

11th annual salary survey

THE LEADING GAME INDUSTRY

APRIL 2012

gd

GAME DEVELOPER MAGAZINE

postmortem

kingdoms of amalur: reckoning

+
building bokeh
photographic depth-of-field on the PS3
prototype 2
magic behind the mayhem



ONLIVE®

YOUR GAMES. MORE PLACES.



MONETIZE YOUR GAMES ON TVS, TABLETS, PHONES, PCS AND MACS WITH **ONE BUILD.**

- ➔ Millions of users worldwide
- ➔ Industry-leading cloud technology
- ➔ Free instant demos on any device

VISIT ONLIVE.COM/DEVELOPER TO LEARN MORE

24 HOUR ON DEMAND ACCESS TO A LIBRARY OF DEVELOPER KNOWLEDGE.



GDC Vault

Streaming video, audio, and PowerPoint presentations from GDC 2012, GDC Europe, GDC China, and GDC Online.

For more information visit: www.GDCVault.com



POST MORTEM

28 KINGDOMS OF AMALUR: RECKONING

Making an open-world role-playing game isn't easy—especially if your studio is used to developing strategy games instead. Despite two acquisitions, senior management shuffles, and numerous false starts, Big Huge Games turned out a big huge game indeed. This postmortem explains how usability testing, production processes that kept developers accountable, and a good relationship with Electronic Arts made KINGDOMS OF AMALUR: RECKONING happen. *By Mike Fridley*

FEATURES

7 11TH ANNUAL SALARY SURVEY

We're back with our 11th annual game industry salary survey! Whether you're an indie developer, contractor, or a salaried employee, we'll help you find out how your earnings stack up against the average, how Europe and Canada stack up against the U.S.-based developers, and which development discipline brought home the most bacon in 2011. *By Patrick Miller*

13 FIRE, BLOOD, EXPLOSIONS

Creating effects for open world games is a significant challenge PROTOTYPE 2? Senior rendering coder Keith O'Connor walks you how the Radical Team built a particle effects system that made things burn, bleed, and blow up even better than before. *By Keith O'Connor*

21 NEEDS MORE BLUR

A good depth-of-field effect can help your player focus on the important things in a cutscene. Serge Bernier explains how he was able to reproduce the photographic "bokeh" effect with the PlayStation 3's SPU. *By Serge Bernier*

DEPARTMENTS

- 2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]
Rest Assured
- 4 **HEADS UP DISPLAY** [NEWS]
Game concepts from fridge magnets, and a look at WaterMelon's Magical Game Factory pay-to-play crowdfunding system
- 35 **TOOL BOX** *By Patrick Miller* [REVIEW]
The coolest tools and middleware on the GDC 2012 show floor
- 39 **PIXEL PUSHER** *By Steve Theodore* [ART]
See the Light
- 42 **DESIGN OF THE TIMES** *By Soren Johnson* [DESIGN]
More Than Zero
- 45 **THE BUSINESS** *By Kim Pallister* [BUSINESS]
Me of Little Faith
- 46 **THE INNER PRODUCT** *By Jelle van der Beek* [PROGRAMMING]
13 Ways to be a Better Lead Programmer
- 49 **GOOD JOB** *By Brandon Sheffield* [CAREER]
Q&A with Ben Sherman, who went where, and new studios
- 51 **AURAL FIXATION** *By Damian Kastbauer* [SOUND]
Knowing a Thing or Two
- 52 **GDC NEWS** *By Staff* [NEWS]
Independent Games Festival and Game Developers Choice Award Winners
- 53 **EDUCATED PLAY** *By Tom Curtis* [EDUCATION]
NITRONIC RUSH
- 56 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]
Quarterly Report



REST ASSURED

LET'S TALK ABOUT WHY QA SUCKS

Maybe you're an artist. Maybe you're a coder. Maybe you're a designer, or a producer. Maybe you're all of these. But chances are, if you fit into one of those categories, you have a career path of some kind set before you. You can become an effects lead—you can move laterally and become a tech artist. Learn some behaviors and become an AI-oriented designer.

Your QA team doesn't have a career path though. Quality assurance is one of the most important aspects of game development, and all we encourage our QA staff to do is move "up and out" of the QA slog, if we inspire them to move up at all. Many QA professionals talk about "getting out" and moving to the production or design path. Don't you want good QA folks to keep doing QA? Shouldn't they enjoy and want to start in their jobs? Isn't there something wrong with this picture?

GET BONUS

» Your QA staff is your front line of defense against bad reviews. Obsidian creative director Chris Avellone recently mentioned that the company didn't get a bonus for *FALLOUT NEW VEGAS* because the game missed its Metacritic target of 85% by one percentage point. They had to reduce staff as a result.

Game reviewers play a lot like testers sometimes. They push against boundaries, and find those things you're pretty sure nobody will ever bother to do. They complete quests in the wrong order. They jump over a wall and trigger a cutscene that wasn't meant to happen for hours. In short, they break your game, then call it buggy. What does that do to your Metacritic score?

Not all bugs can be prevented, but with creative testing and enough time (that's the kicker!) you can get most of the showstoppers. Everyone knows QA is important, but let's really think about that. Is your QA team inspired to think creatively, and go that extra mile? Are they treated like important members of the team?

Most QA is hired from a pool of fresh-faced kids who just want to get into the industry any way they can. They are passionate about games, but aren't sure how to break in, so they take the QA route. Maybe they've heard about game designers who started as testers. What they don't want to do is stay in QA forever. And why would they? The hours are long, and the pay is the lowest in the game industry, at under \$48,000 average across all years of experience, almost \$30k under the next lowest discipline. Many are hired on contract, at low wages, then get let go when a project is complete. Does this sound like a job you'd want to stay in? Or a job where you'd be incentivized to think creatively? Is this job with a career path? If the biggest incentive you're given is to become a lead and then move to another department, how much can you really care about working in QA?

How many times have I heard "the dev team" and "QA" spoken of as though they're different things? In many companies there's a physical wall between "the developers" and QA, if not an entire building. QA is part of the dev team. Why is there this mental space between the two?

It's a self-fulfilling prophecy. If you think the QA kids are scrubs, stop hiring scrubs. If you want people other than scrubs to apply, there needs to be a fundamentally different way of thinking about the entire department. If QA is thought of as a viable career path, and a truly important part of game development, it won't be considered lower-tier, and your games will get better, because creative people will be thinking about how to improve your games and processes. At Valve, for instance, everyone has specialties, but everyone is a developer. Everyone plays the game all the time, and thus everyone is QA. That's not so bad, is it? How can we reach this level of integration in our own companies?

QUALITY TIME

» The first thing to do is to change the company culture and mind space surrounding QA. Invite QA leads to important meetings. Creative meetings! Make sure you, or your leads, speak of QA with the same respect you'd have for any other discipline. Make sure your entire team is speaking to everyone else on the team, and regularly.

Next, offer an appealing career path within QA. Certainly there will be generalists that check everything, and some general leads. At the same time, QA professionals that are interested in music should essentially be part of the sound team, working to develop an audio map to test against, with the power to implement changes. Those with a tech bent should be speaking regularly with the leads in that area to monitor frame rates, and suggest areas for reducing load. And so on and so forth.

Finally, don't lay them off when a project completes! If you want loyal employees, your company should be hiring for the long term. You should be recruiting QA professionals with a variety of skills that can be applied across the project, like you would in any other discipline. Unless your team is gigantic, you likely don't hire an artist who is only good at rigging. It's a specialty, sure, and you rely on them for that. But when it's time to do a bit of modeling, or mocap cleanup, they can be counted on. So too should it be for your QA staff. If you really need to ramp up and down our QA, use external QA groups managed by your permanent internal QA team. If you think of any members of your team as an expendable resource, they will not do their best work for you.

This is only the start of the discussion. QA is an incredibly important pillar of game development, and the industry as a whole does not treat it accordingly. If you want to see your next bonus, maybe it's time to change all that. ☹️

—Brandon Sheffield
twitter: @necrosotoy



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
e: gamedeveloper@halldata.com
www.gdmag.com/contactus

EDITORIAL

PUBLISHER

Simon Carless e: scarless@gdmag.com

EDITOR-IN-CHIEF

Brandon Sheffield e: bsheffield@gdmag.com

EDITOR

Patrick Miller e: pmiller@gdmag.com

MANAGER, PRODUCTION

Dan Mallory e: dmallory@gdmag.com

ART DIRECTOR

Joseph Mitch e: jmitch@gdmag.com

DESIGNER

Cliff Scorso e: cliff.scorso@ubm.com

CONTRIBUTING WRITERS

Tom Curtis
Serge Bernier
Keith O'Conor
Mike Fridley
Jelle van der Beek
Steve Theodore
Soren Johnson
Damian Kastbauer
Kim Pallister
Matthew Wasteland

ADVISORY BOARD

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

ADVERTISING SALES

GLOBAL SALES DIRECTOR

Aaron Murawski e: amurawski@ubm.com
t: 415.947.6227

MEDIA ACCOUNT MANAGER

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

GLOBAL ACCOUNT MANAGER, RECRUITMENT

Gina Gross e: ggross@ubm.com
t: 415.947.6241

GLOBAL ACCOUNT MANAGER, EDUCATION

Rafael Vallin e: rvallin@ubm.com
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER

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

REPRINTS

WRIGHT'S MEDIA

Jason Pampell e: 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
e: petercan@SMS-Inc.com
ubm.sms-inc.com





PLANETSIDE 2

Intel® GPA[†] Helps SOE Make PlanetSide 2* Faster, Faster.

INTEL® GPA HELPS PLANETSIDE 2* DEVELOPERS BOOST PERFORMANCE



Intel® Graphics Performance Analyzers (Intel® GPA) is a powerful graphics tool suite for analyzing and optimizing your games, media, and other graphics-intensive applications. With Intel® GPA, you can conduct in-depth analysis from the system level all the way down to individual elements, allowing you to maximize the performance of your applications.

Intel® GPA System Analyzer

Learn whether your game is CPU- or GPU-bound. Quickly analyze game performance and identify potential bottlenecks.

Intel® GPA Frame Analyzer

Optimize graphics performance through deep frame analysis of elements at the draw-call level.

Intel® GPA Platform Analyzer

Visualize performance of your application's tasks across the CPU and GPU.

"Using Intel® Graphics Performance Analyzers is like having a great sniper on our team to guard our optimization needs. GPA helps identify potential threats to PlanetSide 2's frame rate so we are better able to focus on our overall mission: making PlanetSide 2's epic gameplay shine on a massive scale."

- RYAN ELAM,
TECHNICAL DIRECTOR, PLANETSIDE 2
SONY ONLINE ENTERTAINMENT

Register at www.planetside2.com



DOWNLOAD INTEL® GRAPHICS PERFORMANCE ANALYZERS FOR FREE at www.intel.com/software/gpa





From fridge magnets to concept art

Highlights from iam8bit's Super Magnetic Game-O-Matic at GDC 2012

\\ GENERATING A NEW IP IS EASIER THAN YOU MIGHT THINK. AT LEAST, THAT'S THE TAKEAWAY FROM CREATIVE MARKETING THINK TANK IAM8BIT'S SUPER MAGNETIC GAME-O-MATIC, WHICH GAVE GDC 2012 ATTENDEES A CHANCE TO PUT TOGETHER THEIR PITCHES FOR NEW GAMES WITH OVERSIZED REFRIGERATOR WORD MAGNETS—AND GAVE THEM TO INDUSTRY ARTISTS FROM DOUBLE FINE PRODUCTIONS, INSOMNIAC GAMES, BACKBONE ENTERTAINMENT, AND OTHER DEVELOPERS. CHECK OUT SOME OF OUR FAVORITES.



TITLE: Super Donut Whales GENRE: Arcade Real-Time Strategy
ARTIST: Derek Brand (Double Fine Productions)



TITLE: A Toast with Robot Pickles GENRE: New 4D Prehistoric Mystery Experience
ARTIST: Kinman Chan (Independent)



TITLE: Tyler Perry's 3D Dino Ninja Revolution GENRE: Rhythm
ARTIST: Levi Ryken (Double Fine Productions)



TITLE: Meat Gravity Tale 4D Space Edition GENRE: Last Clicked Epic Shooter
ARTIST: Greg Broadmore (Weta Workshop)



TITLE: Prince of Jetpack Whales GENRE: Third-Person Adventure
ARTIST: Seth Forester (Backbone Entertainment)



TITLE: Mystery Sheriff Woman of Life GENRE: Social Sandbox MMO
ARTIST: Will Guy (Independent)

Buy a Vote, Make a Game

Magical Game Factory merges crowdfunding with game design process



\\ INDEPENDENT DEVELOPER WATERMELON CO. MADE A SPLASH IN 2010 WITH PIER SOLAR AND THE GREAT ARCHITECTS, A 16-BIT ROLE-PLAYING GAME MADE FOR THE SEGA GENESIS THIRTEEN YEARS AFTER THE CONSOLE WAS DISCONTINUED. NOW IT'S DISRUPTING THE BUSINESS ONCE AGAIN WITH A NEW CROWDFUNDING SYSTEM CALLED THE MAGICAL GAME FACTORY, WHICH LETS DEVOTED FANS INFLUENCE WHAT THE NEXT WATERMELON GAME WILL LOOK LIKE—AND PAY FOR THE PRIVILEGE. WE CAUGHT UP WITH WATERMELON'S TULIO GONÇALVES AND GWÉNAËL GODDE TO SEE HOW THE MAGICAL GAME FACTORY WAS WORKING OUT FOR THEM.

How does the Magical Game Factory system work?

Tulio Gonçalves: The Magical Game Factory [MGF for short] is WM's innovative way to crowd-source game development. Our system is tied to our e-commerce platform. Users can buy "gems" as a virtual currency that allows them to invest in a project. Once gems are invested it gives the investor access to all active polls and scalars that will define the game development. Your vote has the weight of the gems you invested, so more gems equals more decision power.

