), Windows Phone 8 and Windows 8 (.Net), HTML5 (JavaScript)

PRICE

Basic tier is free; Pro is \$199/month, Enterprise, call for pricing

PROS

- 1) Easy to implement
- 2) Prototyping with Basic account is free
- 3) Great web dashboard

CONS

- 1) Projects can quickly outgrow Basic and Pro tiers
- 2) Cloud Code limited to JavaScript
- 3) Pricing rates subject to change



HAUPPAUGE HD PVR 2 GAMING EDITION

Capturing and streaming video is a hot topic among many devs these days, and with good reason. Whether you're a scrappy indie responsible for putting together your own QA setup (and cutting your own trailers for PR purposes), or you just want to find out how to build your game's online presence with Let's Play videos on YouTube and channels on live-streaming sites like Twitch.TV, it's worth doing a little DIY research in the field for yourself. In order to do that, you'll need a video capture kit.

SETTING UP THE PVR 2 Until relatively recently, if you wanted to capture video from a game console onto your PC, you'd need to install an expansion card (the Blackmagic Intensity Pro, for example) into one of your PC's PCI-E slots and connect the console to that card's video input ports—which meant that laptop owners were out of luck.

With a capture box like the Hauppauge PVR 2, however, you can simply plug your console into the PVR 2's HDMI-in port, plug the PVR 2's HDMI-out port to your TV, and plug your PC into the PVR 2's USB 2.0 port, meaning that you can pass the video signal through to the TV and play on the TV while outputting video to your PC to record it—no PCI-E slots or desktop PCs necessary.

RECORDING AND EDITING The recording process itself is pretty easy: Just install the recording software, tweak your recording parameters to your liking, and click Record in the PC software or press the handy button on the PVR 2 box itself.

The PVR 2 can record video from sources with HDMI, component, composite, or S-video outputs, though the HDMI features don't work on devices with HDCP DRM, so you can't capture PS3 video through HDMI. (A PS3 component video cable is included.) You can choose from a variety of standard broadcast resolutions (both interlaced and progressive-scan) and framerates, though 1080p captures will only be recorded at 25 or 30 frames per second even if the source is 50 and 60 frames per second, respectively.



game developer magazine



Data box:

Hauppauge HD PVR 2 Gaming Edition

Hauppauge Computer Works, Inc.
www.hauppauge.com/site/products/data_hdprv2-gaming.html

PRICE

\$170

SYSTEM REQUIREMENTS

2.0GHz multicore CPU, Microsoft Windows (XP SP3, Vista, 7, or 8), display with HDMI input, 512MB RAM, 220MB free hard disk space, optical drive (for software installation)

PROS

- 1] Easy to configure, record, and edit
- 2] Supports HDMI passthrough
- 3] Package includes software and necessary cables

CONS


- 1] HDMI passthrough requires connected PC to be on
- 2] Included software is very basic
- 3] PS3's built-in DRM prevents user from recording via HDMI



One of the PVR 2 Gaming Edition's key features is its "no-delay HDMI passthrough," which allows you to play your game on a TV while sending video to your PC without using a separate HDMI splitter. According to the Hauppauge tech team, the "no-delay" passthrough adds about 4 milliseconds to the video display time, which is fairly negligible but worth noting for acutely lag-sensitive uses. Also, if you leave your console plugged into the PVR 2 and your PC isn't on, you'll just get a black screen, which is annoying.

You can edit your footage with the included ArcSoft ShowBiz video editing software. ShowBiz is limited, to say the least; you can add transitions and string clips together, but if you have anything else, like iMovie or Adobe Premiere Elements, you're better off using that. Note that the recording software will save your recordings either as AVCHD (.TS and .M2TS) or MP4, so make sure your intended video editor can handle the formats you're trying to edit before you start capturing.

Also included in the software set is a streaming video app called StreamEez, which can output to channels on Twitch.TV or Ustream. If you're looking for a no-nonsense way to start streaming, it's not bad, but compared to dedicated third-party apps like XSplit and FFSplit, which allow you to juggle multiple layouts and A/V inputs and use custom overlays, StreamEez is rather lacking. Unfortunately, the PVR 2 doesn't support any streaming apps besides StreamEez as of this writing.

GOTTA CAPTURE 'EM ALL All in all, the PVR 2 is an easy way to start capturing your own game footage, though it's not without its shortcomings. Compared to its closest competitor, the Roxio Game Capture HD Pro [\$149], the PVR 2 is a bit more expensive and its software isn't as good, but its ability to downscale video on the fly (so you can play at 1080p but record at a lower resolution) is pretty handy. 

Patrick Miller is the editor of Game Developer magazine. Follow him on Twitter via @pattheflip.



ACADEMY of ART UNIVERSITY®

FOUNDED IN SAN FRANCISCO 1929 BY ARTISTS FOR ARTISTS



Gil Weinstock



Wala Alhadad

STUDY ONLINE OR IN SAN FRANCISCO

Acting*
Advertising
Animation & Visual Effects
Architecture
Art Education
Art History
Fashion
Fine Art
Game Design
Graphic Design
Illustration
Industrial Design
Interior Architecture & Design
Jewelry & Metal Arts
Landscape Architecture
Motion Pictures & Television
Multimedia Communications
Music Production & Sound
Design for Visual Media
Photography
Visual Development
Web Design & New Media

ENROLL NOW

EARN

YOUR AA, BA, BFA, MA, MFA OR
M.ARCH ACCREDITED DEGREE

ENGAGE

IN CONTINUING ART EDUCATION COURSES

EXPLORE

PRE-COLLEGE ART EXPERIENCE PROGRAMS

WWW.ACADEMYART.EDU

800.544.2787 (U.S. Only) or 415.274.2200

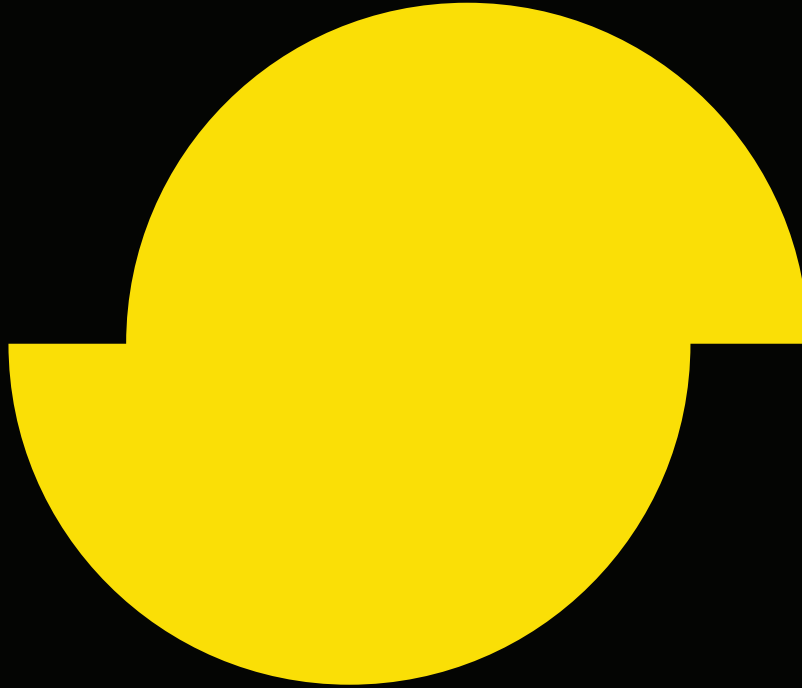
79 NEW MONTGOMERY ST, SAN FRANCISCO, CA 94105

Accredited member WASC, NASAD, CIDA (BFA-IAD, MFA-IAD), NAAB (M.ARCH)

Acting degree program is not available online.

Visit www.academyart.edu to learn about total costs, median student loan debt, potential occupations and other information.

Automagic 3D Optimization



MeshLOD. ProxyLOD. BoneLOD. MaterialLOD.

Simplygon is the leading tool-chain middleware for automatic optimization of 3D-game content and Level of Detail. By replacing tedious and time-consuming manual work, Simplygon offers the benefits of LODs but without the drawbacks of increased production time and development cost.

www.simplygon.com

SIMPLYGON™

Simplygon is used by industry-leading developers including AniPark, Avalanche Studios, Cryptic, CCP Games, Doobic Game Studios, Epic Games, Funcom, Giant Interactive, IMC Games, Nexon, Pearl Abyss, Piranha Games, Quantic Dream, RealU, RedGate Games, Reloaded Studios, Joymax, Neowiz, XLGames, Wemade Entertainment.



STUDYING SPEEDRUNNERS

DEBUG DOOM BY WATCHING ITS TOP PLAYERS AT WORK

Whether you've just written your first "Hello, World!" or you've been programming since the punch-card days, you know that we all make mistakes as a necessary part of the learning process. Sometimes we catch those mistakes in time, and other times those mistakes end up lingering in the final shipped product, becoming part of the game whether we like it or not. In this article, we're going to dissect a few bugs found in DOOM, id Software's seminal first-person shooter—bugs that would be very easy to miss if it weren't for the devoted speedrunners who exploited them to move significantly faster than its developers intended.

SPEEDRUNNING DOOM For the uninitiated: Speedrunners are players who specialize in playing through games as quickly as possible. Naturally, they're concerned with looking for any strategy that allows you to move faster, whether they're engine-wide exploits or specific paths through certain levels. We're going to focus on a bug in DOOM that is a speedrunner's dream: a simple glitch in the movement code that allows us to move much faster than intended.

This has broad implications. Moving faster means jumping higher and farther (you can't jump in DOOM, but a faster speed still lets you cross wider gaps than normal—see **Figure 1**). Some levels intend you to go out of your way to find a switch to raise a bridge, for instance. With a trick that lets you move faster, you might be able to



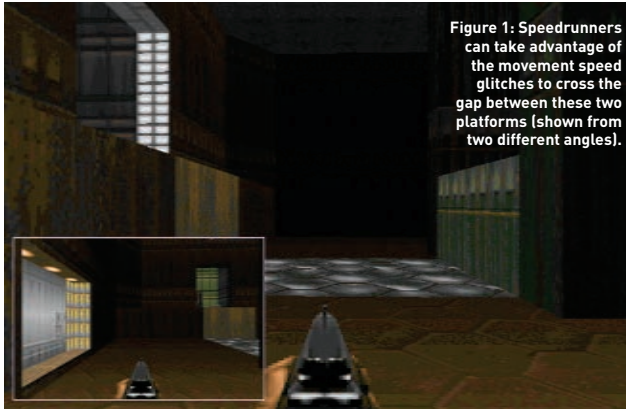


Figure 1: Speedrunners can take advantage of the movement speed glitches to cross the gap between these two platforms (shown from two different angles).

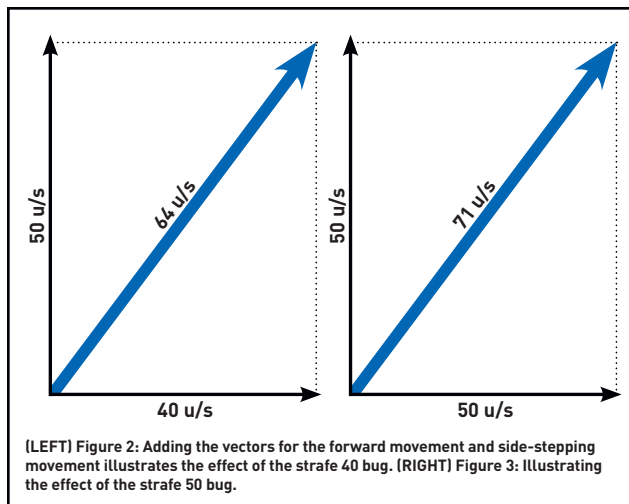
cross the gap without the platform, which means skipping much of the level.

We will look at three movement bugs in particular:

- Strafe 40: Moving forward while also strafing in either direction will allow you to move 28% faster.
- Strafe 50: Moving forward while also strafing in either direction and also toggling a feature that interprets turning as strafing allows you to move 41% faster.
- Wallrunning: Moving along a wall of a particular orientation will move you at almost twice the normal speed.

Go watch a few DOOM speedruns (you can find a few at the Speed Demos Archive: <http://speeddemosarchive.com/Doom.html>). Pay careful attention to the direction the player tends to face. Most of the time, the player is not running in the same direction they are facing, but at a slight diagonal. Essentially, they are moving both forward and sideways at the same time by pressing two movement keys at once—in a modern first-person shooter, this would be like pressing W and A simultaneously. In DOOM, when you use both movement keys while running, the game erroneously moves your character at 128% the normal speed. This is called the strafe 40 glitch.

KNEE-DEEP IN THE CODE Let's now investigate why these particular bugs happen. DOOM is written in C, which is a very low-level programming language, meaning it is designed for speed rather than programmer comfort. This was a very common language for games in the '90s, since the language was designed



to allow you to write very optimized code and push the limits of the technology of the time. With that in mind, I have pulled out relevant pieces of the actual source (<https://github.com/id-Software/Doom>) with very little changes to make the code more clear.

Turning our attention to the DOOM source: There are several functions that deal with moving the player. The gist of them is that for each frame, the game decides how to move the player before it draws the screen. The game looks at which keys are pressed, and if the "walk forward" key is down, it will update the player's position to move them forward. It may then look at handling collisions and projectiles, but we're going to focus on the movement part.

In order to trigger the strafe 40 bug, all you need to do is move forward and strafe at the same time. We can find out why this happens in **Listing 1**—refer to the `P_MovePlayer` function. The `cmd` element holds the distances to move per frame. `cmd>forwardmove` and `cmd>sidemove` contain the distances to travel, either ahead of the player or to the side of the player, respectively. It will set an on-ground value to "true" if the player's z position (their distance from the ground) matches that of the floor the player is currently over. Therefore, it only wants to move if the player is in contact with the ground.

Given that the player is on the ground, the code checks to see if the player is due to move forward (`cmd>forwardmove` will be non-zero) and then calls another piece of code that simply repositions the player to reflect that movement. It does the same thing for a strafe, except in a different direction.

From here, we can see the mistake. We can move forward or strafe independently, and it would work as expected. However, if we move forward and strafe, the player will thrust forward, and then afterward, thrust sideways. From the perspective of the game, these two movements are done at the same time, because both are performed before the screen is drawn and the enemies react. Therefore, the actual speed is given by the sum of the vectors; that is, the length of the hypotenuse as illustrated in **Figure 2**.

Of course, there are many ways to repair this bug and handle movement more correctly. One way would be to determine the angle of the movement and always use the same distance instead of positioning the player twice. Instead of moving in the player's direction and then moving again in another direction for the strafe, simply calculate the movement angle (around 50 degrees for walking and strafing), and `P_Thrust` only once in that direction.

Notice that the code does not account for which direction you are strafing. This is because of a naive optimization: The distance (in the code, this is the `cmd>forwardmove * 2048`) you give to `P_Thrust` can be negative to move in the opposite direction. For the fix, you will have to account for the direction in which you are strafing to get the correct angle, but now you always give a positive distance.

FIXING STRAFE 50 To understand how to exploit the strafe 50 bug, we have to look at how it decides `cmd>forwardmove` and `cmd>sidemove`. These values determine how many units the player will travel per frame in each of those two directions. The flaw is that you can artificially affect these values by having the game accidentally count two different keys as movement during a single frame.

Basically, you tell it to move you to the right... twice, and it diligently listens to you. For this, let's look at the input handling code and the function `G_BuildTiccmd` in **Listing 2**.

We can see the familiar `cmd>forwardmove` and `cmd>sidemove` at the bottom; this chunk is the code that determines those, and we're going to explore how it translates the player's keypresses into in-game values.

In DOOM, you can strafe one of two ways: Either you use a modifier key that causes your "turn left" and "turn right" keys to change function to "strafe left" and "strafe right" while it's depressed, or you assign dedicated strafe key for each direction, which works the same way as the A or D keys on most modern games.

With that in mind, look at the code. By the "**Section 1**" comment in **Listing 2**, we see that the game looks to see if that strafe toggle is held. Depending on whether the toggle is held or not, DOOM either stores the `cmd>angleturn`, which tells the

LISTING 1 Dissecting the strafe 40 bug in the player movement code

```
void P_MovePlayer (player_t* player) {
    ticcmd_t* cmd;
    cmd = &player->cmd;

    // Turn the player
    player->mo->angle += (cmd->angleturn<<16);

    // Do not let the player control movement
    // if not onground.
    onground = (player->mo->z <= player->mo->floorz);

    // Move the player forward, if allowed
    if (cmd->forwardmove && onground)
        P_Thrust (player, player->mo->angle, cmd->forwardmove*2048);

    // Move the player sideways, if allowed
    if (cmd->sidemove && onground)
        P_Thrust (player, player->mo->angle-ANG90, cmd->sidemove*2048);
}
```

LISTING 2 The Doom code responsible for the strafe 50 bug

```
void G_BuildTiccmd (ticcmd_t* cmd) {
    boolean strafe;
    int speed;
    int forward;
    int side;

    // We are strafing if a strafe key is pressed
    strafe = gamekeydown[key_strafe];

    // Is the run key pressed?
    speed = gamekeydown[key_speed];

    // The distances we are moving are initially zero
    forward = side = 0;

    // Determine distances to move
    if (strafe) {
        // (Section 1)
        // If the strafe toggle is on, interpret moving left and right
        // as strafing left and right.
        if (gamekeydown[key_right]) // Strafe right
            side += sidemove[ speed ];
        if (gamekeydown[key_left]) // Strafe left
            side -= sidemove[ speed ];
    }
    else {
        if (gamekeydown[key_right]) // Move right
            cmd->angleturn -= angleturn[ speed ];
        if (gamekeydown[key_left]) // Move left
            cmd->angleturn += angleturn[ speed ];
    }

    if (gamekeydown[key_up]) // Move forward
        forward += forwardmove[ speed ];
    if (gamekeydown[key_down]) // Move backward
        forward -= forwardmove[ speed ];

    // (Section 2) Strafe right
    if (gamekeydown[key_straferight])
        side += sidemove[ speed ];

    // Strafe left
    if (gamekeydown[key_strafeleft])
        side -= sidemove[ speed ];

    // (Section 3) Cap speed
    if (side > forwardmove[ speed ])
        side = forwardmove[ speed ];
    else if (side < -forwardmove[ speed ])
        side = -forwardmove[ speed ];

    cmd->forwardmove += forward;
    cmd->sidemove += side;
}
```

P_MovePlayer function above to turn the given degrees before drawing, or it completely ignores the turning and instead strafes by adding a distance to move (affected by whether or not run is enabled) to the variable `side`, which is initially zero.

So: We know that when we have the strafe toggle on and we press the right arrow key, it will handle that as a strafe to the right and add some distance to the variable `side`. Now take note of the “**Section 2**” comment in **Listing 2**. This code happens independently of the strafe toggle. If you also press the strafe right key, this code will add even more distance to the current value (Note: `side+=sidemove[speed]` is the same as writing `side=side+sidemove[speed]` in C). That means if we press the dedicated strafe right key and we also press right while the strafe toggle is on, then we will effectively strafe twice!

Interestingly, the programmer doesn’t seem very optimistic about the code; if you look at the “**Section 3**” comment in **Listing 2**, you’ll notice that the speed of side movement is capped to the maximum speed you can run forward. However, since strafing was intended to be slower than running forward at only 40 units per second, this code incorrectly caps the sideways strafe speed to 50 units per second! **Figure 3** shows how our movement is now calculated. Since this does not interfere with the strafe 40 bug we investigated earlier, we just found a way to make it more effective.

Even though this bug seems more severe and tricky, it’s far easier to solve than strafe 40—all you need to do is put the code in **Section 2** into the else block after **Section 1**, such that the normal strafe is only considered if strafe toggle is off. Alternately, if you want to allow for strafe to be pressed by the dedicated strafe key even if strafe toggle is on, just fix the code in **Section 3** by capping the player’s sideways movement speed to the proper maximum strafe movement speed (40 units per second). You will now not be able to do any better than the original strafe 40.

FASTER THAN ROCKETS:

WALLRUNNING If you thought the strafe 40 and strafe 50 bugs were handy for speedrunners, this next one is even bigger: When you encounter a wall of a particular orientation, running against this wall propels you to nearly twice the already-quick strafe 50 speed.

In the last two sections, we have looked at the code that handles input and determining the player position. There is nothing left to discover in this code that would yield this bug, so we’ll need to look elsewhere. Specifically, we must shift our focus away from how a player moves, but what stops them from moving.

In the real world, we know that an object in motion stays in motion until it

Scotland. Famous for golf and innovative games development.

All the best players
come here.



We've got quite a reputation for invention, innovation and discovery. And it stretches way beyond Highland Games, the bicycle and golf. We were the first to award a degree in Computer Games Technology and our pioneering games work ranges from the creation of Grand Theft Auto to Bloons and Quarrel. The fact is, Scotland is one of Europe's top games development locations. We have a growing hive of creative and talented games developers and our universities are

developing new and converging technologies across a range of platforms.

Above all, our people are dedicated, committed and passionate for success. And this passion, combined with our world-class academic institutions, outstanding research and superb facilities make Scotland financially irresistible. We can develop your products and help shape your business. And that's what makes Scotland such a popular place to live, work and play.

To see what we can do for your business, visit www.sdi.co.uk/games

SCOTLAND. SUCCESS LIKES IT HERE.



interacts with another body. In the virtual world, we cannot always exactly recreate this phenomenon, but we can simulate this idea by making use of very simple collision-detection algorithms.

We start with a moving object with some sort of trajectory. In the movement code above, we determined the new position of the player, so the trajectory is simply the line drawn from the old position to this new one. With this trajectory, all we must do is check if this line intersects an object or a wall, which just requires some basic algebraic geometry.

So what happens when the moving object hits a wall? It may seem reasonable to say that the wall impedes your movement, and so you end at the intersection of your trajectory and the wall. You stop dead, so to speak. However, an action game developer wants the player to always feel like they are moving, and a dead stop wouldn't produce that effect, so the game needs to simulate some momentum. But how do we want to model this? What happens when you hit a wall at an angle?

In DOOM, if you run into a wall at an angle, you don't stop so much as slide along it. The wall cannot contain all of the energy, and so you keep a bit of your velocity and move along the wall only coming to a stop as the result of friction. As you may be able to gather, there are a lot of factors that determine how far you slide and in which direction, while avoiding the proper, yet computationally expensive physics calculation proves to be a tricky problem.

Let us now review the code in **Listing 3**. The reasoning behind this bug has been, within the speedrunning scene at least, a bit of a mystery, so let's figure out why this happens.

To put the prior code in perspective, DOOM goes through two stages before it renders. The first stage is the one we have reviewed for the strafing bugs. In this stage, DOOM determines the new positions of all objects, including the player. The next stage will then review all of these trajectories and determine final placements of objects due to collisions. This is the stage we are concerned with now.

Check out the **P_XYMovement** function in **Listing 3**. This piece of code handles the update of the player's x and y coordinates, which correspond to the player's position with respect to the floor. The remaining z coordinate, handled elsewhere, is the player's height. The function computes the momentum of the player and uses this to see if the player will collide with walls or objects using the **P_TryMove** function. As you can see, if the player cannot move to the given position, it will call **P_SlideMove**, which will move the player to a position along the wall given their current momentum.

Can you spot the mistake? Hint: It has to do with the purpose of the loop itself. It seems that the programmer was very pessimistic about the collision detection code, and decided that when the player is moving very quickly, the collision detection function **P_TryMove** should be called twice: Once at half the distance, and then again for the rest. The idea is that this improves correctness since you will be less likely to skip over smaller triggers. (Ironically, this attempt to avoid one bug caused another one.)

This is a problem because the "xmove" and "ymove" variables are divided in half to ensure the loop runs twice, however, the **P_SlideMove** function receives the untouched original momentum structure in "mo." Therefore, when running full speed (strafe 40)



Figure 4: Wallrunning lets you survive the run through this moat, which is much faster than beating the level in the intended manner.