How much money are people donating?

T.G.: As of now, the system has been running for less than a month and we already have hundreds of investors. Some of them are pretty generous.

How have the first few rounds gone? Is there anything you'd like to change?

Gwénaël Godde: We got some interesting results so far but it is indeed an experiment, as the result is really dependent on the number of people investing and their individual motivation. So far everyone who participates seems very enthusiastic. One funny fact: Some people believed that the comments posted by our members were suspiciously "fake" because they were too positive. How great is that?

What kind of decisions are you exposing to the audience? Are you worried that your contributors will end up forcing you into a game you don't want to make?

TG: The decision range varies according to the project. The factory can run up to three projects simultaneously, and as an example, Project SF is already booted as an ARPG style, but Project Y was booted entirely empty, so the investors got to choose the platform, the genre/sub-genre, and now the setting and languages. We read every single comment and message, and that influences us on what the next poll will be. Still, WaterMelon keeps control of the game process, considering the fact that we only put up options that make sense for the poll. This way we can avoid choices that could be offensive to some people, for example.

Will contributors be receiving anything else in exchange for their money besides influence in the polls?

TG: Yes. To start, all gems invested become a discount when the game is finished, and if you

invested enough gems, they're automatically converted into a game. So one may think of the investment in a project as a pre-order with benefits. Additionally, for all investors there will be a special "Investor Edition" (we're working on that name) as an exclusive benefit for them.

For the first poll, the fans decided you'd make a beat-'em-up. How'd you decide which genres to include?

TG: For genres we decided to include most of the options available, without being too specific and then, there was also a sub-genre (also not very specific). It didn't surprise me that beat-em-up won the poll though, because it's a genre with thousands of fans worldwide but that has somehow been forgotten since the rise of 3D consoles.

How did you come up with the idea for the MGF?

GG: We can't make new old school games without involving the fans, because it's a niche market and making it profitable is very, very hard. We bring original content, and investment needed for original content is absolutely huge.

The idea started long time ago, back in 2006 with the "posterity" version of PIER SOLAR where people could pay a bit more to get their name in the credits. That kind of thing is common now, but I think we were early. Then, in 2010, we wrote the final guidelines for the Magical Game Factory. It's interesting that since then, crowdfunding has emerged as a big thing for game development, but we've been doing it since the beginning in a way that really involved the fans.

We have noticed that public voting tends to "blend" the choices available, thus making games less innovative than they could be, and voters can get frustrated when their preferred choice doesn't win a poll. Our polls are a little bit more complicated than they seem, since we want to make a great game and keep everyone happy.

Do you think other independent development studios will adopt a similar system?

GG: We are very open to any collaboration and widening our audience, though we did our process. In our past experience, we've found that the good ideas that we all hear about rarely come from their original inventors. But is the Magical Game Factory a good idea? We have yet to make it happen! 🎮

—PATRICK MILLER



INIS BREAKS NEW GROUND IN MOBILE GAMING WITH UNREAL ENGINE 3

Since 1997, Japanese game development studio iNiS has focused primarily on rhythm and music console games such as *Gitaroo Man*, *Elite Beat Agents*, the *Lips* series and *The Black Eyed Peas Experience*. But that last title for Ubisoft marked the company's first experience with Epic Games' Unreal Engine 3 technology. Now, iNiS is using UE3 to develop five new mobile projects, including *Infinity Blade Cross* (developed in cooperation with DeNA and ChAIR Entertainment, only available in Japan) and its most recent original project, *Eden to GREEEN*, a tower offense game developed for NVIDIA Tegra 3-powered devices.

Game designer Keiichi Yano, director and vice president at iNiS, said that *The Black Eyed Peas Experience* project started its life as a first-party project with Microsoft Studios when Kinect was in its prototype phase. It was Microsoft that suggested the developer use Unreal Engine 3 to leverage technology that would significantly shorten the time to get the final product to market.

The company's transition to mobile game development was a direct result of their involvement with UE3. While high-end smartphones and tablets were still very new to Japan, they were making an impressive impact on the market. That's when iNiS received a call from DeNA, the company behind the Morage platform.

"We were adding the final touches to *The Black Eyed Peas Experience* and DeNA approached us to work on a game they had just licensed, *Infinity Blade*," said Yano. "They wanted a studio with deep Unreal Engine 3 experience to handle the creation of a free-to-play version that would run on their social game platform. That game was *Infinity Blade Cross*, our first foray into high-end smartphone game development."

Yano said that UE3 allowed his team to rapidly prototype ideas and get them to a graphically polished state quickly.

The process worked so well with *Eden* that the studio now has five UE3 projects in development.

"Our in-house pipeline is now seriously geared towards making UE3 sing, and it was no different when creating *Eden to GREEEN*," said Yano. "In a short time, we were able to get the game running well on the latest NVIDIA Tegra devices, including smartphones and tablets.

"The combination of a strong tech base and a powerful chipset really allowed us to give *Eden* an ambience, an atmosphere that we just couldn't have achieved previously on mobile devices. We have fire, wind, dust and shadows from overhead clouds... a smorgasbord of effects. It's nice to be able to rely on proven tech and platforms to make *Eden* come alive."

iNiS plans to utilize the cross-platform capabilities of UE3 with *Eden* by releasing the game on Android, iOS, Windows, Mac OS X and Flash. From a development standpoint, Yano is extremely pleased to be able to create a base that can be quickly ported to other platforms.

The studio makes extensive use of UE3 tools such as Matinee, which Yano values as an ideal system to create cinematics that blend seamlessly in and out of gameplay. "It's very cool that we can impact the cinematics in a way that emotionally connects with the player," said Yano. "Cascade is another favorite, and we make heavy use of it to bring ambience to the world of *Eden*."

Additionally, when it comes to troubleshooting and researching, the team found the Unreal Developer Network (UDN) to be an indispensable tool. iNiS was able to work with other developers, as well as Epic, to solve development challenges and enhance productivity. Yano explains it was particularly helpful when developing for Android, "It really helped to be able to share ideas with others who had already gained some knowledge and experience deploying UE3 games on the platform.

With *Eden to GREEEN*, Yano's goal was to create a new

type of experience in a strategy game. While the game has elements of tower defense, Japanese role-playing games, turn-based strategy and Pixar movies all rolled into one, the end result is something he calls "tower offense." In addition to his creative aims, Yano also sought to create this game for the burgeoning free-to-play gaming space, opening up the game experience to a wide audience across many platforms.

With its beautiful stylized graphics and vibrant backdrop, *Eden to GREEEN* looks unlike any other game currently available. With its humorous storyline of invading alien machines trying to destroy an Eden defined by 18 unit types of flowers, plants and trees, *Eden to GREEEN* offers fast-paced multiplayer gameplay and a never-ending challenge of new maps. Powered by Unreal Engine 3, iNiS's new game is helping the game developer forge its way to the top of the mobile development community, much as it did its rhythm-based console games.

WWW.UNREAL.COM



Canadian-born Mark Rein is vice president and co-founder of Epic Games based in Cary, NC. Epic's Unreal Engine 3 has won Game Developer magazine's Best Engine Front Line Award eight times, including entry into the Hall of Fame. UE3 has won four consecutive Develop Industry Excellence Awards. Epic

is the creator of the mega-hit "Unreal" series of games and the blockbuster "Gears of War" franchise. Follow @MarkRein and @UnrealEngine on Twitter.

UPCOMING EPIC ATTENDED EVENTS

Gadget Show Live
Birmingham, UK
April 10-15, 2012

East Coast Game Conference
Raleigh, NC
April 25 & 26, 2012

E3 Expo
Los Angeles, CA
June 5-7, 2012



Please email licensing@epicgames.com for appointments

PATRICK MILLER

If we had to pick a theme for our eleventh annual *Game Developer* magazine Salary Survey, it would be "cautious optimism." Compared to the general industry-wide downturn we saw in 2009 (and the reasonable recovery in 2010), 2011 seems to have shaken off the boom-bust cycle in favor of small-but-steady growth.

In case you're new to the survey, we ask thousands of *Game Developer* and Gamasutra readers to tell us what they made in the last year, asking a slew of related questions along the way. From this group we learned that the average salary across the entire game industry is \$81,192, hovering near the same level as 2010's \$80,817 reported average. What that number doesn't tell you is that the industry was significantly more stable this year than it has been in the past several. 66 percent of survey respondents made more money in 2011 than they did in 2010, compared to 56 percent from 2010 to 2009. Only 13 percent of respondents were laid off in 2011, compared to 14 percent in 2010 and 19 percent in 2009, and the folks that were laid off were 6 percent more likely to find a new job elsewhere in the games industry (58 percent, up from 52 percent in 2011).

Having a little more money and stability in turn made developers feel more optimistic about their careers as well as the industry as a whole. 65 percent of developers said they felt "satisfied" or "extremely satisfied" with their potential career path (up 4 percent from 2010), 34 percent believed that there were more jobs in the industry than the year before (up 5 percent), and 54 percent felt that there were more opportunities for game developers than before (up 7 percent).

Nevertheless, there's still plenty of change going on. In the comments section of the survey, developers weighed in (both positively and negatively) on the ever-growing business of mobile and social games, the increasing viability of free-to-play business models in the U.S. market, and the maturing indie space. Indies in particular made a big step in 2011: Individual indie developers reported an average \$23,549 in primary compensation, more than double 2010's \$11,379, while members of independent developer teams made an average of \$38,239, up \$11,459 from 2010's \$26,780. Those numbers might not look great to a senior developer accustomed to triple-A salaries, but if you're tired of working on sequels or just looking to break in the industry, there are more avenues available for you than ever before.

Perhaps the best takeaway from this year's survey also came from the comments: "People are beginning to realize that games are one of the few legal ways to escape reality. Since people would rather live in the clouds than face reality, we can expect more growth in the industry."

SALARY SURVEY ELEVENTH ANNUAL

ELEV
THAN
gd
N N
U A L

programmers

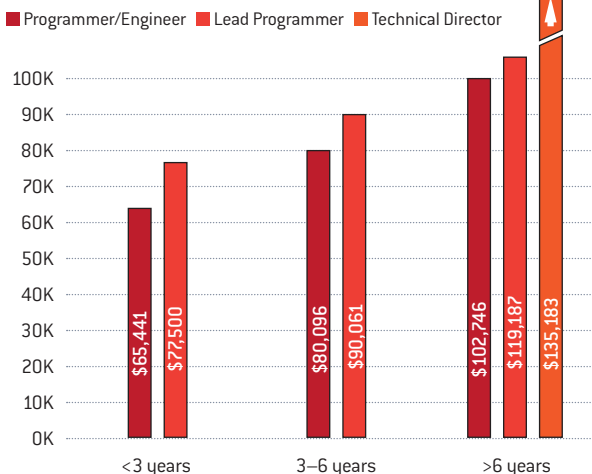
AVERAGE SALARY
\$92,962

PROGRAMMERS ARE BE IN HIGH DEMAND, AND THEIR SALARIES HAVE continued to rise as well: Their average salary is up about \$7,200. What's more, programmer salaries have increased across the experience spectrum, with newer workers reporting a whopping \$10,700 average increase, compared to a \$7,700 increase for people with 3–6 years, and a \$5,800 increase for people with over six years. Considering we had more programmers respond to the survey than we did last year, it seems that the industry simply can't hire talented programmers fast enough.

Canadian developers didn't see the same boom, though—their \$74,970 (USD) average survey was only up about \$500. European programmers reported an average of \$46,801 (USD), down about \$1,400 from 2010.

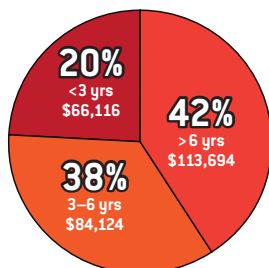
While women were underrepresented even more than usual (2.9 percent in 2011, down from 4 percent in 2010), they did report an average increase of about \$8,800, compared their male counterparts' \$7,000 increase. However, the wage gap is still alive and well; the average female programmer's salary (\$83,333) is nearly \$10,000 lower than the average male programmer's (\$93,263).

Programmer salaries per years experience and position



ALL PROGRAMMERS AND ENGINEERS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **83%**

Average additional income: **\$21,009**

Type of additional compensation received

Annual bonus	46%
Pension/Employer contribution to Retirement plan	38%
Profit sharing	16%
Project/title bonus	20%
Royalties	8%
Stock options/equity	35%

GENDER STATS FOR PROGRAMMERS

Percent receiving benefits: **94%**

Gender	Percent Represented	Average Salary
Male	97%	\$93,263
Female	3%	\$83,333

Type of benefits received	Percent
Medical	84%
Dental	80%
401K/Retirement	72%

artists and animators

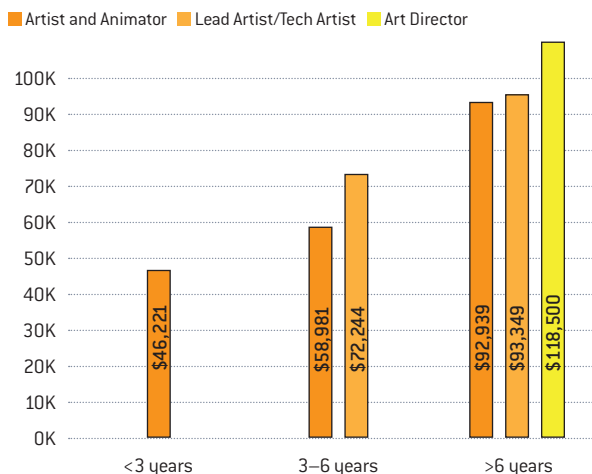
AVERAGE SALARY
\$75,780

ARTISTS' AND ANIMATORS' AVERAGE SALARIES INCREASED \$4,400 over the previous average of \$71,354. Artists with less than three years of experience saw an average increase of \$3,500, those with 3–6 years saw an increase of \$1,300, and the over-six-years crowd saw an average bump of \$6,100. It's worth pointing out that the largest gains went to art directors and lead/technical artists, which bodes well for the industry veterans and less so for the younger artists and animators.