LISTING 3 Analyzing the Doom code responsible for determining object placement

```
void P_XYMovement (mobj_t* mo) {
    int xmove;
    int ymove;

    // Cap movement in all directions
    if (mo->momx > MAXMOVE)
        mo->momx = MAXMOVE;
    else if (mo->momx < -MAXMOVE)
        mo->momx = -MAXMOVE;

    if (mo->momy > MAXMOVE)
        mo->momy = MAXMOVE;
    else if (mo->momy < -MAXMOVE)
        mo->momy = -MAXMOVE;

    xmove = mo->momx;
    ymove = mo->momy;

    do {
        // Divide fast movements into two steps
        if (xmove > MAXMOVE/2 || ymove > MAXMOVE/2) {
            ptrix = mo->x + xmove/2;
            ptriy = mo->y + ymove/2;
            xmove /= 2;
            ymove /= 2;
        }
        else {
            ptrix = mo->x + xmove;
            ptriy = mo->y + ymove;
            xmove = ymove = 0;
        }

        if (!P_TryMove (mo, ptrix, ptriy)) {
            // We collided with a wall, slide against it
            P_SlideMove (mo);
        }
    } while (xmove > 0 || ymove > 0);
}

else {
    if (gamekeydown[key_right]) // Move right
        cmd->angleturn -= angleturn[tspeed];
    if (gamekeydown[key_left]) // Move left
        cmd->angleturn += angleturn[tspeed];
}

if (gamekeydown[key_up]) // Move forward
    forward += forwardmove[speed];
if (gamekeydown[key_down]) // Move backward
    forward -= forwardmove[speed];

// (Section 2) Strafe right
if (gamekeydown[key_straferight])
    side += sidemove[speed];

// Strafe left
if (gamekeydown[key_strafeleft])
    side -= sidemove[speed];

// (Section 3) Cap speed
if (side > forwardmove[speed])
    side = forwardmove[speed];
else if (side < -forwardmove[speed])
    side = -forwardmove[speed];


cmd->forwardmove += forward;
cmd->sidemove += side;
}
```

against a wall where both collision checks will fail, `P_SlideMove` is called twice with equal momentum. Essentially, this means you are moving twice.

This can actually break the game quite well. In **Figure 4**, we see a level that contains a large amount of toxic blood. You can think of the level as a castle surrounded by a toxic moat. The red waste in this screenshot causes you a lot of damage, because it's supposed to keep you from skipping the castle that comprises most of the level. However, wallrunning allows the player to move so quickly that they can reach the other end with just enough health to survive to the exit. They essentially bypass the entire level as it was intended.

There is more strangeness in play here; it just so happens that if the player and the exit were reversed in this level, the bug would not trigger. That's because this bug only occurs when the player is moving in two of the four cardinal directions. Take a look at the code and see if you can spot why—again, it is all about that if statement in the loop. That if statement only checks to see if the momentum is greater than half of the maximum, so it will only call `P_TryMove` and `P_SlideMove` twice when moving quickly either north or east, since the other directions would be represented by a negative value. In the level shown in **Figure 4**, the bug worked because the player was moving north to the exit for the entire length of the toxic moat.

We could all see ourselves making this kind of mistake, and would probably never imagine that it would shatter world record times on certain levels. All you need to do to fix this is to either give `P_SlideMove` the correct momentum, or ensure that it is never called twice.

HEY, NOT TOO ROUGH We should not dwell on these failures; as you can see, it was some of the best programmers in the biz who made these mistakes. All of us are capable of writing code and only considering one case at a time (“Does moving forward work? Good. Does moving sideways work? Awesome.”), and never considering somebody will mash all of the keys at once. And on the bright side, sometimes, our bugs make a speedrunner's day! 

Dave Wilkinson, commonly known as wilkie, is a systems researcher who seems to break more code than he writes. In his free time, he runs the open-source game coding competition (<http://osgcc.org>) to introduce game development to students.

GROW IN THE GAME INDUSTRY



- Reference industry news and features
 - Consult your digital counselor
 - Play student games and join the forum
- VISIT YOUR YEAR ROUND MENTOR AT GAMECAREERGUIDE.COM



- Examine tutorials and exclusive features
 - Check out the Annual Salary Survey
 - Reference the premier Game School Directory
- DOWNLOAD YOUR FREE DIGITAL COPY AT GAMECAREERGUIDE.COM



- Learn from the pros
 - Attend deep-dive sessions with Q&A
 - Connect with your game making peers
- VISIT GOCONF.COM FOR INFO ABOUT THE NEXT SEMINAR AT GDC 2013

-- FOR PROFESSIONALS --



- Search for active junior to senior level jobs from 100+ leading game companies
- Upload your resume and get in front of direct hiring managers
- Organize your career research with your personalized Job Seeker dashboard

VISIT GAMASUTRA.COM/JOBS TO ADVANCE YOUR GAME CAREER





Are you awesome enough?

PROVE IT

Find us at GDC

» www.ccpgames.com/jobs

With offices in
Atlanta, Silicon Valley, Reykjavik,
Newcastle, and Shanghai



BLIZZARD[®]
ENTERTAINMENT



DIABLO[®]



**WORLD
WARCRAFT**[®]



STARCRRAFT[®]

BLIZZARD[®] IS HIRING

We are actively recruiting a new age of visionaries for key positions spanning a variety of skill levels across our game development and Battle.net teams.

SOFTWARE ENGINEERS:

SERVER | AUTOMATION | TOOLS | R&D | CONSOLE | RISK SYSTEMS
SECURITY | JAVA | ENGINE | USER INTERFACE | GAMEPLAY | GRAPHICS

3D ENVIRONMENT ARTISTS | 3D CHARACTER ARTISTS | TECHNICAL ARTISTS
GAME DESIGNERS | USER INTERFACE DESIGNERS | USER EXPERIENCE DESIGNERS

JOBS.BLIZZARD.COM

UR.BLIZZARD.COM

Follow us on Twitter: [@blizzardcareers](https://twitter.com/blizzardcareers)



infinity ward

IS HIRING AT GDC

WANT TO JOIN THE STUDIO BEHIND
CONSECUTIVE BLOCKBUSTER HITS?

EMAIL US AT JOBS@INFINITYWARD.COM
TO MEET UP AT THE CONFERENCE.

WWW.INFINITYWARD.COM/JOBS

CALL OF DUTY

CALL OF DUTY 2

CALL OF DUTY 4
MODERN WARFARE

CALL OF DUTY
MODERN WARFARE 2

CALL OF DUTY
MW3

Love games?!

Join our team

King.com is the 2nd largest games developer on Facebook, with an audience of over 50 million. Now taking mobile by storm!

We're looking for fun, talented people to join our European games studio teams.

Roles:

Business Intelligence
Business Management
Developers
Graphics/Game Artists
Product Management

Locations:

London
Stockholm
Malmö
Bucharest

We have other great roles in Europe & San Francisco.
Full relocation can be provided. **Talk to us at GDC 2013 in the Career Pavilion or apply online.**

Visit: about.king.com/jobs





SLEDGEHAMMER GAMES

WE'RE HIRING

SLEDGEHAMMERSGAMES.COM

ACTIVISION®

SLEDGEHAMMER
GAMES

CALL OF DUTY

INNOVATIVE. ARTISANAL. OUTSTANDING.

We're always searching for new talent.

We are ArenaNet, creators of **Guild Wars 2**[®], which has sold over 3 million units since launch! We make online worlds and love shaking up the industry and we're looking for kindred spirits with drive, talent, and passion.

Visit our job openings page at www.Arena.net. We may be searching for you.



©2010–2013 ArenaNet, LLC and NC Interactive, Inc. All rights reserved. Guild Wars, Guild Wars 2, ArenaNet, NCSOFT, the Interlocking NC Logo, and all associated logos and designs are trademarks or registered trademarks of NCSOFT Corporation. All other trademarks are the property of their respective owners.

Track Your Game Development with

DevSuite

Integrated Quality Management,
Defect Tracking & Agile Development.



- Multisite Enabled
- Web, Windows and iPad Based Clients
- Small Teams Pay Maintenance Only!

DITCHING DIFFUSE MAPS

PRACTICAL IMPLEMENTATION OF PROCEDURAL
TEXTURING ON TODAY'S REAL-TIME TECH

As an artist, one of the most amazing things for me in the current tech generation is the concept of shaders. No longer do we describe surfaces merely by flat images that already incorporated the majority of the lighting. Our materials are vastly richer, with per-pixel lighting that's much more dynamic. Shaders have the biggest impact on the visual definition of the current generation of real-time graphics.

But the programmable pixel and vertex pipelines are capable of much more than just calculating normal map and specular contributions. The wide array of things you can do with a pixel is now at the fingertips of a much broader audience, including the visual content creators themselves, who finally have a chance to become the architects of their own tech. From animated wet surfaces in the MODERN WARFARE ship level to heightmap-based vertex and up-aligned blending popularized by UNCHARTED 2, shaders have further driven the visual splendor that is this generation of games, albeit at a price: We needed more textures and VRAM to define a single surface, and more people to make the assets that made the games look good.

When I come back to the games that I used to enjoy five or 10 years ago, I'm amazed at just how crude the technology was back then: the resolutions, the poly counts, the effects. Yet somehow I and millions of other people who played those games managed to enjoy them on all levels, including visually, and I think it's because people are dazzled by beauty, not just cutting-edge tech.

The visual world around us is infinitely complex, and we can never have enough resources to recreate every single process that shapes the world around us. (The image by Zhu Haibu on page 70 illustrates this point rather vividly.) Trying to distill what makes reality feel real and feel beautiful is key in balancing the quality of your art with the amount of time required to produce it.

This logic and small personal findings in my own work led me closer to the concept of procedural materials. For example, I kept noticing that you can get away with plywood, plastic, and metal having the same base diffuse texture as long as you sell their specular properties or wear and tear correctly. Or tileable textures diffuse at times could be just noise, if it wasn't for the AO on top.

Somewhere along that time I also stumbled upon a shader implementation of an interesting piece of old tech: gradient mapping. All it does is take a grayscale heightmap you provide, and paint it with colors you assign to different pixel heights (brightness values).

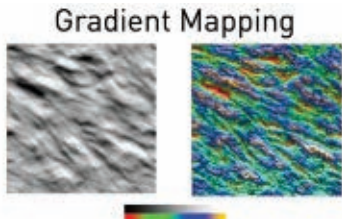


Figure 1: Gradient mapping.

For example, in **Figure 1**, all pixels with brightness from 0.0 to 0.33 will gradually transition from red to green, 0.33 to 0.66 will transition from green to blue, and blue eventually to yellow. You've seen that technology used in *LEFT 4 DEAD 2* by Valve, where it allows them to pack numbers of blood splatter and detail onto a single zombie texture, as well as to variably color them at runtime.



Figure 2: *LEFT 4 DEAD 2*.

PROCEDURAL ENVIRONMENT I really wanted to push this concept in a personal project, in order to see how far you can go with "procedural" materials even on current-day tech, so I made a demo clip called "The Desert" (see **Figure 2**).

Not a single surface here uses a dedicated RGB diffuse texture. In fact, I wanted to take it as far as manually inputting only one texture per one type of surface. (There were some other little textures buried in the shader, but they were just small grayscale masks stacked together.) The main idea in this procedural approach is separating surface volume from surface detail (see **Figure 3**).

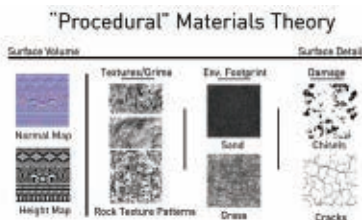


Figure 3: Breaking down the idea behind procedural materials.



IMAGE: ZHU HAIBO

Generally, rocks in the same area will have a similar geological origin, thus requiring a similar diffuse rock texture component. All objects in the same environment usually accumulate identical types of dirt. Objects made from the same materials will generally wear, tear, and decay similarly. Objects in a damp environment will start growing a similar type of moss. What differs is the diffuse texture pattern and how this dirt, moss, wear, and tear accumulate on a surface, which is governed by a heightmap.

OPTIMIZING STORAGE OF HEIGHTMAPS

Now whether we import our heightmaps as an alpha channel in a DXT5 texture, or use a separate grayscale texture, we'll still end up using another 128kb for a 512x map. We can stack three different heightmaps into a single DXT1 texture and save about 43kb, but there is a better way (see **Figure 4**).

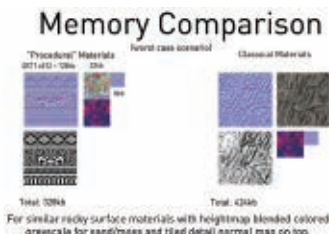


Figure 4: Comparing memory usage between the two approaches.

Normal maps' blue channel hardly stores any vital information, so why don't we put it

to a meaningful use? You don't even need to recreate the blue channel in-shader; merely replacing it with a neutral normal color works fine in most situations. All in all, you'll spend two additional pixel shader instructions, which is a puny price to pay for cutting the memory footprint in half. This is why further on in this article you're going to see normal maps and heightmaps listed as one single entity.

DAMAGE Now imagine you wanted to vertex blend damage to your material, as a lot of games do this day: **Figure 5** explains how we can do that.

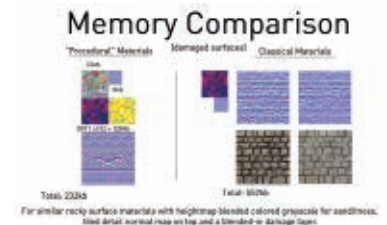
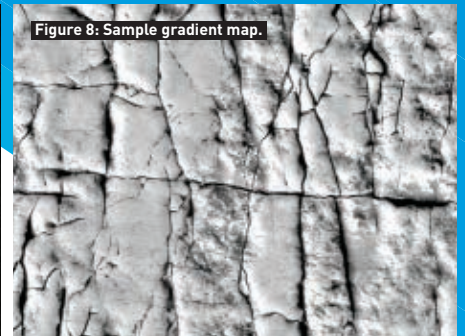


Figure 5: Comparing memory usage for damaged surfaces.

With this tech, you get your damage smartly blended only where it could exist in real life: on the most protruding parts of your surface volume. Now you use half the memory, and you get an opportunity to dynamically tile and tweak the intensity and available range of your damage, which will always be taken into account in every kind of heightmap-based blending further on. And you can change it for every single material instance, creating exactly the type of damage you need.



Figure 2: Still taken from "The Desert."
http://www.youtube.com/watch?v=QISbrlupvIQ&feature=player_embedded



MULTITUDE Now imagine that we need a bunch of similar surfaces for our level (as we usually do): **Figure 6** compares the memory usage between the two approaches. We save almost 50% with a procedural approach, and if we also did a damage pass for each of those textures, we'd be looking at 488kb vs. 1576kb.

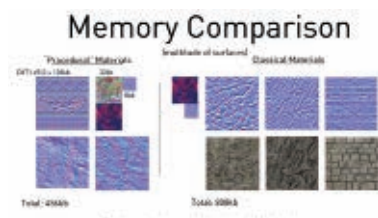


Figure 6: Comparing memory usage with multiple similar surfaces.

Now, just how many different-but-similar surfaces could we produce if we just keep

swapping the combination normal and heightmap? Almost all of them, as it turns out (see **Figure 7**).



Figure 7: Generating different materials by changing just one texture.

How is that possible? In the beginning, we said that the surface volume and diffuse texture patterns were an integral part of any material, but that doesn't mean they have to be separate entities. You don't have to strictly feed heightmaps to gradient mapping. In fact, feel free to forget the term

heightmap, because from now on it'll be a part of a broader term: gradient map.

GRADIENT MAP Gradient map is your main diffuse component, so you make it work as such: Blend your depth info with AO and every single texture pattern you might need, and paint in all the details and accents, or tweak surface values just like you would with a regular texture. In fact, I created a preview system in Photoshop that allows you to see the final material with all the normal and spec contributions right inside your Photoshop canvas immediately transforming as you paint your gradient map on it.

The process is even easier if you're working with heavily photo-sourced textures, because you can generate a normal map and a heightmap from your diffuse, then blend your grayscale diffuse map with your heightmap to create a gradient map, thus keeping both your surface depth and surface detail info. If you think



LET YOUR CAREER REACH THE PEAK!

Are you willing to have fun in the workplace and
change the future of the gaming industry in the world?

COME AND JOIN **PEAK GAMES!**

Talk to us at

hr@peakgames.net

peak
GAMES

www.peakgames.net

Figure 9: Comparing diffuse texture to two-point gradient-mapped grayscale version.



Examples of procedural coloring.

about surface detail, it is still depth, just on a much smaller scale, so the gradient mapping function processes it greatly (see **Figure 8**). Think of it as evaporating water from juice or soup to create a concentrate. Colors are water that we can spruce back in at runtime.

The gradient map works as well as the heightmap for blending, so no worries here. You can even use it as an opacity mask (for foliage, for example), as long as you make sure that your background is completely black, and your gradient has its level pushed up so it doesn't have any black pixels. Just clip the opacity mask from black pixels and that's it.

Another amazing fact is that gradient mapping works perfectly well even if you use just your usual diffuse turned grayscale. **Figure 9** shows a comparison of a material

from Epic, and the same material only with the diffuse map desaturated and put through the gradient mapping function. See for yourself: Do you think this difference really warrants a whole diffuse texture for this and every similar object?

The silver lining here is that you can even use your diffuse textures as gradient maps without any modification, significantly trimming your GM production speed.

PROCEDURAL COLORS IN TILEABLE TEXTURES VS. UNIQUELY MAPPED TEXTURES

As an artist, I was deeply concerned with how this approach would affect our color palette, but to my surprise, four colors for a gradient map are more than enough! I even had to create a lighter version of the gradient mapping function that just blends between two colors and

has no texture overlay. I've obviously used it mostly for environment textures, which are generally tiled, thus requiring certain uniformity from their colors to make the tiling seem unapparent, and that actually works greatly to the advantage of gradient mapping. If you're working with tiled textures a lot, you definitely want to try this out. [And if you're not working with tiling textures a lot, how the heck do your games even work?]

Also important to note: From an artistic standpoint, good lighting, fog, and post-processing greatly influence colors, usually creating the broadest and most important strokes. Your scene hardly ever is supposed to be about every little piece screaming for attention with a different color. Uniformity is good in a lot of ways, and it is definitely not something to be fighting with a lot of the time.

Figure 10: Textures from Uncharted 3 by Melissa Altobello.



Viewport screenshot of a LEFT 4 DEAD 2 character.

Now you don't have to take my word for what the textures should look like. To make it fair, let's analyze a real-life example and see what makes a high-quality modern-day texture (see **Figure 10**).

These textures are made in the good, old RGB-diffuse-map fashion, yet you can see that all that grunge and damage is really just an additional layer on top of the base textures. If you strip that away, the textures have a lot of shades of pretty similar colors; it's the overall color tone and the brightness of each pixel that describes them. I hope that previous examples have convinced you that "procedural" materials could do all of that (red brick/white brick, dirty/clean, damaged) in a matter of a few button clicks, saving you a lot of production time and memory as a bonus.

By now you probably think "Okay, what about uniquely mapped stuff?!" Do not despair: Valve has used it to create vast varieties of undead hordes for LEFT 4 DEAD 2, and so can you.

You probably won't want to gradient map your main characters, but there's still a lot of mileage you can get out of it. (Read all about it here: http://www.valvesoftware.com/publications/2010/GDC10_ShaderTechniquesL4D2.pdf)

The main hurdle is color variety; if you need a lot of color in a single texture, gradient mapping probably won't suit your needs. But hey, you can choose between gradient mapping and diffuse mapping and get the biggest bang for your buck.

RELINQUISHING CONTROL VS. UNEXPECTED VARIETY

Another tricky thing for me as an artist was fear of losing complete and utter control. Now I'm the first guy to be plain anal about every little piece of my artwork; I'll be carefully planting old pieces of chewing gum and cigarette butts on my textures in places where most people won't even bother. Yet throughout my career, I have constantly had to teach myself to choose my battles wisely.

I don't believe in things existing for their own sake. Anything is only as good as it performs its purpose, and the purpose of details is to be sufficient enough not to break the illusion of the imposed reality, as much as beginning artists think detail is all there is. I was and probably still am a guy who loves his details, yet I had to admit that most players will never notice a difference between gradient mapping and diffuse mapping, just like all the professional artists whom I've shown this environment to.

Only further down the road did I notice just how much flexibility I could get from this system. Plants, rock, bark—it's just a matter of swapping the only texture and tweaking the colors, diffuse patterns, damage, and specular values. One moment you're using this texture as an orange canyon wall, and the next moment that texture is already blue and is a part of a cave. Your bark is brown, but just turn it green, tile it more, and you have a vine texture. Need more cold color in your shadows? Just make them appear in the cavities of your surfaces to amplify the effect. Want it dirty? Just push a button and determine how much. Mossy or sandy, it's all just a matter of a button push. Procedural damage was another unexpectedly awesome thing, since I could reuse a single asset with different damage tiling and intensity and create a whole lot more variety than I could ever imagine doing the old way. I could even paint my cracks green and invert their intensity to create an illusion of vines overgrowing assets further in the distance.

As an artist I've come to love this workflow. I can't imagine not being able to tweak any object's color, diffuse noise scale or dirt at my slightest whim. It's like working with bigger LEGO pieces. Instead of modeling every part of our levels individually, we can create a set of modular meshes to work with, so why on Earth shouldn't we do it with our materials and textures?

PROCESSING POWER VS. MEMORY

CONSUMPTION While "procedural" texturing technology frees up a whole lot of memory, it also requires additional processing power. And as always with software optimization, it's a question of what you've got to spare.

The PS3 has just 256MB of VRAM, and it shows: Even the most gorgeous games sometimes put blurry textures right in your face. Notice the dramatic texelation difference in characters and the background truck in this *The Last of Us* screenshot. Yet the PS3 has eight SPUs that technically could be used to alleviate the issue by implementing gradient-mapping shaders, which would cut the diffuse texture footprint in half or even by 3/4, which means we could selectively increase texelation on some surfaces. The Xbox 360 has two SPUs and a combined RAM/VRAM module, yet it's still half the amount of memory of an iPad.



THE LAST OF US.

With the desert environment, I didn't use diffuse textures at all, just a couple of masks that are insignificant in terms of the whole level's memory footprint—so it would be fairly accurate to say that I cut my texture memory expenses in half by relying on this technology. It's also worth noting that to create damage, I use in-shader normal map generation from a grayscale mask, which is a somewhat pricey operation (though there are plenty of workarounds). Yet on a modern-day PC, the environment you've seen has no trouble doing 60+ FPS.

According to UDK's custom node instruction calculations, it would take us 13 instructions to replace a diffuse map with a gradient map. I have fully functional shaders that at 67 instructions provide full diffuse, specular, gloss, masked opacity, and normal functionality with only one texture sampled, where the most basic classical analogs clock at around 50 with three textures and zero in-shader flexibility. The most complex version of the "procedural" shader is 158 instructions, and features vertex-paintable, heightmap-based sand blending, as well as two types of procedural "smart" damage generation that are also both vertex-paintable. These instruction counts are at the very least comparable to the instruction

counts of Unreal Engine games materials with similar functionality.


PRODUCTION COSTS When it comes to production costs, there aren't any downsides to the procedural materials approach: You'll save a lot of production time, and subsequently, a ton of money. If I had to create every single diffuse map by hand, I can assure you that the environment you've seen would've been much smaller or taken more time to produce.

It's funny how when I explained this to a more business-savvy person, the first question I got was: "So how can many people can you replace?" This is definitely not about replacing people; it's about how much more and how much faster you can produce. There is always a lack of time in our industry, and having a chance to free some of it up for more important things is an amazing opportunity that yields better games and subsequently more profit.

ONLY THE BEGINNING For me, there's no doubt that diffuse textures simply aren't as indispensable as we thought. Gradient mapping carries enough information to make your brain perceive surfaces as completely believable, and that's all there is to it. So if

there's somewhere we can trim fat, it's in the diffuse texture. We can cut the workload and reduce the technical constraints, all without sacrificing visual quality.

If we as an industry want to keep moving forward, we should not only advance the quality of the product we produce, but also the quality of the production process itself. The industry slowly moves toward becoming more "procedural": We switched from animating dudes being blown away by a shotgun to simulating it; we have specific tools to build trees, roads, terrain, or LoDs; we no longer model every single piece of our levels, but rather create a set of highly modular LEGO-like pieces to build our levels with; gameplay scripting in UDK is now visual node-based editing—no more writing code! All of this saved our industry years of man-hours and millions of dollars in production costs. And I believe that this is something we should be doing with materials. Give our artists bigger LEGO pieces so they can dedicate themselves to the bigger picture with no real loss in detail. I would love to see next-gen engine creators take that into account.

Technology is meant to be a tool to achieve artistic results, so instead of making us try to keep up with the crazy amount of fidelity the next generation is about to bring, let's make it help us concentrate on adding details and creating real beauty where it would matter most. 

Andrew Maximov is a senior environment artist based in Montreal. His credits include ORDER OF WAR, WORLD OF TANKS, NATURAL SELECTION II, and MODERN COMBAT 4. Contact him at Andrew@ArtIsAVerb.info.

do it yourself

I've prepared a package with all kinds of procedural materials for you to check out, as well as a couple of example textures and meshes. There's also a .PSD that makes for very smooth gradient map production, as it allows you to preview your gradient mapping, normal, specular, damage, and diffuse pattern influence right in Photoshop! I've made a little video that will hopefully make things more visual for you [YouTube link]. You can grab the materials package here:

http://www.artisaverb.info/Desert/ProcMat_Demo.zip

http://www.youtube.com/v/a4TcnfX5c3Y?version=3&hl=en_US&rel=0

The 13th Annual Game Developers Choice Awards

Wednesday March 27, 2013 6:30-8:30pm [following IGF Awards] GDC Ballroom West Hall Street Level

Game Developers Choice Awards are the premier accolades for peer-recognition in the digital games industry, celebrating creativity, artistry and technological genius. Join us at GDC when we announce the 2013 GDCA winners live!