Artists and animators in Canada received an average salary of \$66,651 (USD), up about \$3,300. Artists in Europe didn't fare so well, however. Their \$35,887 (USD) average salary fell about \$5,000 from 2010.

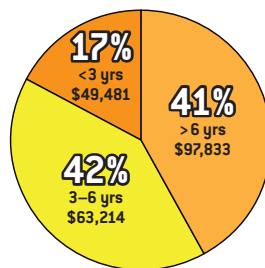
Women comprised 13 percent of our surveyed artist pool, up 2 percent, though their average salary (\$52,875) actually decreased about \$6,800. Men in art and animation, on the other hand, made \$79,124 on average, which is up about \$6,200 from 2010.

Artist and Animator salaries per years experience and position



ALL ARTISTS AND ANIMATORS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **83%**

Average additional income: **\$16,163**

Type of additional compensation received

Annual bonus	40%
Pension/Employer contribution to Retirement plan	36%
Profit sharing	11%
Project/title bonus	30%
Royalties	14%
Stock options/equity	28%

GENDER STATS FOR ARTISTS

Percent receiving benefits: **95%**

Gender	Percent Represented	Average Salary
Male	87%	\$79,124
Female	13%	\$52,875

Type of benefits received	Percent
Medical	80%
Dental	76%
401K/Retirement	65%

game designers

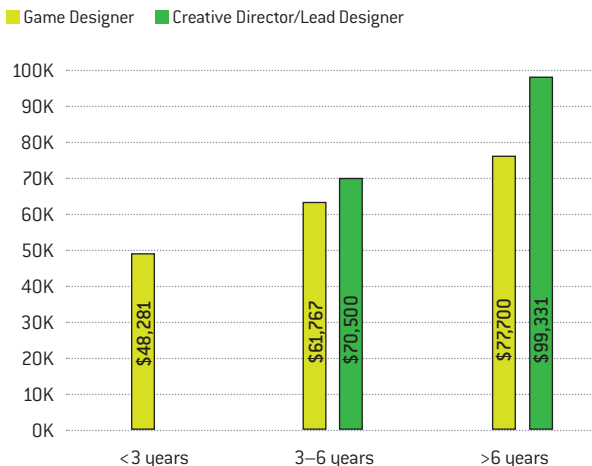
AVERAGE SALARY
\$73,386

GAME DESIGNERS, WRITERS, AND CREATIVE DIRECTORS WERE PAID \$73,386 on average in 2011, up \$3,100. Game designers with less than three years of experience received the biggest boost (\$3,500), while the 3–6-year crowd made \$2,000 more, and the over-six-years designers only made an extra \$500 or so. Creative directors and lead designers with 3–6 years under their belts made \$2,600 less compared to 2010, while the over-six-years group pulled in an extra \$3,700.

Canadian game designers averaged \$60,240 (USD), up about \$1,950 from 2010, while European game designers averaged \$38,281 (USD), which is down about \$3,000.

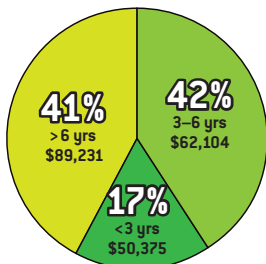
Female designers made up 10.9 percent percent of our designer responses, compared to 7 percent from 2010, and their average salary of \$67,000 is up \$2,850 from 2010. Male designers, on the other hand, made an average of \$74,180 in 2011—up \$3,500 from 2010.

Game Designer salaries per years experience and position



ALL GAME DESIGNERS

YEARS EXPERIENCE IN THE INDUSTRY



GENDER STATS FOR DESIGNERS

Gender	Percent Represented	Average Salary
Male	89%	\$74,180
Female	11%	\$62,000

Percent receiving additional income: **79%**

Average additional income: **\$15,216**

Type of additional compensation received

Annual bonus	35%
Pension/Employer contribution to Retirement plan	34%
Profit sharing	13%
Project/title bonus	23%
Royalties	10%
Stock options/equity	31%

Percent receiving benefits: **96%**

Type of benefits received

Medical	84%
Dental	80%
401K/Retirement	70%

producers

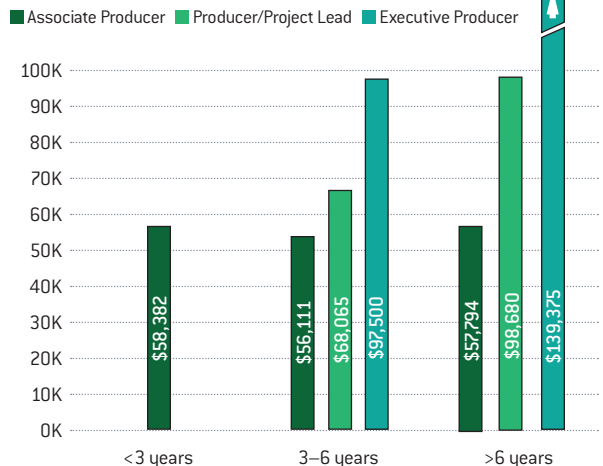
AVERAGE SALARY
\$85,687

PRODUCERS SAW A \$2,850 SALARY CUT COMPARED TO 2010'S AVERAGE salary of \$88,554, though they still draw a higher salary than all other departments except programming and business. Most of these cuts were for producers with 3–6 years of experience, whose average salaries fell about \$4,900 from 2010, and producers with over six years, whose salaries were cut by \$3,300 on average. Producers with fewer than three years of experience, on the other hand, made \$3,600 more in 2011.

Canadian game producers averaged \$71,500 in 2011, down \$1,000 from 2010. European producers' salaries increased \$3,500 to \$56,346. Interestingly, producers in Europe are paid more than any other department of game development in the region.

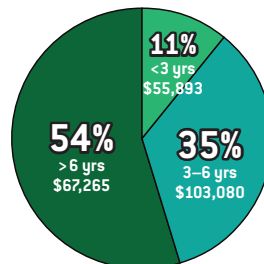
Women are well-represented in the production department this year (16 percent), down 1 percent from 2010. Interestingly enough, men absorbed most of the salary drop; their average salary dropped from \$90,744 to \$87,119, while women producer's salaries actually rose slightly from \$77,870 to \$78,354.

Producer salaries per years experience and position



ALL PRODUCERS

YEARS EXPERIENCE IN THE INDUSTRY



GENDER STATS FOR PRODUCERS

Gender	Percent Represented	Average Salary
Male	84%	\$87,119
Female	16%	\$78,354

Percent receiving additional income: **85%**

Average additional income: **\$19,050**

Type of additional compensation received

Annual bonus	49%
Pension/Employer contribution to Retirement plan	37%
Profit sharing	13%
Project/title bonus	23%
Royalties	6%
Stock options/equity	37%

Percent receiving benefits: **98%**

Type of benefits received

Medical	61%
Dental	60%
401K/Retirement	54%

audio professionals

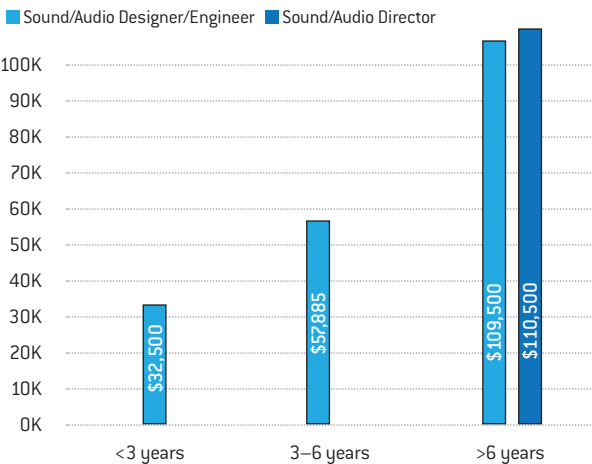
AVERAGE SALARY
\$83,182

THE AVERAGE U.S.-BASED AUDIO PROFESSIONAL'S REPORTED SALARY in 2011 was \$15,000 higher than 2010's \$68,088. However, we don't get nearly as many survey responses from audio professionals as we do from any other discipline, making it hard to draw meaningful conclusions. We did get about 30 percent more responses from audio professionals than we did in 2010, though.

Of that \$15,000 average increase, most of the gains went to the veterans: sound/audio directors with over six years of experience received \$5,500 more than they did in 2010. Audio professionals working in contract roles also made an extra \$3,200 in 2010. Salaried audio workers were the least likely to receive extra compensation for their work out of any discipline, though the \$9,875 they received is up \$2,200 from 2010.

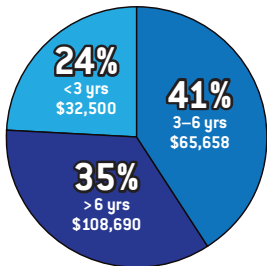
Canada-based audio professionals received an average salary of \$67,955 in 2011, down \$600 from 2010. Unfortunately, we didn't collect enough responses from European audio workers to make any significant conclusions.

Audio Developer salaries per years experience and position



ALL AUDIO DEVELOPERS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **74%**

Average additional income: **\$9,875**

Type of additional compensation received

Annual bonus	40%
Pension/Employer contribution to Retirement plan	47%
Profit sharing	12%
Project/title bonus	23%
Royalties	9%
Stock options/equity	28%

GENDER STATS FOR AUDIO DEVELOPERS

Gender	Percent Represented	Average Salary
Male	93%	\$83,963
Female	7%	\$72,500

Percent receiving benefits: **90%**

Type of benefits received	Percent
Medical	79%
Dental	79%
401K/Retirement	63%

qa testers

AVERAGE SALARY
\$47,910

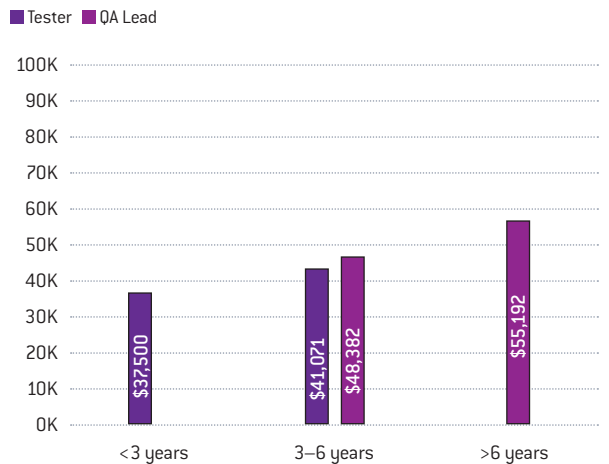
QUALITY ASSURANCE PROFESSIONALS (TESTERS AND QA LEADS) are the lowest-paid profession in the game industry for yet another year—and in 2011, they got paid an average of \$1,100 less than they did in 2010.

That average salary actually reflects the higher end of the QA spectrum, because many entry-level QA employees are hired on a contract basis, with salaried positions mostly given to QA leads. QA contractors made an average of \$27,065 in 2011, up about \$4,150 from 2010.

Interestingly enough, average salaries for testers with less than three years of experience rose about \$6,200 from 2010, and QA leads with 3–6 years of experience saw a \$3,700 boost. The salary loss was mostly felt by QA leads with over six years of experience, who took a hit of about \$7,200 in 2011.

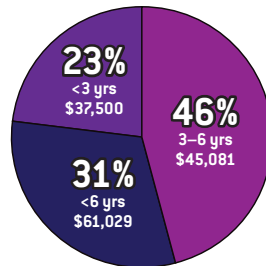
Canadian QA professionals made an average of \$43,125 (USD), up \$5,100 from 2010. European QA workers made \$32,500 (USD), which was about \$6,750 less from 2010.

QA Tester salaries per years experience and position



ALL QA TESTERS

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **77%**

Average additional income: **\$12,640**

Type of additional compensation Received

Annual bonus	49%
Pension/Employer contribution to Retirement plan	33%
Profit sharing	7%
Project/title bonus	16%
Royalties	5%
Stock options/equity	30%

GENDER STATS FOR QA TESTERS

Gender	Percent Represented	Average Salary
Male	87%	\$49,196
Female	13%	\$39,375

Percent receiving benefits: **95%**

Type of benefits received	Percent
Medical	68%
Dental	70%
401K/Retirement	64%

business and legal people

AVERAGE SALARY
\$102,160

THE "BUSINESS AND LEGAL PEOPLE" CATEGORY INCLUDES CHIEF executives and executive managers, community managers, marketing, legal, human resources, IT, content acquisition and licensing, and general administration staff.

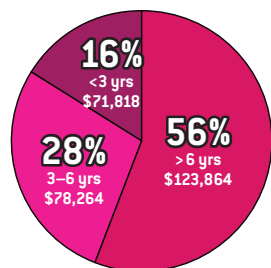
Business professionals received the highest average salary in the industry (\$102,160), as well as the most additional non-salary income (\$24,874). However, both those numbers are actually down from 2010 by \$3,300 and \$4,000, respectively.

Younger people on the business side made an average of \$71,818 in 2011, \$14,000 up from \$57,778 in 2010. Business people with 3–6 years of experience averaged \$78,269 in 2011, down \$3,200 from \$81,528 in 2010, while those with over six years in the industry made \$123,864 in 2011, down almost \$8,000 from 131,786 in 2010.

Women were relatively well-represented in the business side of the games industry, reaching 17.6 percent in 2011, which is 7.6 percent higher than average across the entire industry this year, and 3.6 percent higher than in 2010.

ALL BUSINESS AND LEGAL PEOPLE

YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **82%**

Average additional income: **\$24,874**

Type of additional compensation Received

Annual bonus	46%
Pension/Employer contribution to Retirement plan	30%
Profit sharing	19%
Project/title bonus	15%
Royalties	11%
Stock options/equity	44%

GENDER STATS FOR BUSINESSPEOPLE

Gender	Percent Represented	Average Salary	Type of benefits received
Male	82%	\$108,402	Medical 82%
Female	18%	\$73,534	Dental 78%
			401K/Retirement 62%

LAYOFFS

JOBS IN THE GAME INDUSTRY APPEAR TO BE GETTING SLIGHTLY MORE stable. Of 3,100 respondents, 13 percent had been laid off in 2011, compared to 14 percent in 2010 and 19 percent in 2009.

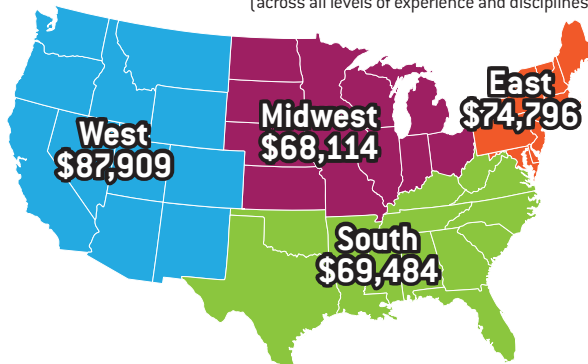
From those people who were laid off, 58 percent found new employment in the games industry, 19 percent went into contracting or consulting, 10 percent founded a new company, 13 percent went into independent games development, and 13 percent haven't found new game development work. (Note that for this survey question, multiple responses were allowed.)

A significant amount of respondents reported being laid off and rehired by the same company, either as a contractor, or as a salaried employee with a different job title (but the same responsibilities).

Fewer laid-off developers opted to start their business, join an independent studio, or go into contract work this year. This could be a sign of a slowing bubble in the social and mobile sectors, which were giving away massive amounts of cash in recent years.

AVERAGE SALARY BY U.S. REGION

(across all levels of experience and disciplines)



TOP 10 STATES WITH HIGHEST AVERAGE SALARIES

(across all levels of experience, excluding states with low sample size)

	AVERAGE SALARY	PERCENT WHO OWN HOMES	AVG. SALARY OF HOMEOWNERS
1 California	\$93,696	32%	\$118,513
2 Washington	\$89,674	49%	\$108,479
3 North Carolina	\$82,500	61%	\$93,382
4 Texas	\$79,017	49%	\$95,761
5 Massachusetts	\$78,567	44%	\$94,688
6 Maryland	\$75,521	44%	\$94,688
7 Utah	\$74,722	70%	\$78,816
8 Illinois	\$71,319	47%	\$84,167
9 New York	\$69,697	29%	\$85,395
10 Florida	\$67,136	33%	\$90,000

AVERAGE SALARY BY U.S. REGION BY DISCIPLINE

	EAST	MIDWEST	SOUTH	WEST
Programmer	\$84,167	\$73,397	\$82,740	\$101,387
Art and Animation	\$64,235	\$68,676	\$71,818	\$81,229
Game Design	\$69,575	\$61,818	\$52,616	\$82,204
Production	\$77,813	\$87,045	\$68,889	\$90,395
Audio	\$60,625	\$55,00	\$68,333	\$97,885
QA	\$47,500	\$44,167	\$39,643	\$49,545
Business	\$97,870	\$87,500	\$83,750	\$108,056

AVERAGE SALARY FOR HOMEOWNERS VS. NON-HOMEOWNERS BY U.S. REGION

	EAST	MIDWEST	SOUTH	WEST
Homeowners	\$93,662	\$81,378	\$89,415	\$111,842
Non-Homeowners	\$66,602	\$58,777	\$58,223	\$79,148

AVERAGE SALARIES IN THE U.S., CANADA, AND EUROPE

(across all levels of experience, by discipline, given in USD)

	U.S.	CANADA*	EUROPE**
Programmer	\$84,124	\$74,970	\$46,801
Art and Animation	\$63,214	\$66,651	\$35,887
Game Design	\$62,104	\$60,240	\$38,281
Production	\$67,265	\$71,500	\$56,346
Audio	\$65,658	\$67,955	\$25,500
QA	\$45,081	\$43,125	\$32,500
Business	\$79,269	\$100,938	\$47,222

*Most Canadian respondents were from British Columbia, Quebec, and Ontario.

**Most European respondents were from the United Kingdom (24%), Germany (12%), France (11%), Spain (9%), Poland (10%), Spain (6%), Sweden (5%).

AVERAGE SALARY BY EDUCATION LEVEL AND DISCIPLINE

(across all levels of experience)

	PROGRAMMING	ART	DESIGN	PRODUCTION	AUDIO	QA	BUSINESS
High school/GED	—	—	\$79,500	—	—	—	—
Some College	\$104,907	\$95,379	\$77,500	\$94,643	—	\$45,000	\$113,250
Associates Degree	\$93,676	\$79,737	\$69,583	—	—	—	—
Bachelors Degree	\$88,649	\$71,680	\$73,703	\$81,685	\$81,364	\$49,667	\$88,684
Some Graduate	\$105,109	\$71,667	\$59,423	\$91,667	—	—	\$106,731
Masters Degree	\$90,291	\$59,868	\$68,438	\$90,870	—	—	\$121,288
Some Doctoral	\$93,864	—	\$67,500	\$107,500	—	—	—
Doctoral Degree	—	—	—	—	—	—	—

METHODOLOGY

NOW IN ITS ELEVENTH YEAR, the Game Developer Salary Survey was conducted in February 2012 for the fiscal year January 1, 2011 through December 31, 2011 with the assistance of Audience Insights. Email invitations were sent to Game Developer subscribers, Game Developers Conference attendees, and Gamasutra.com members asking them to participate in the survey.

We gathered 4,132 responses from developers worldwide but not all who participated in the survey provided enough compensation information to be included in the final report. We also excluded salaries of less than \$10,000 and the salaries of students and educators. The small number of reported salaries greater than \$202,500 were included to prevent their high numbers from unnaturally skewing the averages. We also excluded records that were missing key demographic and classification numbers.

The survey primarily includes U.S. compensation but consolidated figures from Canada and Europe were included separately. The usable sample reflected among salaried employees in the U.S. was 1,742, for Canada 403, and for Europe 339; and 524 for indies and independent contractors who provided compensation information worldwide.

The sample represented in our salary survey can be projected to the U.S. game developer community with a margin of error of plus or minus 2.4% at a 95% confidence level. The margin of error for salaried employees in Canada is plus or minus 5%, and is 5.4% for Europe.

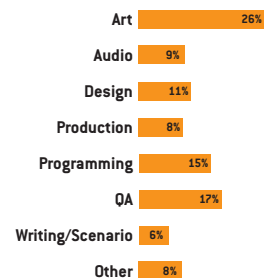
THE INDIE REPORT

THIS IS THE THIRD YEAR WE'VE collected data for our indie report, where we survey individual independent developers, independent teams, and individual contractors for their perspective on the industry. Out of those three groups, independent contractors made the most, though both individual indies and members of indie teams pulled in significantly more in 2011 than they did in 2010.

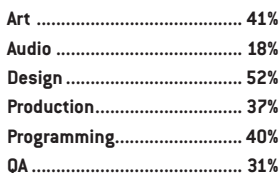
Independent contractors averaged \$56,282 in 2011 (up \$800 from 2010), individual independent developers averaged \$23,549 (up from \$11,379 in 2010), and members of independent developer teams averaged \$38,239 (up from \$26,780 in 2010). As the indie game community continues to mature and grow financially, it also seems to be consolidating somewhat. Compared to 2010, more independent developers are working in teams rather than going solo.

Indie games made a bit more money in 2011, too. 48% of independent developers made less than \$500 from the sale of their game, down from 55% in 2010. 16% of independent developers made over \$60,000 from the sale of their game in 2011, compared to 8% in 2010. Meanwhile, non-game revenue streams (non-game DLC/additional content, sponsorship or ad opportunities, and awards/grants), remained relatively hard to obtain—79% didn't receive any additional income whatsoever (down 2%

CONTRACTORS BY JOB FUNCTION



INDIES BY JOB FUNCTION



from 2010). The developers that did cash in through non-game revenue streams generally didn't make a whole lot, either; 44% made under \$1,000, compared to 35% from 2010. In general, it appears that the developers who are good at designing games to take advantage of non-game revenue streams are able to pull in a decent amount; in both 2010

and 2011, 40% of developers with these sources of income were able to make over \$5,000.

JOB FUNCTIONS

While we survey both indie game developers and contractors for their job function in the game industry, we structure the survey differently to adjust for the difference between the two sectors. Developing a game in a small independent team means most developers don't have completely specialized roles—usually people are wearing multiple hats, so asking an independent games developer to only report one discipline wouldn't be completely accurate. On the other hand, that's not the case for most contract game developers. As such, the indie chart should be read as "what percentage of independent developers do at least this job function," rather than "how many independent developers do this job exclusively."

More and more indie developers are finding themselves in roles involving design (59%, up 7%), programming (53%, up 13%), QA (43%, up 12%), and production (47%, up 10%), while art and audio have declined slightly (40% and 17%, each down 1% from 2010). Considering our number of independent developer responses overall were roughly equal, this means independent developers are wearing more hats than ever before—a good indie team member is someone who can code, test, design, produce art, and manage a production schedule.

LITERALS

In order to hear what developers are saying about the industry right now, we allowed space at the end of our survey for direct comments. Here are some of the more notable responses.

THE BAD

“The attitude toward work life balance is absolutely terrible. It is not an honor to work in games. Engineers are treated like garbage, especially when compared to the treatment they can easily get outside of games.”

“The desire for games to be fun, creative, and unique is slowly being diminished by the never-ending need for money, money which is best acquired by making derivative boring titles often seen in the social games market. The game industry is going the way of the movie industry with constant remakes and prequels and no new innovation besides independents who often give themselves too much credit.”

joined a company where my gender isn't used against me as an argument to dismiss me, and it's been a really empowering experience. Also, a lot of companies are closing, many are hiring. There's always a job open somewhere. But we need to stop the firing of employees once a game has shipped as a viable 'saving the company money' measure, and we need to find ways to welcome students within our ranks better, as opposed to always ask for only experienced personnel, thus not giving newcomers a chance to learn.”

“I'm still surprised by how much sexism/racism the industry exhibits, and by how difficult it is to change perspectives on it. While the industry as a whole

“It's been a terrible year in the U.K. There have been hardly any new design positions, and so many studios have closed.”

“Audio designers in the game industry are often treated as a disposable resource, and the opportunities for us, both inside and outside the game industry, are incredibly scarce, as sound is very much misunderstood and undervalued as a craft. There are bright glimmers, but we have to really band together, speak up, and make a lot of noise about examples of great sound design to help continue to validate our work. Great games where audio plays a prominent role like *BASTION* and *PORTAL 2* really help us to move forward.”

“The outsourcing of art jobs has made this a less than desirable position for starting artists. Why spend thousands of dollars on an art degree, and countless hours perfecting your skills, only to come into a studio and do lackey clean up on work received from the outsourcers? A lot of talented artists don't even really create art anymore, just mundane clean-up tasks.”

THE GOOD

“I think the advent of crowd-funding and self publishing is going to see a huge shift in the coming years in terms of what games get made and who makes them. Indie game studios once again possess the tools and funding necessary to retake the game development industry much like they did in the '80s and early 1990's.”

“We're at a big turning point. The digital distribution model, along with the explosion of mobile gaming, is ushering in a new age of smaller studios and quicker development

cycles. The age of triple A, 2–3 year games is coming to an end.”

“We're in a time of great change, not only in business models and distribution platforms, but also in ethics. I'm optimistic for the future—more people than ever are playing games and when this recession is over, we're going to see some incredible revenue across all aspects of the industry.”

“It seems that the industry is on the precipice of a creative revolution as SDKs and self-publishing options are getting more accessible, creative direction will come less from the publisher, who relies more on proven methods, and more from wild-eyed developers who would rather experiment than replicate.”

“Developers that have been able to adapt to the rise of social and mobile gaming have done well despite an overall economic decline in the world. Mobile and social gaming is exciting because it allows a developer to focus on simple, tight, and polished gameplay. In that regard I feel the recent trend is somewhat of a return to the 'golden age of games'.”

DON'T FEAR SOCIAL GAMES

“There is a lot to learn about game design and how to appeal to a wide audience. Think about the average time until someone who has never played a game before starts having fun in your game—social games do this better than most traditional games. This is an incredible and exciting thing. If some of the early companies in social games were unscrupulous, don't let this keep you from learning exciting lessons from social gaming and let this improve your own designs.”

“Video games!!!!!!!” 🎮



“The longer I work in the industry, the less I can relate to gamers. The vocal minority is more annoying than ever.”

“This is an industry I've at times wanted to leave. I've often come very close to it. Because, plain and simple, it's not easy being a woman in this industry. Thankfully I've recently

is slowly improving, I frequently find myself trying to explain to coworkers why certain content—however hilarious they find it—might offend certain groups of people. There still seems to be a “boys' club” atmosphere in the office sometimes and many women are put in the unenviable and unfair position of political correctness enforcer.”



MAGIC PIXEL GAMES

We craft original games.

We love what we do and we want your help.

WE'RE HIRING!



ENGINEERS, ARTISTS, DESIGNERS

VISIT US AT WWW.MAGICPIXELGAMES.COM

©2012 MAGIC PIXEL GAMES, ALL RIGHTS RESERVED

FIRE BLOOD EXPLOSIONS

Prototype 2's over-the-top effects tech

KEITH O'CONNOR

One hallmark of the PROTOTYPE universe is over-the-top open-world mayhem. We rely heavily on large amounts of particle effects to create chaos, filling the environment with fire, blood, explosions, and weapon impact effects. Sgt. James Heller (the main character) can go just about anywhere in the environment. He can run up the side of a building, glide across rooftops, or even fly across the city in a hijacked helicopter. Because of this we need an effects system that scales to support the hundreds of complex effects and thousands of particles that could be visible at any one time. Here's how we built upon the effects system developed at Radical for SCARFACE and HULK: ULTIMATE DESTRUCTION by improving and adding features that would allow us to push the effects to the level we needed for PROTOTYPE and PROTOTYPE 2.