PRODUCED AND HOSTED BY



GAME DEVELOPER
MAGAZINE

CHOICE PLATINUM SPONSOR



Award Categories

- _ Game of the Year
- _ Innovation Award
- _ Best Debut
- _ Best Audio
- _ Best Game Design
- _ Best Technology
- _ Best Visual Arts
- _ Best Narrative
- _ Best Downloadable Game
- _ Best Handheld/Mobile Game

Honorary Awards

- _ Lifetime Achievement Award
- _ Pioneer Award
- _ Ambassador Award

STAY UPDATED AT

gamechoiceawards.com

*All badges allowed entry



ADC 13

APP DEVELOPERS CONFERENCE

CO-LOCATED WITH **GDC**
NEXT

APP DEVELOPERS CONFERENCE™

LOS ANGELES, CA
NOVEMBER 5-7, 2013
EXPO DATES: NOVEMBER 5-6

2013

ADCONF.COM



PLAYER-ACTING

HOW TO CHECK YOUR DESIGN BLIND SPOTS

I've been looking into psychology and its applications in game design over the past few years. It has been fun, and it has improved my design skills significantly, but when I bring it up, people ask me this question so often that I decided a few months ago that I needed to answer it once and for all.

The question: "But, what's it *good* for?" Sometimes, the people asking this question are actually curious. More often, the question is asked in a tone of voice that I've come to call "interventionary"—a mix of "I hope this doesn't insult you, but..." and "You really need to get over this fairy-tale stuff."

For a long time, the inanity of that question really bugged me. Why *wouldn't* we want to understand a player's motivation? Isn't it obvious? Then, about six months ago, I decided to actually answer it for myself, and suddenly, the question didn't seem so stupid.

Think about it for a second: Why is understanding human psychology so deeply *essential* to game design? Asking myself that question was pretty intimidating. We can talk in vague terms about better player metrics and better playtesting and better target player demographics, but are these things really essential? As it turns out, the answer is "No, they're not." After all, people designed great games long before anyone ever worried about player metrics.

So what is essential about connecting psychology to game design? After a month or two of brain-wracking, I managed to

find my answer. It is this: Psychological models are the best tools we have for showing game developers their *empathy blind spots*. Those who learn to overcome these blind spots will, slowly but surely, become better game developers.

I have a technique for doing this, something I call "player-acting" (thanks to Mike Capps for more-bettering the name). It's straightforward in concept: Find your blind spots, and then learn to enjoy the games that are loved by people with motivations in your blind spots.

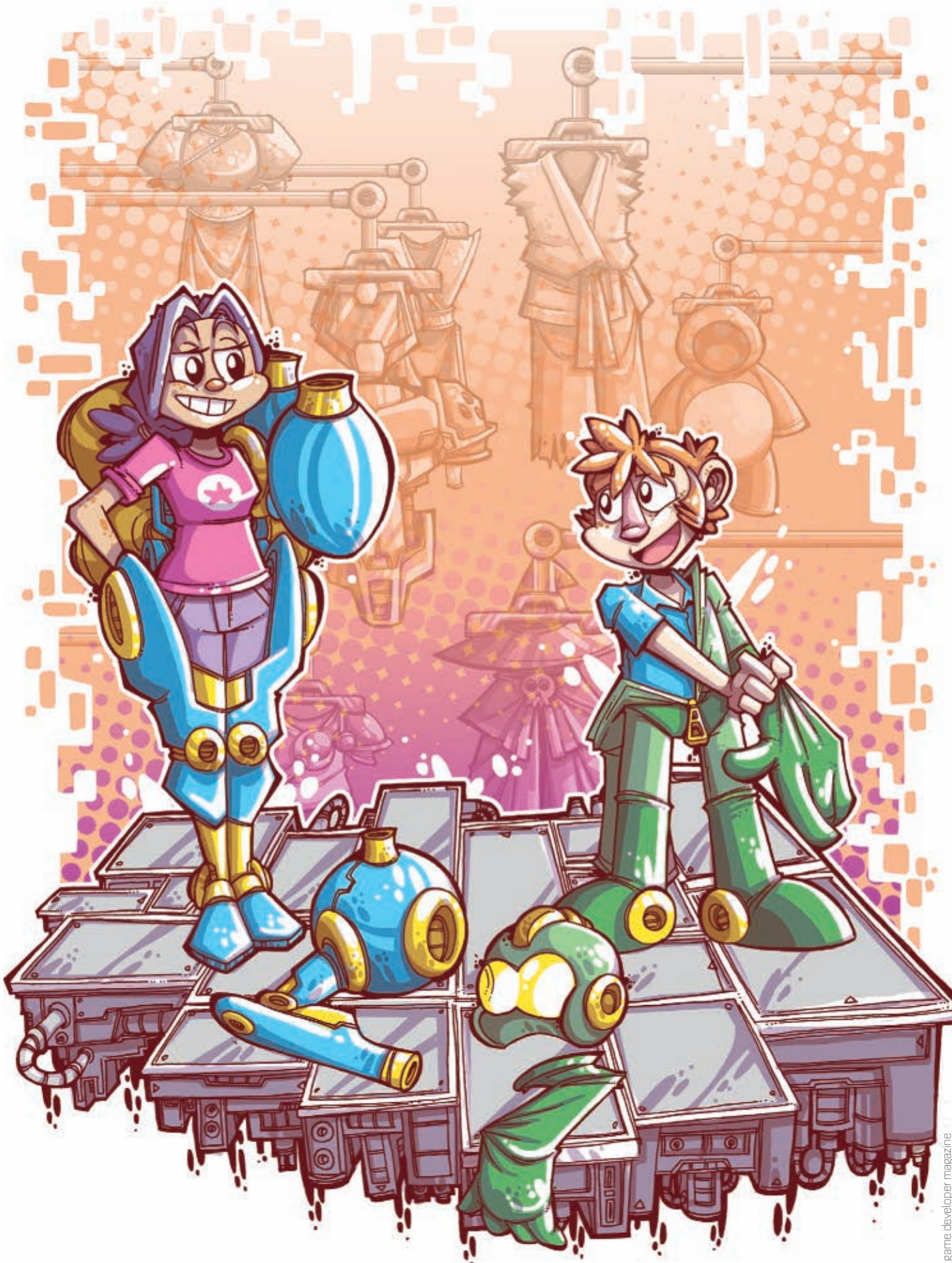
Let's break this down.

WHAT IS AN EMPATHY BLIND SPOT? Pick a personality test. A lot of people like the Myers-Briggs. I've recently become a fan of the Big Five. Really, it's almost irrelevant which theory you choose; what matters is that it's *substantial*. Pick something that's backed by long years of application, or by lots of scientists. Don't use that personality test you saw in the sidebar ad on that one website you surf. Don't use your colleague's pet theory. That way lies madness.

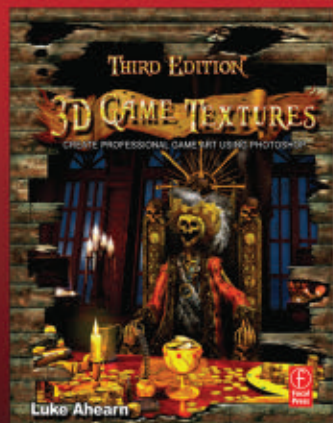
Whatever test you pick, take it. Then, take your results and reverse them.

Generally, personality tests rate you on a series of spectrums. The Big Five, for example, will give you scores in facets like "Adventurousness" and "Cautiousness," where the Myers-Briggs will score you on things like "Introvert/Extrovert." So, if you scored a high score in Adventurousness, you would reverse that score (giving you something they call "Desire for Routine"). If you scored Extrovert, you would reverse that to Introvert.



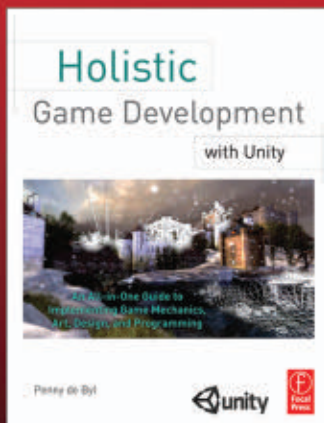


Focal Press has new tools for your game design toolkit



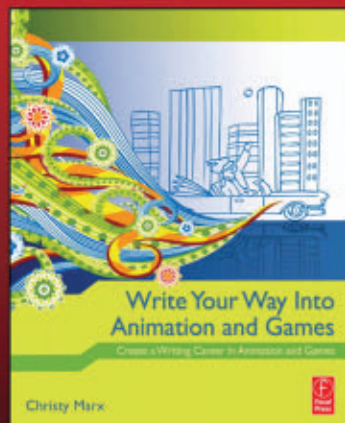
By Luke Ahearn

Learn everything you need to create stunning, professional textures. Give your digital art depth and breadth.



By Penny de Byl

An authoritative guide to creating games in Unity. Taking you through game design, programming, and art.



By Christy Marx

Launch your career in writing for video games or animation. Let our award-winning writers and game developers show you how to generate ideas and create compelling storylines, concepts, and narratives.

Visit us at GDC 2013 for special show discounts, give-aways, and more!

Booth # 306/308



Creativity has an endless shelf life.
focalpress.com



The behaviors described by those reversed scores are your *empathy blind spots*: human behavior that you will have to work the hardest to understand.

Okay, cool! We know where our blind spots are. Great! But how do we convert that knowledge into better design skills?

WHY YOU LIKE WHAT YOU LIKE In order to make sense of my argument here, I need to first establish this fact: Over the past few years, research findings have emerged that show that personality tests like these can predict your game preferences with pretty reasonable accuracy.

What this means is that if you want to understand why people like a particular gameplay genre or mechanic, their motivation profile is an excellent source for that understanding. This idea is built into this entire article, but it isn't something that every designer takes as granted at this point. [Yet.] If this is challenging for you, I encourage you to take a look at Nick Yee's Daedalus Project, my own work on the Five Domains of Play, and the research coming out of academia on this topic.

MAYBE... ASK? One approach to curing your empathy blind spots is to find people who have the personality traits you lack, and interview them. This method may sound basic, but it works. It *does* require a lot of listening, but you probably need practice in that anyway, and this approach is especially fruitful when attempting to overcome the initial "Why in the hell would *anyone* ever behave like that?" question that designers so often ask themselves.

For example, if you are an introvert, go find an extrovert, and find out what it is about being around other people that gets them charged.

If you can't wrap your brain around the idea of spending actual money on a free-to-play game, go find someone for whom that is not the case and ask them (with sincerity) what *positive value* they are getting from that behavior.

If you can't conceive that anyone would want to skip every piece of backstory your game presents, then find someone who does that, and talk to them until you uncover the positive value that they receive from their behavior.

Now, while this "just ask them" approach can get you a certain distance, does it actually make you a better designer? My experience is "not really." That moment where you go "Huh! Well, I guess that makes sense," is easily forgotten, and, more importantly, it's not attached to a personal gameplay experience. It's a place to start, but we need something better.

WALKING A MILE IN YOUR PLAYER'S SHOES In order to really empathize with the people in your blind spots, you'll need to learn to enjoy games for the same reasons that those people enjoy games. I call this skill *player-acting*: the skill of playing a game as though you were someone else. I don't know the *best* way to develop this skill, but I do know how I did it, way back when I was getting started in this crazy career.

Early in my career, I was entranced by the various player-type models (Bartle, Lazarro, and so on). I was blown away to find that there were gamers who were not like me. And yet, they liked games! I was struck by the idea that, maybe, just maybe, the games that I hated that other people liked weren't simply terrible games (and players who liked them "wrong"). Maybe, I thought, there was another set of rules and values out there that those players were living under!

However, I lacked any kind of practical understanding of what those rules might be, because like much of the rest of the planet, I had spent most of my life playing only the games that I *preferred*. I knew next to nothing about what made the games I didn't prefer awesome. I had never played them.

This was way back in the Pliocene Epoch, before there was a viable indie scene or anything resembling a Kongregate. I was at EA Redwood Shores at the time, and they did have a thing called the IRC (Information Resource Center). It was a well-shelved room, filled to the rafters with games. There were books and movies and magazines

there, too, but the majority was nothing short of every major game released during the previous decade, available for checkout.