FIRE BLOOD EXPLOSIONS



SIMULATING AND AUTHORIZING PARTICLES

/// Our particle systems are composed entirely of a component-based feature set. A feature describes a single aspect of how each particle behaves—like changing position according to gravity or some other force, spinning around a pivot point, animating UVs, changing size over time, and so on. The effects artist can choose any set of features to make a particular particle system, and each chosen feature exposes a set of associated attributes (such as velocity, weight or color) that she can tweak and animate. This is all done in Maya with the standard set of animation tools, using the same simulation code as the runtime compiled into a Maya plug-in to make sure Maya and the game both behave consistently.

Once the artist is satisfied with the look and behavior of a particle system in Maya, it is exported as an effect that can be loaded in the game. This effect is then scripted for gameplay using our in-game editor, “The Gym,” a complex state machine editor that allows designers to control every aspect of the game (see our GDC 2006 presentation for more details, Reference 1). When scripting an effect to play in a particular situation, the effects artist has access to an additional set of controls: biases and overrides. For each attribute that was added as part of a

feature, the effects artist can choose to bias (multiply) the animated value, or to override it completely. This allows a single loaded effect to be used in a variety of situations. For example, the artist can take a standard smoke effect and make small, light, fast-moving smoke or large, dense, black hanging smoke, just by biasing and overriding attributes such as emission rate, color, and velocity (see Figure 1 for an example).

Artists can use The Gym to tailor each instance of that effect to match its use in-game instead of authoring and loading many similar versions of the same effect or using an identical generic effect in multiple situations. This reduces memory usage and improves the artist's workflow, allowing them to tune the effects live with in-game lighting and animations. The biases and overrides are also a major part of our continuous level-of-detail system, which we'll describe later in this article.

Each particle system's attributes are stored as separate tightly packed arrays, such as the positions of every particle, then the lifetimes, then the velocities, and so on. This data-oriented design ensures that the data is accessed in a cache-efficient manner when it comes to updating the simulation state every frame, which has a huge impact on CPU performance when doing particle simulation. This way, we take up only a small percentage of the CPU's time to simulate

thousands of particles with complex behaviors. It also makes implementing an asynchronous SPU on PS3 relatively straightforward, as updating each feature means only the necessary attribute arrays for that feature need to be DMA-ed up, without any extraneous data.

Having the particles' positions separated has other performance benefits as well, such as allowing for fast, cache-efficient camera-relative sorting for correct alpha blended rendering. It also enables other features, such as particles that emit other particles by using the position output attribute array of the simulation update as an input to another system's particle generation process.

REDUCING MEMORY USAGE AND FRAGMENTATION

/// Having many short-lived particle effects going off all the time (during intense combat situations, for example) can start to fragment your available memory. Fragmentation happens when many small pieces of memory are allocated and freed in essentially random order, leading to a “Swiss cheese” effect that limits the amount of contiguous free memory. In other words, the total amount of free memory in the heap might be enough for an effect, but that memory could be scattered around the heap in chunks that are too small to be actually usable. (For an introduction

to fragmentation and memory allocators, check out Steven Tovey's great #AltDevBlogADay article, Reference 2). Even though we use a separate heap for particle allocations to localize fragmentation, it is still a problem. Fortunately, we have a few tricks to limit fragmentation—and handle it when it becomes an issue.

Whenever possible, we use static segmented memory pools (allocated at start-up) to avoid both fragmentation and the cost of dynamic allocations. The segments are sized to match the structures most commonly used during particle system allocations. Only once these pools are full is it necessary to perform dynamic allocations, which can happen during particularly heavy combat moments or other situations where many particle effects are being played at once.

Our effects system makes multiple memory allocations when a single particle system is being created. If any of these fail (because of fragmentation, or because the heap is just full), it means the effect cannot be created. Instead of half-creating the effect and trying to free any allocations already made (possibly fragmenting the heap further), we perform a single large allocation out of the effects heap. If this succeeds, we go ahead and use that memory for all the allocations. If it fails, we don't even attempt to initialize the effect, and it simply doesn't get played. This is obviously undesirable from the player's point of view, since an exploding car looks really strange when no explosion effect is played, so this is a last resort. Instead, we try to ensure that the heap never gets full or excessively fragmented in the first place.

Toward this end, one thing we do is partition the effect into "stores," based loosely on the class of effect. We have stores for explosions, ambient effects, bullet squibs, and a number of other effect types. By segregating effects like this, we can limit the number of effects of a particular type that are in existence at any one time. This way, our effects heap doesn't fill up with hundreds of blood-spatter effects, for example, thus denying memory to any other type of effect. The stores are structured as queues; when a store is full and a new effect is played, the oldest effect in that store gets evicted and moved to the "graveyard" store (where old effects go to die). Their emission rate is set to zero so no new particles can be emitted, and they are given a certain amount of time (typically only a few seconds) to fade out and die, whereupon they are deleted.

Having effects partitioned into stores also allows us to perform other optimizations based on the type of effect. For example, we can assume that any effect placed in the "squib" store is a small, short-lived effect like sparks or a puff of smoke. Therefore, when one of these effects is played at a position that isn't in the camera frustum, or is further away than a certain distance, we simply don't play the effect at all, and nobody even notices. Another example is fading away particles from effects in the "explosion" store when they get too close to the camera, as they will likely block the view of the action, and also be

very costly to render. When the player is surrounded by legions of enemy soldiers, tanks, and helicopters all trying to get a piece of him, these optimizations can lead to significant savings.

We also cut our memory usage by instancing effects. In our open-world setting, the same effect is often played in multiple places—steam from manhole covers and smoke from burning buildings, for example. In these cases, we only allocate and simulate one individual "parent" effect, and we then place a "clone" of this parent wherever that effect is played. Since only the parent needs to generate and simulate particles, and each clone only needs a small amount of bookkeeping data, we can populate the world with a large number of clones with a negligible impact on memory and CPU usage. To combat visual repetition, each clone can be rotated or tinted to make it look slightly different.

MANAGING VERTEX BUFFER MEMORY DEMANDS

```
/// Each particle system (cloned or not) needs
memory to store its vertex buffers in addition
to the memory required for simulation. As the
number of particles in a system can change
every frame due to new particles being
generated or old ones dying, the amount of
memory required for its vertex buffer varies
similarly. While we could simply allocate
enough space to store the maximum possible
number of vertices when the system is
created, that would be wasteful if only a
few particles are emitted for the majority
of the effect's duration.
```

We instead use a dynamic vertex buffer heap, out of which we allocate all vertex buffers that are only needed for a single frame. Because the particle vertices are built on the fly every frame and don't need to be persistent (besides being double-buffered for the GPU), we can use a simple linear allocator. This is an allocator that is cleared every frame, and every allocation is simply placed at the beginning of free memory. This has a number of advantages; fragmentation is completely eliminated, performing an allocation is reduced to simple atomic pointer arithmetic, and memory never needs to be freed—the "free memory" pointer is just reset to point at the beginning of the heap at each frame.

In addition, this heap doesn't have to be limited to the particle systems' buffers. It is used by any code that builds vertex buffers every frame, including skins, motion trails, light reflection cards, and so forth. With this large central heap, we only ever pay for the memory of objects that are actually being rendered, as any dynamic objects that fail the visibility test don't need any memory for that frame. If we allocate memory for each object from when it's created until it's destroyed (even if you rarely actually see the object), we use far more memory than we do by consolidating vertex buffer allocations like this.

FINE-TUNING RENDERING PERFORMANCE

```
/// It's easy for effects to get out of control
in PROTOTYPE 2's game world. Explosions,
smoke, blood sprays, fires, and squibs all
go off regularly,
```



FIGURE 1

The original effect (left) and three variations scripted with different biases and overrides

Alternately, we could instead just perform per-frame allocation in the effects heap, but creating and destroying these buffers every frame adds churn, increases the possibility of memory fragmentation, and demands more processing overhead for doing many dynamic allocations.

often all the same time. When this happens, the large amount of pixels being blended into the frame buffer slows the frame rate to a crawl. So we had to dedicate a significant amount of our effects tech to identifying and addressing performance issues.



Figure 3A (Top): has improved colors over Figure 3B (Bottom).

effect at a certain distance. These values are then interpolated between all LODs based on the effect's distance from the camera. For example, the artist might choose to lower the emission rate and increase the particle size of an effect when it's far away—this would reduce the amount of overdraw while still maintaining a similar look, but with less of the detail that would only be noticed up close. The interpolation results in a continuous LOD transition that doesn't suffer from any popping or other similar problems—although they still have the option of switching to a completely different effect at a certain distance (with a cross-fade) or disabling the effect altogether. While reducing GPU cost is the main goal, these LODs usually end up saving both memory and CPU time, too.

The other metric we use when choosing LOD is the rendering cost of the previous frame's particles. This uses the same occlusion query results as the statistics given to artists and is fed back into the LOD system. If the previous frame was relatively expensive, we don't want to make the current frame worse by spawning even more expensive effects, so we instead play cheaper LODs in an attempt to recover faster. The artist has full control over what LOD to choose and at what level of performance it should be used.

When the frame rate drops significantly due to particles, it is often not because of one expensive effect but due to many moderately expensive effects all going off at the same time. Any optimizations done in this regard must take into account what other effects are playing. For this we have "effect timers." Using an effect timer, we can check whether a particular effect has already been played recently, and choose



to play different effects based on this. A prime example is a big expensive explosion; we might only want one big explosion to go off at a time, and for any other simultaneous explosions to be smaller less expensive ones. This often happens when a missile is shot into the middle of traffic and three or four cars explode at the same time—one car will play a good looking effect, while the other cars play smaller, cheaper ones. The visual impact is similar, but at a much lower rendering cost.

Although our effect scripting system is meant mostly for optimization of rendering, it has useful applications for gameplay too. For example, when the player fires a tank shell into the distance, we want a suitably impressive and impactful explosion, but if the same explosion were to play right in front of the camera when the player is hit by an AI's tank shell, the result

would likely blind the player for a few seconds and completely block their view of the action. This can be very frustrating in the middle of combat, so in these situations we can use different LODs to reduce the number of particles, lower the opacity, and make the effect shorter and smaller. Not only does this make PROTOTYPE 2 play better, it also uses a cheaper effect that has a lower impact on framerate.

PARTICLE RENDERING

/// Even with our LOD and scripting system doing its best, the mayhem of PROTOTYPE 2 means it is still possible for particle effects to become too expensive. When this happens, we take the more extreme measure of switching to bucketed multi-resolution rendering (as presented by Bungie's Chris Tchou at GDC 2011—see Reference 3). The

decision to switch to a lower resolution render target (in our case half resolution—25% the number of pixels) is also based on the previous frame's particle rendering cost. When it is low, all particles render to the full resolution buffer. This avoids having to do a relatively expensive upsample of a lower resolution buffer, which in simple scenes can be more expensive than just rendering particles at full resolution.

Once performance slows to a certain level and the upsample becomes the better option, we switch certain effects to render to the lower resolution buffer while the rest of the effects stay at full resolution. In this case, the artists need to choose which effects need to stay at full resolution, usually small ones with high-frequency textures that suffer the most from the drop in resolution, such as sparks, blood, and fire. All other effects drop to rendering at the lower resolution. When even that results in too much GPU time, as a last resort we switch to every effect rendering into the lower resolution buffer, regardless of artist preference. For the upsample, we chose a nearest-depth filter (as used in *BATMAN: ARKHAM ASYLUM*—see Reference 4), which we found to be cheaper and better quality than a bilateral filter.

We wanted to keep the actual shader used by the majority of our particles as inexpensive as possible, so it's relatively simple. We call it the add-alpha shader, as it allows particles to render either additively (for effects like sparks or fire) or alpha-blended (for smoke) using the same shader. Whether the shader is additive or alpha-blended is determined by the alpha channel of the particle's vertex color. To do this

we pre-multiply the texture's color and alpha channels and use a particular blend function—see Listing 1 for the relevant shader code.

This is not a new technique, but it's one that is nonetheless central to our particle rendering; particles from every effect that uses this shader (and a shared texture atlas) can be merged together, sorted, and drawn in the same draw call. This eliminates the popping that would happen; otherwise if two overlapping effects were drawn as separate draw calls, there would be a visible pop when the camera moves, and the order in which they are drawn changes. The vertex alpha can also be animated over time, so a particle can start its life as additive but finish as alpha-blended, which is very effective for explosions that start with a white-hot bang and end with thick smoke that fades away.

You'll also notice in the code listing that there are two texture fetches. This is for simple subframe interpolation of our texture animations, which allows us to use fewer frames and still produce a smoothly animating image.

LIGHTING PARTICLES WITHOUT PIXEL SHADERS

/// IN PROTOTYPE 2, the world is split up into three zones; green, yellow, and red. Each zone has a distinct style and color palette, as well as a few different times of day. Without lighting and shadowing, particles look wrong in many situations—too flat, too light or dark, and sometimes just the wrong color (see Figure 3 for example). We realized they needed lighting but didn't want to add the expensive pixel shader

LISTING 1 ADD-ALPHA SHADER CODE

```
// Add-alpha pixel shader. To be used in conjunction
// with the blend factors {One, InverseSourceAlpha}

float4 addalphaPS(
float4 vertexColour : COLOR0,
float2 uvFrame0      : TEXCOORD0,
float2 uvFrame1      : TEXCOORD1,
float subFrameStep   : TEXCOORD2 ) : COLOR
{
// Fetch both texture frames and interpolate

float4 frame0 = tex2D( FXAtlasSampler, uvFrame0 );
float4 frame1 = tex2D( FXAtlasSampler, uvFrame1 );
float4 tex = lerp(frame0, frame1, subFrameStep);

// Pre-multiply the texture alpha. For alpha-blended particles,
// this achieves the same effect as a SourceAlpha blend factor

float3 preMultipliedColour = tex.rgb * tex.a;
float3 colourOut = vertexColour.rgb * preMultipliedColour;

// The vertex alpha controls whether the particle is alpha
// blended or additive; 0 = additive, 1 = alpha blended,
// or an intermediate value for a mix of both

float alphaOut = vertexColour.a * tex.a;
return float4( colourOut, alphaOut );
}
```

REFERENCES

- 1: www.gdcvault.com/play/1013444/The-Gym-Where-The-Incredible
- 2: <http://altdevblogaday.com/2011/02/12/alternatives-to-malloc-and-new>
- 3: www.gdcvault.com/play/1014348/HALO-REACH-Effects
- 4: <http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/OpacityMappingSDKWhitePaper.pdf>
- 5: http://developer.amd.com/media/gpu_assets/R2VB_programming.pdf

code in order to do per-pixel shadowing and image-based lighting, as this would have vastly reduced the number of particles that we could render.

Our solution was to do lighting per-vertex, but as a pre-pass into an intermediate “particle lighting” buffer. For each particle vertex, we render the lighting contribution to a pixel in the lighting buffer. This way we can use the pixel shader to do lookups into the shadow buffer and image-based lighting textures, using the same lighting code as the rest of the game and avoiding the performance pitfalls of vertex texture lookups on some platforms.

This lighting buffer is then read in the particle's vertex shader and combined with the vertex color, resulting in no extra instructions in the pixel shader. The only concern here was the performance of the vertex shader texture lookup on some platforms, particularly the PS3 and some earlier DX9 GPUs. In these cases we actually rebind the particle lighting buffer as a vertex buffer and just read from it as we would any other vertex stream. This is trivial on the PS3 as we have full control over how memory is viewed and accessed, and for the DX9 GPUs that support it, we use the ATI R2VB extension (as detailed in Reference 5).

PUTTING IT ALL TOGETHER

/// Particles are a significant part of bringing the world of PROTOTYPE 2 to life. Various performance management systems work together to deliver effects without exceeding available resources. Lighting and shadowing add a huge amount of visual quality, and by doing it per-vertex, we are able to light every particle in the world at considerably less cost than we otherwise could have. And finally, one of the most important aspects of effects tech development is giving the artists the tools they need to do their job—and to help us do ours. After all, they're the ones that make us all look good! 🙌

KEITH O'CONNOR is a senior rendering coder at Radical Entertainment in Vancouver, where he is currently working hard to ship PROTOTYPE 2, which will be out any moment now. He can be reached at keith.oconor@gmail.com, and random 140-character thoughts can be found at [@keithoconor](https://twitter.com/keithoconor).

[The author would like to acknowledge Kevin Loose and Harold Westlund who authored many parts of the original Radical particle effects systems.]

needs more

BLUR

BUILD A BOKEH DEPTH-OF-FIELD EFFECT WITH THE PS3'S SPU

Ever wanted your games to have that artsy, blurred depth-of-field effect during a cut scene? You know, the kind where the bright points in the background look like slightly blurry circles, and nothing in the background has any edges or details to distract from the focal point of the scene? That effect is called *bokeh*, a Japanese word meaning “confused” or “dizzy,” and in this feature we’ll show you how to replicate that effect using the PlayStation 3’s SPUs to analyze the frame buffer luminance and generate bokeh draw calls.

WHAT IS BOKEH?

/// To understand what the bokeh effect is, we need to first understand depth of field. Depth of field (DOF) is the distance from the focal plane in which objects appear in focus.



Depth of field is the distance between the nearest and farthest objects in a scene that appear acceptably sharp in an image.

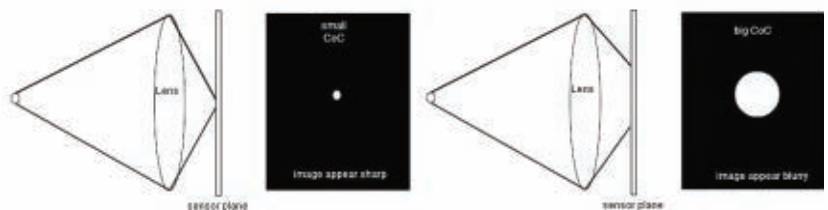
Objects outside this region are considered out of focus and will appear blurred. In film or photography, depth of field is mainly a characteristic of the lens and focus distances. Note that objects gradually go from sharp to blurry as they get further from the focal point. Since focus is a gradient, we quantify the blur amount with the size of the circle of confusion (CoC). The bigger the CoC value for a particular pixel, the more blur you apply to it. Conversely,

as the CoC value gets lower, the sharper that pixel will look.

Many games use depth of field to focus the player’s attention on a particular part of the frame and reinforce the depth illusion in the rendered image. Depth of field is especially useful in cut scenes, which typically have a more cinematographic feel, since it lets a cinematic director strategically blur out the background so the player will focus on a particular character or object in the scene.

Typically, to create the in-game depth-of-field lens effect, we blur a copy of the frame

buffer and interpolate between the blurred and nonblurred version depending on the distance of each pixel to the focal plane. For many years the blurring technique we used was just a simple Gaussian blur to approximate the lens effect. Bokeh, however, creates a much more real, filmic DOF effect. Typically, the bokeh effect will make highlights or light sources blur out into discs or disc-like shapes created by the number of iris blades in the camera lens. The bokeh effect starts with the lens aperture, or more precisely, the aperture shape. On a real camera the quantity of light passing through the lens is controlled by



Variation of the CoC for a combination of subject distances. For a particular pixel, the CoC value will vary between [0,1] and will indicate how focused or unfocused the pixel is. The blur amount, generally in pixel size, will be multiplied by the CoC of the pixel to find the blur amount for a particular pixel in the frame buffer. The maximum blur value will be game-driven, and the artists can tweak it to achieve the desired DOF effect.

BLUR

the aperture. A set of blades mounted into the lens controls the light entering the camera.



Aperture shape depending on the f-stop. We can see now where the bokeh shape is coming from!

We can now see how the lens design affects the shape of out-of-focus highlights. Typically, you will have a nice circular bokeh shape with a fully opened aperture. Some lens manufacturers have iris blades with curved edges to make the aperture more closely approximate a circle rather than a polygon.



Bokeh example from a real camera.

BUILDING BOKEH WITH SPRITES

/// Sprite-based bokeh effects are simple to understand and very flexible, since the artist can modify them to tweak the effect as desired. The main idea is to take each pixel of the frame buffer and analyze its luminance. You can work with a downsampled version of the frame buffer to improve performance. This will introduce some artifacts in the rendering of the bokeh sprite, but I have found that the speed gains make it worthwhile.

To analyze each pixel of the frame buffer, we start with a filter kernel and use it to analyze

FRAME BUFFER EXAMPLE WITH A 5 X 5 PIXEL KERNEL USED TO COMPUTE THE AVERAGE LUMINANCE OF THE PIXEL.

5x5 Kernel example (below).

$$\text{average pixel luminance} = \frac{\sum_{i=-1}^{+1} \sum_{j=-1}^{+1} \text{Lum}(i,j)}{n \times n}$$

$$\text{Lum}(i,j) = 0.2126 R + 0.7152 G + 0.0722 B$$

Luminance equations.

the pixels surrounding the current pixel. The bigger the kernel size is, the more expensive the luminance calculation is. In my tests I decided to go with a simple case, so I set the kernel size to 1, with a buffer downsampled by a factor of 2. The result was quite good in performance and quality.

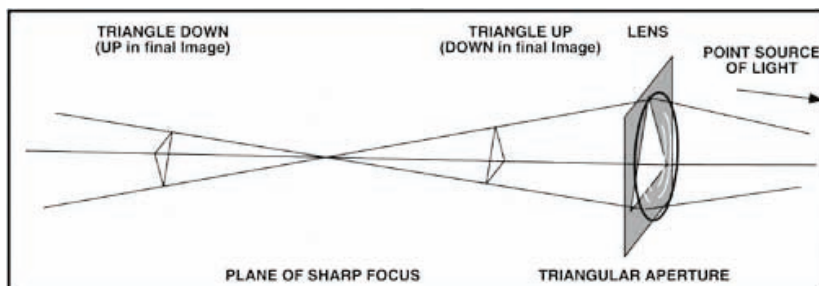
Using the filter kernel, we calculate the luminance for each pixel and simply spawn a sprite at the pixel screen space position if the luminance pixel is higher than a certain threshold. This luminance threshold value can be editable by the artist so they can adjust at which luminance value the pixel will produce

a bokeh sprite when the pixel is out of focus. Performing the threshold test on each pixel will give you a list of pixels that will spawn a bokeh sprite.

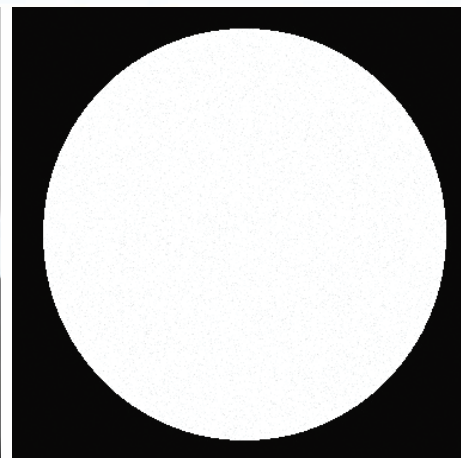
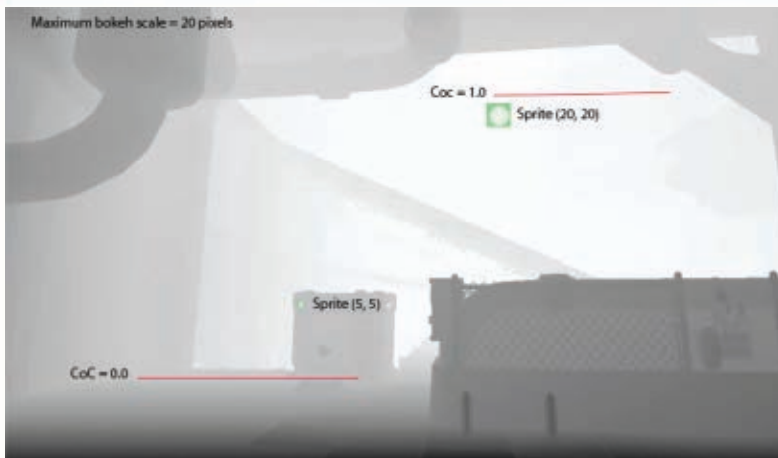
The last step is to calculate a proper scale for the bokeh sprite. The scale typically has a maximum size that artists can edit to their preference. In this step, the depth value of the pixel is used to determine how much the pixel is out of focus. This represents the circle of confusion mentioned earlier, and at its maximum value, it represents a fully open aperture.

In short, you need to calculate how much the pixel is out of focus and apply a pixel scale to the 1x1 bokeh sprite that you will spawn. The more the pixel is out of focus (remember that in the DOF explanation a pixel is gradually out of focus), the bigger the sprite will be on screen. If the scale value is 20 pixels the bokeh sprite spawned at a pixel fully out of focus will be 20x20 pixels.

At the end of this process, you end up with a list of sprites containing the X/Y screen space position of the pixel, the Z linear value of the pixel needed to depth-test the bokeh sprite in the pixel shader, UV coordinates of the bokeh texture, the CoC value of the pixel (needed to



Aperture shape affecting out-of-focus detail in the image.



CoC across the frame buffer (left).
Bokeh shape texture [64x64] (right).

adjust the blending amount to preserve energy conservation), and the color value of the pixel.

Depending on the platform you are developing for, you can do this process with different work units. On DX11, all this is realized in the compute shader with the color and depth buffer used as textures. On PS3, we don't have this stage available, but we do have the SPU.

LET'S SPUify THIS!

/// The PS3's memory is divided between main memory and video memory. Typical pipelines have the main color buffer

placed in video memory for performance and memory footprint reasons. Since the SPU's like to work on buffers placed in main memory (read/write mode), the first step is to transfer the main color buffer into main memory.

After the reality synthesizer (RSX) transfers the color buffer to main memory, the SPU can start analyzing the scan lines to find each possible location where a bokeh sprite should be spawned. Basically, SPU's will write sprite information in a vertex buffer reserved in main memory, and then the RSX will process that information to display the bokeh sprites. The SPU program then patches the draw call previously reserved in the command buffer

and removes the Jump To Self (JTS). JTS are used to synchronize the PPU and RSX on PS3. During this process, we calculate the average luminance of the color buffer needed in the tone mapping step, allowing us to save the Graphics Processing Unit (GPU) downscaling steps to find the average luminance of the color buffer.

The effect is very similar to the bokeh effect shown in the Samarithan DX11 demo realized by the Unreal engine team.

Let's detail the different steps:

1> Transfer the color buffer to main memory. You can transfer at full resolution or half resolution depending on the quality of the bokeh effect you want.



BLUR

Circular bokeh on.



2> Prepare n SPU jobs working on a subsection of the color buffer to analyze the pixel luminance.

3> Each SPU fills a vertex buffer with the bokeh sprite information.

4> On the PPU, reserve space in the command buffer for these draw calls. Since the set draw call command on PS3 has a variable size in the command buffer depending on the number of vertices, we must declare a maximum number of vertices and reserve that space in the command buffer. JTS commands are inserted before each set draw call so that the RSX waits until the SPUs are done.

5> On the PPU we issue n draw calls working on n vertex buffers depending on the number of SPU jobs we decided

to spawn to process the frame buffer. For example, if we decided to create two SPU jobs, both jobs would work on half of the frame buffer, and we would need to issue on the PPU two draw calls, each using their own vertex buffer and patched by the SPU jobs.