One afternoon at lunch, I had an idea. What if I *played* all those games—but instead of playing them as *myself*, I tried to play them as if I were a different player type?

I launched into my "research" project with youthful abandon. Being the obsessive-compulsive completionist I am, I started at "A," and over the course of roughly two years made my way through to "Z." I played good games, bad games, action games, puzzle games, adventure games, sports games, kids' games, outdated games, new games, games for girls, games for boys, real-time and turn-based strategy, PC, console... everything.

Does that sound like fun? It shouldn't; just go to your local game store and imagine what it would be like to play *everything on the shelf*, without regard to your personal taste. Sure, you'll find some fun stuff in there, but an awful lot of it is going to be off the mark for you. This, of course, is the whole point.

This project was so not-fun, in fact, that I had to create rules to force myself to play. I had to give everything I tried at least one solid hour of play, no matter how bad or how buggy or how frustrating it was. After an hour, if the game had shown me at least *one thing* that was new to me (good or bad) or if I had some insight about the kind of player who might enjoy this game, I had to play a second hour. After that, I was on my own recognizance.

Many games got a solid two hours of play. A lot didn't. I found a lot of gems, one-hundred-percented a bunch of really good ones, and struggled through a huge number of unfinished disasters.

Most importantly, I played a lot of very popular, well-made, high-production value games that I didn't naturally like very much. I'm not a sports guy, but I put a lot of time into MADDEN, NASCAR, NHL, and so on. I had not spent much time with the BARBIE franchise up to that point, I must admit, but there were a stunning number of them on the shelf to plow through. I also wasn't much of a military shooter fan going in, but I sure was when I came out!

LAB RESULTS ARE IN While playing, I held all the models of player motivation that I knew in my mind. Was this game catering to the Achiever? The Socializer? What kind of person would like this game?

Slowly, game after game, I eventually started to perceive the gameplay elements that were there for people who weren't like me. All the WWII games seemed to use the same few guns, which made sense, but they put a lot of effort into making them feel a particular way, which felt clunky and frustrating to me, but it was consistently clunky and frustrating, in a way that I later learned feels about right for those guns. Me, I didn't care if it felt right—it was a fantasy, right? Who cares if the guns feel realistic? Well, as it turns out, a lot of people care.

Slowly, I increased my skills in playing games I wouldn't have otherwise played. And, gradually, at work and among my friends, it became easier and easier to understand what people were getting out of the crazy games they were playing. FINAL FANTASY fans started making sense to me. MICROSOFT FLIGHT SIMULATOR didn't seem like such a stretch. I started to be able to enjoy wider genres than I had before, because I could see their value.

PLAYER-ACTING HOMEWORK So, this is my point. Do you have empathy blind spots? Hit the Flash games, the demos, and the free-to-plays until you start to get your head wrapped around why people like them. Play the five-star earners in genres you despise. Read the comments of fans of those games, and try to get into their headspace. Break up your belief that other people are like you, and become a better designer. And it's also a great excuse to play a lot of games. **d**

Jason VandenBerghe is a creative director at Ubisoft, which he has to admit doesn't exactly suck. You can read his intermittent blog and various scribbles at www.darklorde.com. He can be reached by email at jason.vandenberghed@ubisoft.com.



MunkyFun is located in Historic Jackson Square in the Heart of San Francisco.

Connect with us at:

MunkyFun.com



@munkyfun



www.facebook.com/MunkyFun

Open Positions

- + Sr. Network Engineer
- + Core Engineer
- + UI/UX Designer
- + Gameplay Engineer/ Designer
- + Data Analyst
- + Mobile Social Project Lead



Bringing Next Gen to Mobile

MunkyFun is comprised of both veteran game developers and some of the smartest people from the social space. Some of our Munkies have been making games since the 80s and MunkyFun as a company has been making mobile games since the early months of the App Store. We bring console grade production values to mobile free-to-play in order to provide our ever discerning gamers with a high quality experience.

MunkySee MunkyDo

Team play is big around these parts—especially when it comes to making great games. We are proudest of our GameJams—a regular event that involves the whole company movin’ their desks into all new team formations (cells) in order to prototype new games. Every morning, those games are presented to the team where we all SEE what people are working on and give some feedback. Then the teams go off and DO! It has led to some of our best games—coming from collaborations we never would have expected! If something is not going in the right direction we nip it in the bud as early as possible. Success comes from failing fast.

The Willy Wonka of Game Development

We work in a chocolate factory. Literally. It isn’t still used as a chocolate factory but some say if you lick the walls...there is still a hint of cocoa. We have not tried it. But you are welcome to if you work with us. We do promise to provide chocolatey snacks.

Basically...

Our focus is on fun. We aren’t traditional, we aren’t stuffy, and we aren’t afraid to make mistakes.

Join us in the game development revolution where everyone is a designer and rapid iteration is king.



OBSIDIAN

entertainment

Winner of multiple industry awards, including RPG of the Year for Star Wars: Knights of the Old Republic II, and Fallout: New Vegas, Obsidian is dedicated to the development and advancement of role-playing games for PC and console platforms. Obsidian was founded in 2003 by industry veterans Feargus Urquhart, Chris Avellone, Chris Parker, Darren Monahan, and Chris Jones who are best known for their work on critically acclaimed classics such as Fallout and Planescape: Torment.

Located in Irvine, California, Obsidian is minutes from beautiful beaches and just over an hour from the mountains and deserts. Irvine is home to excellent schools, wonderful housing communities, and countless bike and hiking trails. Boasting nearly year-round sunny weather, Orange County offers the tropical living you see in movies filmed in Hollywood, only Irvine sits on the safe side of the San Andreas fault - so when the big one hits, you and your family can explore the ruins of earthquake-decimated LA (less than an hour's drive away - not counting vault-dweller traffic).

Obsidian is looking for YOU!

Obsidian Entertainment is always looking for creative individuals interested in a challenging environment creating the best role-playing games in the world. If making the next generation of role-playing games in a friendly, enjoyable atmosphere is what you're looking for, we want to hear from you!

We're currently working on one of the most anticipated RPGs of 2013, South Park: The Stick of Truth, as well as the currently most funded video game on Kickstarter, Project Eternity. We continually iterate on our own suite of RPG dedicated tools and technology to evolve the RPG genre.

Obsidian Entertainment has the best of both worlds: cutting-edge projects of a large studio combined with the family feel of a smaller company. We emphasize employee development, while encouraging open communication both within and across projects.

Benefits at Obsidian

Being a part of one of the few remaining independent developers in the world creating role-playing games is only the start! Obsidian offers many benefits to its employees:

- Competitive salaries.
- Immediate 401k with matching and no vesting periods.
- Complete comprehensive employee health coverage, including vision and dental. Health coverage for family as well.
- Paid vacation, sick leave, holidays (including special company holidays) and an additional week off during the December holidays.

Apply

If you think you have the chops (that is, experience, qualifications and passionate desire) to create AAA role-playing games, send us a cover letter, resume, and samples of your work (portfolios and demo reels, sample code, personal websites or anything else that demonstrates your abilities) to:

jobs@obsidian.net



Obsidian Entertainment, Inc.
8105 Irvine Center Dr Ste 200
Irvine, CA 92168 USA

jobs@obsidian.com
www.obsidian.com



we are looking for

- + Programmers
 - Mid-Sr Level Engine
 - Mid-Sr Level Gameplay
 - Mid-Sr Level Network
- + Artists
 - Senior Environment Artist
 - Senior 3D Artist
 - Concept/UI Artist
 - Senior Technical Artist
- + Producers
 - Mid-Sr Level Producers (Internal experience please)

GDC 13

GAME DEVELOPERS CONFERENCE® 2013

SAN FRANCISCO, CA | MARCH 25-29, 2013 | EXPO DATES: MARCH 27-29, 2013

JOIN US FOR FIVE DAYS OF SESSIONS, TUTORIALS, BOOTCAMPS, AND ROUNDTABLE DISCUSSIONS ON A COMPREHENSIVE SELECTION OF GAME DEVELOPMENT TOPICS TAUGHT BY LEADING INDUSTRY EXPERTS.

Learn

SUMMITS AND TUTORIALS [MONDAY & TUESDAY]

- AI
- Free to Play Design & Business
- Game Narrative
- GDC Education
- Independent Games
- Localization
- QA
- Smartphone & Tablet Games

MAIN CONFERENCE SESSIONS [WEDNESDAY-FRIDAY]

- Audio
- Business, Marketing & Management
- Game Design
- Production
- Programming
- Visual Arts
- Monetization [SPONSORED]
- Game Career Seminar [FRIDAY]



network

**GDC
Play**
[TUESDAY-THURSDAY]

Business Center
[WEDNESDAY-FRIDAY]

Career Pavillion
[WEDNESDAY-FRIDAY]

inspire

 **INDEPENDENT
GAMES FESTIVAL**
[MONDAY-FRIDAY]

 **game developers
CHOICE AWARDS**
[WEDNESDAY]

Expo Floor
[WEDNESDAY-FRIDAY]

VISIT GDCONF.COM FOR MORE INFORMATION.





EVERY YEAR I BUY THE MAGIC BEANS

THE PAINS OF BEING FREELANCE AT HEART

In the story “Jack and the Beanstalk,” a poor farmer’s son sets off for the big city to sell his family’s last cow in order to provide for his aged mother. Along the way he meets a peddler who would have him trade his last cow for a handful of magic beans, promising riches beyond measure and glory without compare. Berating Jack for allowing himself to be swept up in a dream, his mother tosses the beans out the window, and from them spring a beanstalk leading to adventure and opportunities. In my life working freelance, every year I take the risk and buy the magic beans.

PROMISE IN THE AIR As each new year begins (and really, after each contract ends), I can see the landscape of potential stretch out over the forthcoming months. If I’m lucky, this means that I’ve had some conversations or heard some whispers in the wind that could one day bear fruit. We all know that potential doesn’t pay the bills, but the process of turning potential into pay is often about maintaining a positive feedback loop between people. I say feedback loop because it is often this discourse, this give and take, in a relationship that helps to grow strong roots between friends or colleagues. For me, it means that every conversation is like planting the magic beans or cultivating ideas which could result in potential opportunities given the right combination of time, magic, and growth that the relationship inspires.

This process entails more than just talking, though; these beans are the myriad thoughts, conversations, articles, and inspirations... and not necessarily just the ones related to game audio. They are the kindnesses you extend to others, the support you offer to those in need, the

ability to speak or listen with compassion and sensitivity in a constructive way, or the clarity to honestly express your feelings in a nonconfrontational way. Whether these exchanges are with co-workers, mentors, people within the community, or just “regular” people, the beans you plant by engaging in these discussions continue to grow throughout your career (and life). They could be forum posts lending a well-tuned ear to someone just starting out, a blog you create to educate people on a technique or tool, or simply a spirited discussion between passionate people.

YOU HAVE TO BELIEVE Unlike plants, there’s no telling how much time you’ll need to grow a new opportunity. It is the unknown, unexpected, and unforeseen machinations that erupt from nowhere and inspire growth, and that is powered by your faith that the beans we sow will grow into opportunity. That is to say, it’s not entirely up to the magic what will come of the beans that have been planted, but the beans themselves and the positivity of their cultivation.

It’s not uncommon to wait years for a rare or exotic flower

to bloom, and much is the same, in my experience, with magic beans. Surrounding yourself in an atmosphere of constant gardening means continually focusing on the road ahead and giving attention to the beans that you’ve planted. (Think how many times have you seen a blog, Soundcloud, or tutorial started with the best intentions only to find it overgrown with weeds a year later with nothing but the original post to show for motivation.) It’s one thing to plant a bean and another to maintain and grow it. As a freelancer, this is the same in your professional relationships; you must continue cultivating new ones while exercising special care and handling for your existing ones.

BRING ALL YOUR DREAMS TO LIFE It’s something I keep telling myself year after year: All of the time and effort I spend sharing my passion unconditionally will return to me in the future (though perhaps not as golden eggs, magic harps, or sacks of gold pieces). I feel that I have experienced the direct result of my efforts returned to me when someone thanks me

for providing some guidance or is enabled by some of my frantic technical ramblings. Whether it’s the financial reality of putting yourself in front of people at conferences or the emotion of working with a team through the most difficult time of development, every year I go “all in” and roll the dice on making the magic happen.

Seeing the fruits of that labor is its own reward. It’s the kind of thing that drives me to work even harder between contracts to keep up the momentum on personal or community projects, or reconnect with people to hear about what they’ve been up to. This can’t be overstated, because people are the one constant in this industry. If you invest your time in caring about people, your investment will always inspire magic in your career. **af**

“You have to believe we are magic, nothing can stand in our way. Don’t let your aim ever stray.” —Olivia Newton-John & Electric Light Orchestra

Damian Kastbauer is a wandering minstrel of game audio, traipsing across the land at LostChocolateLab.com and on Twitter: [@lostlab](https://twitter.com/lostlab).

CRYSTAL BALLS VS. OUIJA BOARDS

GETTING BETTER AT PREDICTING THE FUTURE

The old adage “The only constant is change” is especially true in the games business, where new platforms and business models seem to materialize on a daily basis. Despite this (or perhaps because of this), many in the industry do a surprisingly poor job of attempting to predict the ways in which the business will shift. The results are painful, and clearly visible: canceled projects, studio layoffs, and wasted investment results when industry players big and small make bets based on a fixed worldview. As I write this, we’re heading out of the last round of year-end retrospectives on top trends and events of 2012, and heading into pre-GDC predictions for 2013 and beyond. Looking back on those from the previous year, it’s clear that we get these wrong as often as right—perhaps more so. Can’t we do better?

In my role at Intel, I have an acute version of this problem. The design cycles on silicon mean that we have to make decisions four to five years ahead of time about what games and other applications are going to require, and then make very large bets to prepare for that time. It’s not something we’ve perfected. In fact, like anyone in this industry, we’re often wrong, and managing the trade-offs and costs involved in correcting our course is part of the challenge. Independent analysts don’t fare much better, and are held with surprisingly little accountability to their track record, at least beyond one or two quarters’ worth of predictions.

That said, it’s important to try to gauge where things are heading before investing in your business’s future directions. I thought it might be interesting to review some of the methods people use in making predictions, as well as some of the common traps people fall into in trying to predict where things are going.

Personal litmus test (“That’s what I/my mom/my kid likes”): This is probably the most common method of “prediction” that people use, particularly when evaluating a consumer product, especially games. Almost everyone in the business is a game player, so it’s only natural to first gauge the inspiration for a product opportunity, or potential of a product in development, by your own personal desire for that product.

This is a specific form of Forecast by Analogy (more on this later), which I believe generally has great value. The analogy, in this case, is in comparing one consumer to the target market population, and there are a couple risks in doing this. First off,



the old saying “The plural of ‘anecdote’ is not ‘data’” applies here. Secondly, the further one’s target market is from that anecdotal example, the more room for error; assuming that kids in another country or continent will like similar things to your children, for example, could be quite a stretch.

Extrapolation: This is one of the more common forms of forecasting, and we often see it employed in simple market forecasts going out three to five years from the present; sometimes they’re just linearly extrapolated, and sometimes they’re based on an exponential curve. The basic flaw with this is so obvious that it’s surprising how often it’s given undue trust: The forecast is entirely based on the past performance of the factor we’re attempting to predict! Markets and products don’t exist in a bubble. Past performance is a factor, but only one of many. Meanwhile, markets are influenced by so many outside forces that can cause an inflection point in that extrapolated line at any time that it is unwise to lean too heavily on this method of forecasting.

Forecast by Analogy: This can take many forms beyond our example above. It comprises any time a comparison is made to a product, market, business model, or other market factor that shares some common characteristics with another market. For example, someone examining an all-you-can-eat subscription business model for games might start by comparing with all-you-can-eat subscription music services.

Cyclical models: There are numerous models that people have used to describe a repeating pattern of how certain market aspects behave over time. The Technology Adoption Life-Cycle proposed by Joe Bohlen (and others) and later improved by Geoffrey Moore (http://en.wikipedia.org/wiki/Technology_adoption_lifecycle) is one oft-used model that is useful in describing the characteristics of consumer adoption of a technology or class of products. Daniel Cook's Platform Power cycle (<http://www.lostgarden.com/2011/03/gdc-2011-game-of-platform-power.html>) is another great example that everyone in games should read and consider.

The downside of applying models like this is that they don't always translate between markets, and even within a given market, the rules change. For example, until recently, it was generally agreed that consoles behaved according to a five-year life cycle. However, with the Xbox 360 and PlayStation 3 having shot well past that point, it's now clear the rules have changed—though what they have changed to is still unclear.

Trend-influence: The idea here is to look at outside influencers of fundamentals to the business you are evaluating—rising

development costs, or falling LCD display costs, for example—and to apply some judgment as to the impact these will have.

Follow-the-money: While all the above will typically incorporate financial research, the techniques I refer to in this category involve analyzing the specific financials of individual players in the market, and using that information to build scenarios of possible outcomes. One can use financial results of players in the market as a trailing indicator of market performance and to speculate as to next moves. Also, investments in the form of VC and M&A activity can serve as an indicator as well.

Ultimately, all of these methods serve as tools to do various kinds of scenario building. All of them have their place, but none of them is perfect, especially if used on its own. Those attempting to make predictions should use multiple methods to see how they agree, compare with peers if possible, and most importantly, carefully document all the inputs used and the assumptions made, and revisit them regularly. Also, go back and visit past forecasts, asking where you went wrong and how it can be improved. Nobody can accurately predict the future, but we can better the odds of aiming in the right direction. **b**

Kim Pallister works at Intel doing game industry forecasting and requirements planning. When not prepping the world for super-cool hardware, he blogs at www.kimpallister.com. His views in this column are his and do not reflect those of his employer.



BECOME A LEADER IN DIGITAL MEDIA

With digital media in mind from conception to completion, the new CENTRE FOR DIGITAL MEDIA features student apartments, project rooms and classrooms all designed to inspire creativity and collaboration. Located in Vancouver, Canada the new CENTRE FOR DIGITAL MEDIA offers a full and part-time Master's program that focus on real-time, industry-facing collaborative projects.

Learn more about our MASTERS OF DIGITAL MEDIA PROGRAM at: www.thecdm.ca/gd

The future of work is at the new CENTRE FOR DIGITAL MEDIA.





entertainment[®]
software
association

E3



WHAT'S JUNE 11-13, 2013
LOS ANGELES, CALIFORNIA
NEXT NOW
E3Expo.com

BE A PART OF WHAT'S NEXT IN
THE VIDEO GAME INDUSTRY.

Register at E3Expo.com

Produced By
 IDG
WORLD EXPO

E3 is a trade event and only qualified industry professionals may attend.
No one under 17 will be admitted, including infants. Visit www.E3Expo.com for registration guidelines.
© 2013 Entertainment Software Association

International
Media Sponsor

MCV



READ-ONLY MEMORIES

HOW DO YOUR EARLY GAME EXPERIENCES INFORM YOUR DESIGN PREFERENCES?

They say you're forever dating your first love. Not literally, of course, but the early patterns set by your first relationship, and the relationships of your parents, tend to strongly influence how you approach love and relationships for many years to come. I wonder: Is the same true for games? Do those early games we played in our formative years influence what we now perceive as "good" and "bad" in interactive media? Do they influence how we design games? I submit that they may.

FIRST KISS IS DEADLY Let's think about a series like *DARK SOULS*/*DEMON'S SOULS*. These games are punishing, require rather exacting inputs from players, and have somewhat fiddly controls that require getting used to. That sounds like a nice recipe for a failure stew. So why did these games succeed?

One of the praises you often see from reviewers is that the series reminds them of the glory days of Japanese console and arcade games, which were built with much the same recipe. It's like a new love affair with an old flame—the same problems as always, yet sweetly, lovingly familiar. Japanese publication Dengeki said of *DEMON'S SOULS*, "Fans of old-school games will shed tears of joy." IGN reviewer Sam Bishop echoed the sentiment, saying, "Those that can remember the good ol' days when games taught through the highly effective use of intense punishment and a heavy price for not playing it carefully should scoop this up instantly."

But what about people who didn't grow up with that experience? What about those who are more used to frequent checkpoints, and the game providing a full experience to blaze through in one go, rather than in halting steps? For them, the game is a harder sell, which is why Sony passed on publishing *DEMON'S SOULS* in the West, and core-oriented niche publisher Atlus had to step up and do it instead.

For *DEMON'S SOULS*, its link to the past helped it succeed. But perhaps the reverse can also happen: Our personal game heritages could, at times, make us slaves to our past interests. For example, I tend to like games that are interesting, but flawed. To me, a glitch in an otherwise super-polished *CALL OF DUTY* is extremely glaring and illusion-shattering, but I'll happily forgive poor graphics and the occasional invisible wall in a game like *NIER*, which stabs out in all directions with new ideas. If a game tries hard to do something different,



I'll forgive its faults—and if I want to be a designer who makes games that are good at making money, this preference for different-but-flawed could hold me back from making games with commercial appeal.

With this thought in mind, I decided to dissect my own past as a player to see what influence it might have had on my current interests.

LESSONS FROM THE TURBOGRAFX-16 My history is a bit odd—I went from the 2600 and Intellivision (which were already old when I got them, but they were affordable!), to the TurboGrafx-16, which I saved up for months to afford. And this is the console that informed my early days as a player of games.

THE *VALIS* series, for example, is not very well known, but I played it to death. It's an action, platforming, hack-and-slash affair that stars a high school girl, out to save the world, with a sword taking on a horde of monsters. Pretty standard fare for the 1990s.

You could jump, perform a sword attack, use magic (and could power up both of these attacks), walk, and roll. I replayed *VALIS III* recently, and I noticed something about those rolls that may have influenced my current

GDC 13 NEXT

CO-LOCATED WITH **ADC**
APP DEVELOPERS CONFERENCE

LEARN . NETWORK . INSPIRE .

GAME DEVELOPERS CONFERENCE NEXT

LOS ANGELES, CA
NOVEMBER 5-7, 2013
EXPO DATES: NOVEMBER 5-6

2013

GDCNEXT.COM





interests and design habits. Rolling allows the player to travel for a set distance, both under obstacles and across gaps. But this distance is such that, at times, beginning a roll just a few pixels one way or the other means life or death in a difficult platforming section. On top of that, the platforms themselves can occasionally have dressings that don't count as area you can stand on.

This is most likely something one would want to avoid in the modern era, because it feels like the game has tricked you, when you've clearly made the roll visually, but it's counted as a death. Less obvious, though, is the triumph you feel after defeating that particularly difficult section. It's as though you've succeeded in spite of the game's efforts to thwart you. You are actually fighting against the game itself, which we're generally told not to do—but in a modern game like *DEMON'S SOULS*, it makes the thrill of victory that much more compelling.

There is a lesson here for me as a designer: I can sometimes focus too much on making things smooth for a player in the immediate term, versus their long-term experience.

I won't bore you with my history as a player, but revisiting these old game-loves continually revealed patterns in my current thinking. For instance, *BONK'S REVENGE'S* somewhat mystical and alchemic systems helped drive me to chase the elusive beast that is emergent gameplay in a simple game world. But is that my white whale? That pursuit has, at times, led to feature bloat (which is exactly what happened in the subsequent *BONK* installment, incidentally).

RECONSTRUCTING OUR PAST Just to make sure I wasn't the only one who's influenced by his past, I asked my friends Tim Rogers of Action Button Entertainment and Frank Cifaldi of Gamasutra.com, with whom I record a weekly podcast (which is also called insert credit), to talk a bit about their formative games, and found them

similarly branded by past experience.

For Cifaldi, it was *THE SECRET OF MONKEY ISLAND*, which gave him the first glimpse of a full, living interactive game world. This colored his interest in games for years to come; when he was young, he made adventure games in *HyperCard*, and later, when he was working at GameTap, he made an interactive community adventure game called *CAPTAIN MCGRANDPA*.

Rogers, meanwhile, thinks *SUPER MARIO BROS. 3* is the best game ever made. *SMB3* is very much about precision and timing of jumps and reactions, but also about secrets—warps, hidden passageways, and coin boxes in the sky. It's no wonder, then, that the first game he directed (*ZIGGURAT* for iOS) is a deceptively simple game about timing, precision, and nothing else—aside from the occasional secret.

TELL ME ABOUT YOUR MOTHER 3

For your human relationship problems, you can go to a therapist—but they'll just reflect back what you already know. I highly recommend you take a self-analysis approach to your game history. Going back and dissecting those early learnings can help you grow past your earliest ideas of what a game is, or can be, because while most lessons will be good, some will be bad as well.

The musical platformer *SOUND SHAPES* is an interesting case study: If you read the postmortem in the December 2012 issue of *Game Developer*, you'll see that the game's mastermind, Jon Mak, said, "I don't like platformers, or level editors, but in the back of my mind they made sense." He also added, "That was a thing that we learned: We couldn't achieve our design goals with what we would do naturally."