6> On the SPUs, each bokeh job analyzes the pixels and spawns a bokeh sprite for each pixel passing the luminance threshold, scaled by the CoC factor. The scale is clamped to a maximum bokeh scale size (in pixel space).

7> Each sprite is written in the vertex buffer (x,y,z position in screen space, UVs, and color) and the set draw call is patched with the correct number of vertices. The rest of the reserved space is filled with NOPs, telling the RSX to go

to the next graphic command.

8> The SPU patches the JTS so RSX can consume the graphic commands.

9> RSX draws each batch of bokeh sprites using additive blending.

10> Depth test is done in the pixel shader since we have the z position of the sprite in the vertex buffer.

11> The blend amount is adjusted to respect energy conservation for the bokeh sprite.

WHAT DOES IT LOOK LIKE DURING A FRAME?

/// There are various ways to hide the luminance analysis and bokeh draw call generation steps done by the SPUs. In my case, I decided to kick the RSX transfer right after the blended objects. This leaves enough time for the SPUs to analyze the frame buffer and fill the vertex buffer that the RSX will use to display the bokeh sprites on top of the frame buffer. The important thing to remember is to be careful not to stall the RSX.

As a bonus, since we're doing the luminance computation on the SPUs, we can have the total frame luminance for free. Normally, a game will



Bokeh effect time line example.



have some kind of luminance/tonne mapping adaptation of the frame buffer at the end of the frame. Adaptation effects usually involve the GPU by adding the work of doing a cascade of downscale passes to find the average luminance of the frame buffer. This obviously has some cost on the GPU and can be removed if you analyze the frame buffer on the SPUs.

THE FINAL PRODUCT

/// We found the present metrics using a half resolution 720p color buffer in main memory. One SPU working on a 640 by 360 pixel buffer took 2.8 milliseconds, while five SPUs working on the same buffer took only 0.65ms. The computations are done in a Structure Of Array (SoA) manner, so four pixels can be processed at once. However, the SPU could certainly be more optimized by better balancing the odd/even pipes and reducing instruction latency.

On the RSX, the cost is totally dependent on the number of bokeh sprites you decide to spawn and the screen coverage each sprite has (which is dependent on the bokeh maximum scale in pixels). The process of transferring to main memory with a draw call from a 1280 by 720 pixel surface to a 640 by 360 pixel surface

(untiled) took 0.44ms.

For testing the bokeh sprite draw calls, I used a capture to show performance, but remember that the number is dependent on the number of sprites per SPU job and the bokeh maximum scale.

The next image shows a test case to analyze the cost of bokeh sprites on the RSX.

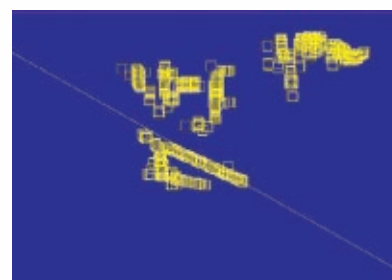
In total, the draw calls for the bokeh sprites cost 0.56ms for 5660 vertices with a 10-pixel maximum bokeh scale factor.

ROOM FOR IMPROVEMENT.

/// There you have it—a basic bokeh effect on the PS3. It's not perfect, though. We'll leave a list of possible improvements as an exercise to the reader. For starters, you could optimize SPU code by distributing the instructions evenly between the odd and even SPU pipelines.


Also, you could try to remove or push the vertex maximum number. One way would be to spawn one SPU job working on the whole color buffer and sort the bokeh sprites to use only the brightest ones. You could also work with a bigger kernel size to generate bokeh sprites.

Since the SPU computes the luminance for each pixel, you could have the total luminance



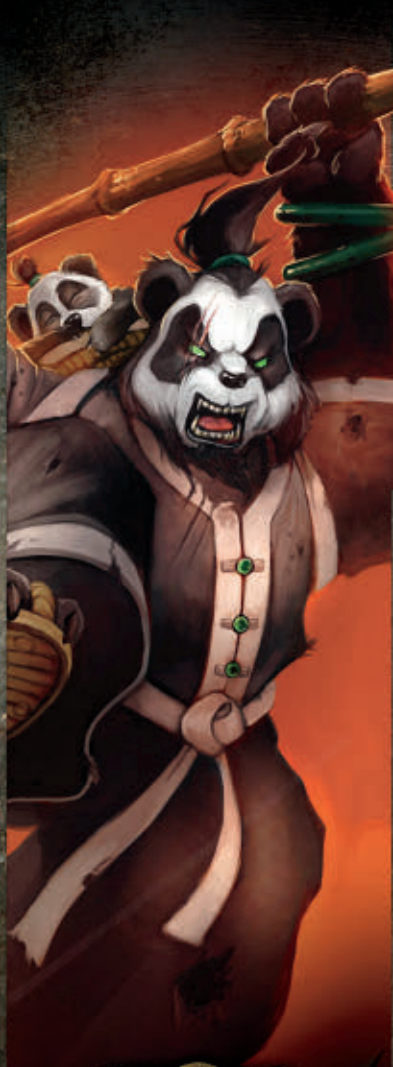
Wire frame view of the bokeh sprite draw calls.

of the color buffer without involving the RSX (downsampling and reading the final target on PPU). This could save the downsampling step on the RSX if you have some sort of eye adaptation post-process in your pipeline.

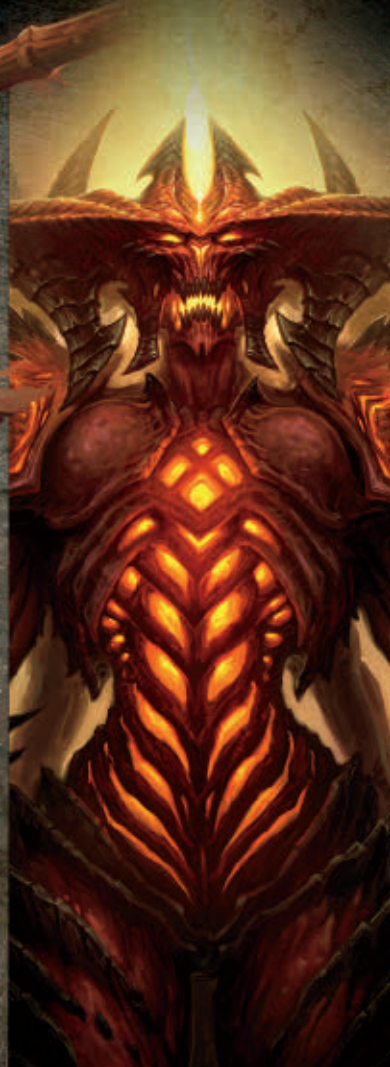
Finally, SPUs could write the bokeh sprite directly in the frame buffer. Instead of writing to the vertex buffer, SPUs calculate for each sprite the 2D transfers representing lines contained in the Bokeh sprites. The RSX would use this buffer in additive blend mode to add it on top of the frame buffer. 

SERGE BERNIER is a senior 3D programmer at THQ Montreal who specializes in the PS3. He has worked on OPEN SEASON (PS2), TEENAGE MUTANT NINJA TURTLES (PS2), SURF'S UP (PS2), FAR CRY 2 (PS3), and FAR CRY 3, as well as PS3-specific optimizations for HOMEFRONT. He is currently working on an unannounced title.

BLIZZARD ENTERTA



WORLD
WARCRAFT®



DIABLO®



STARCRRAFT®

BLIZZARD®

ENTERTAINMENT

jobs.blizzard.com | ur.blizzard.com

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

INMENT IS HIRING

Do you have the passion to create and
the will to forge great games?
If you seek challenge, inspiration, and strive for
excellence then this is the company for you!

We are actively recruiting across
the following disciplines:

ART/ANIMATION | AUDIO/SOUND | BUSINESS INTELLIGENCE
COMMUNITY DEVELOPMENT | CORPORATE ADMINISTRATION
FINANCE/ACCOUNTING | GAME DESIGN | HUMAN RESOURCES
INFORMATION TECHNOLOGY | LOCALIZATION | MARKETING
OPERATIONS | PRODUCTION | PROGRAMMING
PUBLIC RELATIONS | QUALITY ASSURANCE | WEB DESIGN
WRITING/EDITING

For more information regarding our opportunities,
please visit:

jobs.blizzard.com | ur.blizzard.com

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





Publisher: EA Partners

Developer: Big Huge Games/38 Studios

Number of Developers: 100-ish

Length of Development: 2.5 years

Release Date: 2/7/2012

Working Titles: Crucible, Ascendant, Project Mercury

Platforms: Xbox 360, PS3, PC

Number of health insurance carriers we went through from beginning to end: 6

kingdoms of amalur :reckoning

postmortem

MIKE FRIDLEY

OVER FIVE YEARS AGO, BIG HUGE GAMES SET OUT TO COMPLETELY CHANGE THE TYPE OF GAMES WE MAKE. WE SWITCHED FROM MAKING REAL-TIME STRATEGY GAMES TO ROLE-PLAYING GAMES, AND WE STARTED MAKING GAMES FOR CONSOLES IN ADDITION TO PCS. WE MADE THESE CHANGES FOR SEVERAL REASONS, AND ALTHOUGH PROFIT WAS ONE OF THOSE REASONS, IT WASN'T THE ONLY ONE. WE WANTED TO DO SOMETHING CRAZY. WE WANTED TO MAKE A BIG OPEN-WORLD RPG—PRETTY MUCH THE CRAZIEST PROJECT WE COULD THINK OF SHORT OF AN MMO. BUT WE'RE ALL BIG FANS OF THE GENRE AND THOUGHT WE COULD FIND OUR NICHE IN IT, SO WE STARTED OUR QUEST TO CONVERT THE STUDIO INTO AN RPG HOUSE.

At first, our RPG project was named "Crucible" and was being published by THQ. We were making great progress on it, and THQ was happy enough with the progress that they purchased us outright; and we became an internal THQ studio. Around that time we switched some of the key features of the game and renamed the project "Ascendant." We were part of the THQ network of studios for a short period of time right up to the point that THQ started running out of money. Our big, juicy, unproven-in-the-genre studio was a prime target for them to try to sell.

With literally days left on the "close the doors" timer at the studio, THQ sold us to Curt Schilling's 38 Studios, which has R.A. Salvatore as "creator of worlds." It became clear pretty quickly that we would need to change the universe and some of the game features yet again to take advantage of Robert's genius. We changed the project name to "Mercury," which later was given the final shipping name of KINGDOMS OF AMALUR: RECKONING.

>>>



For those keeping track at home, in five years we were bought and sold twice and changed the name and core features of the project three times. Needless to say, it's been a long, strange trip. The rest of the postmortem will be restricted to the two and a half years we spent working on RECKONING rather than the two previous false starts.

what went right

1 / COMBAT—RPGS DON'T HAVE TO HAVE BORING FIGHTS

Shortly after we came out of preproduction, we took a long, hard look at the game we were making and tried to figure out where we were going to be better than the competition. We figured that open-world RPG designs are segmented into four basic quadrants: story, character progression, exploration, and combat.

We discovered that it was easy to identify the games leading the industry in story, progression, and exploration, but there was no clear title that does combat well while still meeting the expectations of the player in the other three quadrants. So we decided to go

all-in on combat and change our staffing plan to really commit to making combat fun in an open-world RPG.

The game wasn't built solely around combat, but it was definitely built with our flavor of combat in mind. Everything from the minimum size of a dungeon's hallway to the number of enemies we could handle onscreen at a time was governed by the guideline that combat had to remain awesome.

Two of the other things that went right during development were direct results of this focus on awesome combat, usability testing and functional group seating.

2 / USABILITY TESTING—EARLY AND OFTEN

We made sure that getting feedback from real players was high on our priority list from the very beginning. Since we couldn't just release work-in-progress builds to the public and take surveys, we did the next best thing and took advantage of EA's usability lab very early in the development process. The lab at EA allowed us to pull in testers from the general public and use them for highly focused testing on systems or content that we were currently developing. For example, if we had

the first pass of a crafting system in the game, we could pull in a dozen or so players for a half day and get some players feedback on whether the interface was easy to navigate or whether blacksmithing felt rewarding.

Since EA's lab recorded videos of the wrap-up sessions, we were also able to show our team what the player thought of their part of the game. If the attack chain you were working on felt bad or the quest didn't make any sense to the normal player, the team that worked on those areas of the game got to hear it straight from the consumer's mouth. That kind of direct feedback from the player really helped us fine-tune the combat system, and ultimately, the entire game.

3 / FUNCTIONAL GROUPS—SITTING TOGETHER PAYS OFF

As part of our development philosophy, we have cross-departmental teams working closely together. A lot of studios do this, but until this project we didn't really push seating functional groups of people together at BHG.

Some of that may have been because the physical structure of the studio didn't lend itself to more than three people in an office, or it could have been just old-school thinking that never changed until it was forced to change. We did eventually break down the walls (literally) and start sitting larger functional groups together in what we called "pits" around the office. For example, the combat pit has animators and designers all sitting side by side.

This way, an animator working on an attack chain could be sitting just a few feet away from the designer implementing and fiddling with it in-game. They could easily look at each other's work and offer comments or critiques very quickly.

However, functional groups are less about speeding up the feedback process and more about forcing interaction. A lot of developers are lazy about socializing or unaware of what is going on outside their office, but when the people you are directly working with are in your face all day, you start to bond with them. A lot of our functional groups became pretty tight-knit and hung out after hours, really bonding as a group. That translated into more and better communication in their work and really increased the quality of the end product.

4 / SCRUM DEVELOPMENT METHODOLOGY

Before preproduction started on RECKONING, Scrum was starting to gain a lot of momentum in the game development community. I don't think that Scrum is the only way that people should be developing games these days, but after running pure Scrum for the entire development of RECKONING, I'm a firm believer in its methods.

I won't go in to the details of Agile development here, but the basic element of Scrum that made it



so successful at BHG is the ability for the individual developer to estimate his or her own work.