So here is an example of developers playing against their type, and against their early imprint. This worked well, and brought *SOUND SHAPES* to critical acclaim, and multiple GDC award nominations. But


at the same time, is it any wonder that (sorry, Jon) the game just doesn't feel like a solid platformer? It feels like an interactive music toy where platforming happens to be the mechanic to drive progress. Without the music element, this would not be a loving homage to the platforming genre.

There are lessons in our past for all of us. Try it out on yourself; think about the first game that really grabbed you. Maybe it's the first game that compelled you to keep coming back, aiming for a perfect score; maybe it's the first game that made you feel like games were a living world; maybe it's the first game that let you play against another player.

Revisit these games with new eyes. While playing them, think about the jump distances for platformers, or how you start a drift in a racing game, and how long that drift lasts. Think about the level progression in RPGs, or the score multipliers in a shooter. How has your current work reinforced those old ideas? How have they strayed? Should you be more critical of those old ideas? It's an interesting exercise which can yield some surprising results. Even if you don't come away with something practical, you may have an easier time explaining why you prefer to sink hours into *MINECRAFT* over *SKYRIM*—or the reverse.

BACK TO THE FUTURE The kids of today expect autosaving, persistence, checkpoints, and massive interactivity on a *MINECRAFT* scale. And they're not wrong to expect it! That's what they grew up with, and that is to some extent the future of entertainment. But when they grow up, what will they expect from games? What will their first love affair teach them to love and hate?

Getting closer to the now, what about kids who grew up with the Nintendo 64? The precise magic of *GOLDENEYE 64* has never been properly revisited. What of a child who grew up with the Dreamcast? Is anyone serving her needs?

I'm not suggesting we need to mine the past and prey on nostalgia. But attempting to serve similar experiences to those people felt in their youth—in new and modern products—can be a valuable goal. Nobody wants to play a new game that's exactly like *GOLDENEYE 64*. They want to play a game that feels like how they remember *GOLDENEYE 64* at the time they were playing it. With a little self-analysis, and a careful study of these bygone eras of games, you might just get at that mystical and elusive feeling. 

Brandon Sheffield is director of Oakland, California-based Necrosoft Games, and editor emeritus of Game Developer magazine. He has worked on over a dozen titles, and is currently developing two small-team games for PlayStation Mobile. Follow him on Twitter via @necrosofty.

**INFORMING, ENGAGING, AND
EMPOWERING THE INDUSTRY**



gamasutra.com

the art and business of making games



the 15th annual

independent games festival awards

Wednesday March 27, 2013 6:30pm GDC Ballroom West Hall Street Level

The IGF celebrates the best, brightest, and most innovative in independent game development. Join us at GDC when we announce the 2013 IGF award winners live!

Award Categories

- Seumas McNally Grand Prize (\$30,000)
- Excellence In Visual Art (\$3,000)
- Excellence In Audio (\$3,000)
- Excellence In Design (\$3,000)
- Excellence In Narrative (\$3,000)
- Technical Excellence (\$3,000)
- Nuovo Award (\$5,000)
- Audience Award (\$3,000)
- IGF Student Showcase Winners (\$1,000)
The IGF's Student Showcase will highlight a total of eight games this year.
- IGF Best Student Game (\$3,000)

Judge for Yourself IGF Finalist games are playable at the IGF Pavilion on the GDC Expo Floor, March 27-29, 2013

View the Submissions at igf.com

IGF platinum sponsor



IGF platinum student showcase sponsor



IGF gold student showcase sponsor



IGF platform sponsor



CONTRACTOR CORNER

A Genius Alternative

SUPERGENIUS

To Art Outsourcing

Specializing in the 5 disciplines of video game art production

www.supergenius-studio.com

FORGE

STUDIOS

Creativity & Innovation

Tailored to your needs

www.forgestudios.com



Complete Art Development and Production!

- USA Based Sales & Customer Support
- 3 Production Studios in China - (North) Beijing & (South) Ningbo
- Professional Project Management & Production Communication
- Competitive Rates based on Overseas Production
- 3D - High Quality Assets for Game Dev & CG Animation
- 2D - Concept, Design & Illustration - Production Concepts
- Keyframe Animation - 2D and 3D, Layout, Rigging & Weighting

CENTURION
ART DEVELOPMENT

1-888-411-9336 or contact@centurionartdev.com www.CenturionArtDev.com - Online Portfolio -



**LOOKING FOR A GAME
DEVELOPMENT PARTNER?**

**WE BRING YOUR GAME
IDEAS TO LIFE!**



GET IN TOUCH NOW!

web: www.iLogos.biz

email: info@ilogos.biz

technicolor



Your Trusted Partner for
Animation & Sound

www.technicolor.com/GameServices



gamasutra.com

the art and business of making games



GDC Vault

UNLOCK A TROVE OF 8,000+ IN-DEPTH DESIGN, TECHNICAL AND INSPIRATIONAL TALKS AND SLIDES FROM GAME DEVELOPMENT INDUSTRY INFLUENCERS. ALL TAKEN FROM 20 YEARS OF THE WORLDWIDE GAME DEVELOPERS CONFERENCES.

STUDIO AND EDUCATION MEMBERSHIP PACKAGES ALSO AVAILABLE.

CONTACT GILLIAN CROWLEY AT [GILLIAN.CROWLEY@UBM.COM](mailto:gillian.crowley@ubm.com)

GDCVAULT.COM

ATTENTION: GDC 2013 ALL-ACCESS PASS HOLDERS
YOUR MEMBERSHIP NOTIFICATION WILL BE SENT TO YOUR IN-BOX NO LATER THAN APRIL 5, 2013.

ADVERTISER INDEX

For more information visit www.jointhegamenetwork.com

COMPANY NAME	PAGE #		
ACADEMY OF ART UNIVERSITY	053	RESEARCH IN MOTION CORPORATION	024
ACTIVISION	064 & 066	SCOTTISH DEVELOPMENT INT.	058
ARENANET INC	067	SHARP SHADOW STUDIOS	094
ART BULLY PRODUCTIONS LLP	094	SHERIDAN COLLEGE INSTITUTE OF	028
ASOBO STUDIO	C2	SIMPLYGON	054
BLIZZARD ENTERTAINMENT	063	SOCIETY FOR THE ADVANCEMENT	035
CCP GAMES- NORTH AMERICA	062	SUPERGENIUS	094
CENTURION ART DEVELOPMENT	094	TAYLOR & FRANCIS GROUP LLC	080
E3 EXPO	088	TECHEXCEL INC	068
ENCHANT.JS INC	020	TECHNICOLOR	094
EPIC GAMES	016	UBISOFT PRODUCTION	
FMOD BY FIRELIGHT TECHNOLOGIES	003	INTERNATIONALE	026 & 044
FORGE STUDIOS	094	VANCOUVER FILM SCHOOL	007
HAVOK	048	WARGAMING AMERICA	012
ILOGOS	094	WOOGA	032
KING.COM	065		
KONAMI DIGITAL ENTERTAINMENT CO LTD	008	IN ASSOCIATION WITH UBM TECH	PAGE #
MASTERS OF DIGITAL MEDIA PROGRAM	087	APP DEVELOPERS CONFERENCE	077
MICROSOFT CORPORATION	C3 & 043	GAMASUTRA.COM	092 & 094
MUNKYFUN, INC	082	GAME CAREER NETWORK	061
NVIDIA CORPORATION	032	GAME DEVELOPERS CHOICE AWARDS	076
OBSDIAN ENTERTAINMENT	083	GAME DEVELOPERS CONFERENCE 2013	084
PEAK GAMES	072	GDC NEXT	090
PLAYSPAN INC	011	GDC VAULT	095
RAD GAME TOOLS	C4	INDEPENDENT GAMES FESTIVAL	093

gd Game Developer (ISSN 1073-922X) is published monthly by UBM LLC, 303 Second Street, Suite 900 South, South Tower, San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as UBM 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: \$59.95; all other countries: \$69.95 (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 gd Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate gd Game Developer on any correspondence. All content, copyright gd Game Developer magazine/UBM LLC, unless otherwise indicated. Don't steal any of it. Or else.

I'VE HAD IT WITH THESE LAZY DEVS

A FORUM-GOER PUTS GAME STUDIOS ON NOTICE

Hey guys, I hate that I have to make this post, but apparently everyone else is too complacent to STEP FORWARD and say something about this. You all read that Stolen Sun Studios said AGAIN that the next update for WANDERER: HORIZONS won't include the "massively co-op" mode we've been suggesting in the forums for MONTHS now. Who do they think they're fooling? I don't buy their excuse for a second that doing it would be "too hard!"

It's not like doing massive co-op is that complicated. It's a new mode, but building the levels or modeling some new vehicles is probably the biggest thing. And it's not like you're building a brand-new game. You just make the existing polygons, textures, and pixels appear in everyone's game at the same time and you're basically done.

Anyone who has programmed before knows this isn't a big deal. All the actions in WANDERER are extremely simple. If the code for moving and shooting is more than a few lines long they did it wrong. Plus the netcode doesn't have to be put in over the whole game. There's "code reuse," which is a technique that lets you use the same code in a lot of different places. I'm sick of the excuses here.

HOW DO I KNOW IT? Someone asked how I know this. Well, I know this because I made my own games for at least two if not more of my computer programming classes in which I was the top student (i.e., college level). If it would take the devs more than a couple hours, well, then they have SPAGHETTI CODE and should reprogram everything. The only reason they wouldn't add it is if they don't actually care about the game. That's right, I said it: The lure of EASY PROFITS is leading Stolen Sun to take the loyal fans for a ride.

Anyway, it's not that I want massive co-op multiplayer personally—I find most "gamers" these days are too stupid or casual to play with. I just hate seeing game developers being so LAZY, succumbing to greed, living like fat cats eating caviar, and driving Bentleys. That's the real truth here, everyone. They've given in to GREED as all game developers do eventually. Remember when Birth of the Titans decided to charge for the "expansion" even though they had been doing FREE patches for over six months? Get real, noobs. Anyone who isn't blinded by BLATANT FANBOY BIAS can see through that.

@BlueBalo022 Well, clearly your college was in it for the money just like the WANDERER devs if they gave you a degree in computer science despite you clearly not even understanding basic programming. Basically everyone that plays WANDERER these days is going to be connecting to the internet using cable, DSL, or fiber to the home (FTTH). Not TCP-IP.



@LittlerFingers It's clear that you're not a coder at all, since you can't tell the difference between the game code (which doesn't need to support networking) and the engine code (which does). In games like this, most of the engine code is about rendering and audio, so adding networking to that is nowhere near as complicated. And once it's in the engine, the game functions just need a few new parameters (I bet you don't even know what those are) to hook into it.

GAME DESIGN ISSUES—OR NOT I know some people on these fine forums have said that massively co-op multiplayer would change the tone of the game from something about a lonely experience to something more like an MMO with people running around, etc. Well, that's easy too: Just have the characters occasionally say a line about how weird it is seeing another person around. It's not like they couldn't record more voices.

@SirMustache I'm not even going to bother responding to this.

Again, I don't care at all if this is implemented or not, I just don't want to see the programmers get away with being lazy and the loyal fan community just accepting whatever they say because they're not educated enough.

LAST POST BEFORE I *DELETE MY ACCOUNT***** Look, anyone who really cares about WANDERER needs to speak up about these SLOTHFUL developers and not accept their pathetic excuses. Everyone here should hold them ACCOUNTABLE and REFUSE to support them anymore until they come clean and admit they've been lying to us. I won't accept anything less and do not plan on returning to these forums until they do. Good day, all. **ad**

Matthew Wasteland writes about games and game development on his blog, *Magical Wasteland* (www.magicalwasteland.com). Email him at mwasteland@gdmag.com. Magnus Underland writes about games and other topics at www.above49.ca. Email him at magnus.underland@gmail.com.



Microsoft is changing the face of entertainment.

Come and learn how at GDC 2013.

- ➔ Smartphone and Tablet Games Summit
- ➔ Microsoft Lobby Bar & Product Showcase
- ➔ Over 25 Microsoft Sessions
- ➔ Microsoft Recruiting

IT'S HERE. BINK

2

Bink 2 Video has up to 6 times the quality than Bink 1 at the same bandwidth. It's also up to 3x faster due to it's SIMD design (70% of all instructions are SIMD in a frame decode) and perfect two CPU scaling. Available for Windows, Mac, Linux, (or any x86 or x64 system), Xbox 360, Playstation 3, PS Vita, iOS and Android.
www.radgametools.com 425-893-4300