The old days are gone. You can't expect producers or leads to come up with a huge waterfall of everything they thought would get done over the next three years. In the game development business, it's insane to think you have any insight into what your team will be doing one year from now. You can set major milestones with hard dates, but filling in all the details between those points is an exercise in futility.

With a basic understanding of our time metrics on content development, for example, we were able to do some good old-fashioned waterfall scheduling as well. But those waterfalls were used only to illustrate to the team the pace we'd need to maintain in order to complete the scope of work in the time allotted. For example, we could tell one of our environment artists that he had three months to create all the base pieces of a particular biome before he would start taking time away from the next biome, but we did not plan out any more detail than that. We didn't account for every tree, rock, and scrub in that biome. The actual planning of what would go in to that base set and how long it would take, meanwhile, came from the sprint planning sessions where the artist would come up with his own tasks and time estimates.

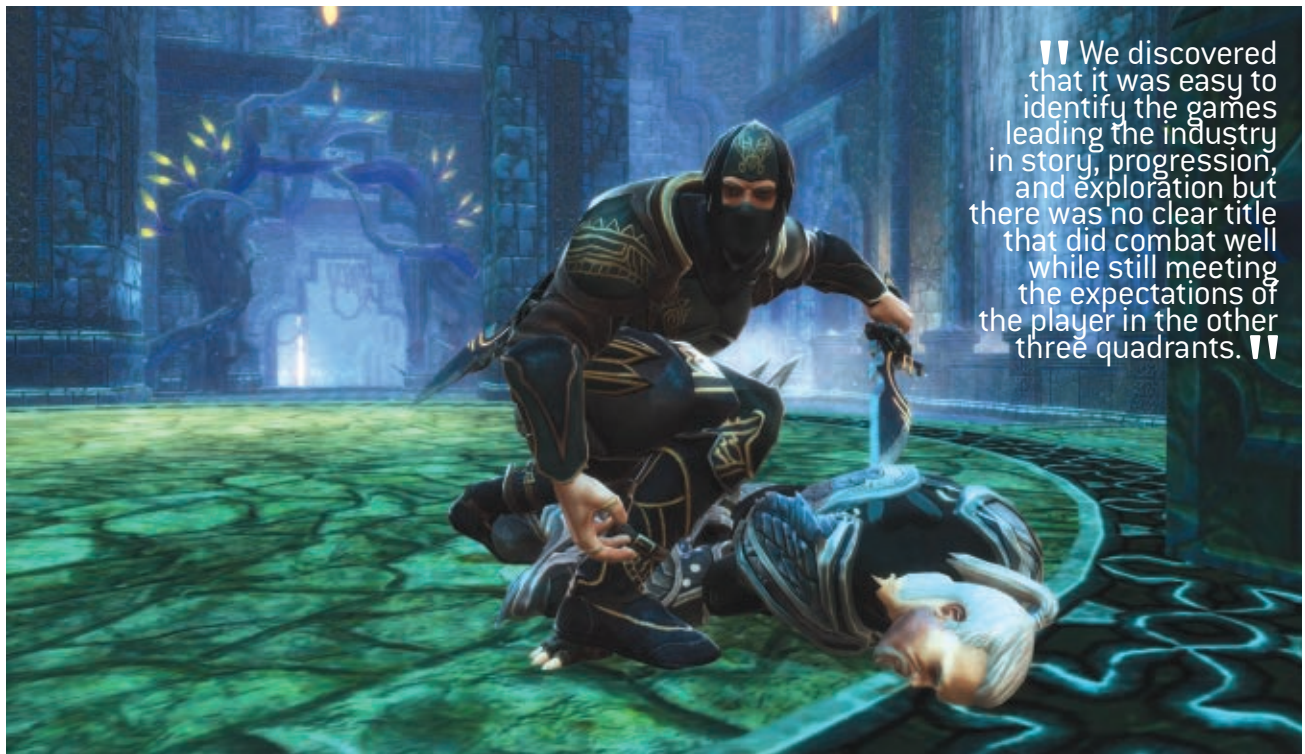
Not only did Scrum allow us to plan better, but it also gave the entire team a lot more ownership and visibility over the game. If something came up (and it always does in game development)



the team knew that not completing what they had already committed to meant that it was in danger of being cut. That resulted in lots of mini-crunches throughout the entire life cycle of the game instead of one humongous death march at the end of development. The team would rather work a little overtime than see something they really wanted in the game get cut or be done

poorly. We still had some end-of-development crunching, so Scrum isn't a silver bullet, but it definitely helped.

Scrum allows for that day-to-day accountability that was missing for so long in game development. You understand within 24 hours of a change what is going on. More traditional development methods wouldn't catch those small losses of time for



▼▼ We discovered that it was easy to identify the games leading the industry in story, progression, and exploration but there was no clear title that did combat well while still meeting the expectations of the player in the other three quadrants. ▼▼



months, which would either force us into a huge crunch at the end of development or make us cut an entire system or group of content. Also, allowing the entire team to add items to the product backlog was a big win for us.

5 / EA PARTNERS—A GREAT RELATIONSHIP

Working with a large publisher often can be challenging. Some publishers want to be too involved in the day-to-day development decisions. Other publishers will go to the opposite extreme and remain silent milestone after milestone until your game hits Alpha, at which point they suddenly have issues with things that have been final for months, like the art style or specific gameplay systems. Fortunately, the EAP production crew was neither of those types of publishers. They gave excellent feedback throughout the development cycle and did what you really want from a publisher: They offered excellent support where we needed it most.

During our first meeting with our EAP producers, they showed up with a bunch of PowerPoints outlining all the services we could take advantage of during development—and take advantage we did. That set the tone for the next couple of years. Any time we had a bump in the road, EAP was there asking what they could do to help. Having options like that available to you when you have an issue is a huge asset.

As they are probably largely unsung for their efforts, I want to call out the major production staff players over at EAP that were our go-to guys: David Yee, Ben Smith, Craig Krstolic, and David Luoto. You guys made a lot of big problems much smaller. Thanks.

what went wrong

1 / PREPRODUCTION—ENTIRELY TOO SHORT

Even though we had a lot of production time during the false starts before RECKONING, our preproduction time for RECKONING itself was entirely too short. At the beginning of RECKONING, we were in heavy pitch mode and our goal became to get a publishing deal signed instead of spending the time to figure out the normal outputs you are looking for in the preproduction phase of development. Once we signed a deal with EAP, we needed to get into full production quickly. Or so we thought.

What we should have done was make sure we had defined everything that needed defining. We had a basic scope of content, but we hadn't done much to understand the feature set or the game's major hook. We decided to go all-in on combat fairly soon afterward, but the development budget and schedule had already been set, and so we weren't able to anticipate how many additional animators and designers we would

need to bring the combat system to life.

Our content pipelines weren't fully fleshed out, and we only had basic or less than basic functionality of some of the tools we would need to create that content. But given our tight schedule and the mountain of content we had to produce for this game, we jumped in to full production with a lot of questions unanswered. Needless to say, this is not ideal.

We also hadn't really figured out the density of our content (quests, reagents, dungeons, and so on), which had long-term negative repercussions for both design and production. We frankly made "too much game," and we probably wouldn't have (at least not to that extent) if we had more time in preproduction to figure out the density question.

2 / TOOLS AND PIPELINES—LAYING DOWN THE TRACK WHEN THE TRAIN IS ON ITS WAY

As I mentioned above, we didn't have a good head start on the development of our tools and pipelines early in development. We knew the basics of what we wanted once we had a feature set figured out for the game, and we knew the type and quantity of content that we were planning on making, but we really didn't have a clue how much tools work we needed to do.

A lot of the systems in the game were still very much in the blueprinting stages where we weren't even sure how they would function in the final game.

The dialogue system is one example. We came out of preproduction without giving much thought to the dialogue system other than, “Yeah, we should probably do that.” We didn’t nail down how we would display dialogue and choices in-game and how designers would enter that data in a tool.

This put a huge amount of pressure on our tools programmers. They had to jump from tool to tool getting functionality to a point where users could actually use it right before they needed it. In a lot of instances, our tools programmers had to roll out a tool before it was fully functional or bug free because of time constraints.

Obviously, this hurts content creation. Devs would submit tool feature requests and not see any movement on them for months (or ever, in most cases) because the tools team had a ton of other issues that were higher priority. It basically meant the majority of our tools were functional but woefully inefficient.

I’m truly amazed that the tools team did as well as they did with such limited time and manpower. For the most part they were able to stay just ahead of the train, and we ended up with a suite of tools that—while still a bit disjointed—work pretty well. Things will only get better as we ramp up into preproduction on our next project.

3 / DEMOS—TOO DAMN MANY OF THEM

People who know me are probably expecting me to go completely off the deep end and start bashing marketing and PR right now, but I’m not going to do it. I understand how hard their job is, and how that job is fundamentally motivated by events that are counter to the way developers like to work.

Developers, especially producers, like to be proactive. We make schedules, and we plan dependencies. That’s how stuff gets done. Sure, problems pop up that we have to react to, but the goal is to reduce those as much as possible.

Marketing and PR are by necessity much more reactive in their work. If you sat down and tried to formulate the next two years of a marketing plan with the same level of detail that a development schedule has, it would be full of every single possibility that could arise while trying to sell the game. A very small percentage of those opportunities would be sure things. There are some major milestones that can be planned well in advance, such as E3, but you have to remain flexible and opportunistic with a new IP to ensure you follow through on opportunities as they open up. Of course, that means that they’ll come to game developers, say “We have this great opportunity, but we’ll need a brand-new demo and 30 never-before-seen screenshots by the end of the month,” and drive us crazy. Being a brand-new IP, we were aware we couldn’t get away with a single demo at E3 and a few dozen screenshots and videos. We knew we had to get our awareness up so people would start paying attention to our game. Marketing

decided that the best way to do that was show the press as many different things about the game as possible over a very long period of time.

I’m trying to remember the number of demos we had to create over the development cycle of RECKONING, and I honestly end up losing count. Doing a demo for us was a pretty major undertaking, like it is for almost everyone in the business. You’re basically taking content and systems that were meant to be first or second pass at a certain point in the schedule and bump it all up to shippable quality long before it’s supposed to be shippable quality. This results in a lot of work that is just thrown out because the real content and the real systems end up changing a few weeks or months later. And there is nothing quite as frustrating as working overtime on something that you know is just going to be seen once and then thrown away.

The consumer demo was another hurdle to overcome. There was no way we were going to be able to complete work on the game and create a downloadable demo in parallel. We just didn’t have the time. In the end, we had to outsource the demo, and they had to build something with old code and not a lot of time. The result was a buggy experience, but still an experience that a lot of fans enjoyed.

In the future, we’ll be sure to plan plenty of time and budget for multiple press demos and work on a better plan to either build the downloadable demo ourselves or better support outsourcers.

4 / MAIN QUEST—NOT ENOUGH OF IT FLESHED OUT EARLY ENOUGH

In RECKONING, a lot of the custom content work we did focused on the main quest. There is a lot of custom content throughout the entire game, but we knew we really wanted to spend more of our time on the main quest, as most players would see the majority of that line. A big chunk of that custom work was cinematics.

Our cinematic team is awesome but very small. Much like most of our teams on the project, they have to produce more content than would normally be expected for a team that size. Not locking down the major beats of the main quest early really hurt the cinematics team.

Going from a storyboard to a finished cinematic takes a long time. Once a cinematic is finished, it is very costly to change. Because we weren’t locked down on the major cinematic moments in the game for so long, we ended up having to cut several cinematics that we really wanted to include. The cinematics we have in the game are awesome, and we got all the major beats that we wanted, but we definitely wanted more.

5 / UPPER MANAGEMENT SHUFFLE

I should probably give a little background on this point before I get into the meat of the issue. When we were purchased by 38 Studios, we retained

all of our senior management and development staff. The BHG studio was reporting to 38 Studios corporate, then based in Massachusetts.

In July 2010, about a year into the development of RECKONING, five of the most senior studio management team at BHG left the company. This easily could have ended RECKONING in a lot of different ways—the studio could have closed, or the game itself could have become a mess of unrecognizable trash. Luckily, that was not the case. Several people in the BHG studio stepped up to fill the leadership void so we could continue to make the game you’ll see at release.

The culture of this studio is unlike anything I’ve seen anywhere else. I don’t want to say we’re a family, because that has become cliché. Curt Schilling is fond of using sports metaphor; I’m more fond of military ones. To me, what drives the folks at BHG to do better and go that extra mile is our loyalty to each other. To use a military metaphor, it’s like fighting a war, but without all the courage and killing. When you’re in the thick of battle, you don’t fight a war for the general back at HQ. You fight it because you don’t want to let down your buddy next to you in the foxhole. We had other motivations, like wanting our fans to have a great game, but our day-to-day drive came from not wanting to fail each other.

If anything, the senior-management shuffle may have even increased the resolve of the studio to finish this game. It ended up being yet another obstacle that the fates threw at us, and we’d be damned if we weren’t going to get past it and make an awesome game.

CONCLUSION

I can’t possibly hope to cover in this article all the things we did right and wrong on a project this size. It was a huge undertaking to make a game of this scope, and we learned a lot along the way. The studio has definitely leveled up as a whole, and we’ll be heading into our next project with a better understanding of our game and with better tools and pipelines to make that game.

In the end, any success that this new IP will enjoy has largely been brought into being through the force of will and talent of its developers. We were understaffed and underfunded, but we simply had too much personal skin in the game to let it fail. The team that finished this game did it through dedication to what we all believed could be the next big single-player RPG franchise. I have never seen a team with so much ownership of a game as this one. Many nights were spent working on some minute detail simply because that developer didn’t want to let something that wasn’t perfect into their game. That kind of passion is a rare commodity to find in a handful of people—much less an entire studio—and I can’t wait to see what we can accomplish next. 🎮

MIKE FRIDLEY was executive producer, KINGDOMS OF AMALUR: RECKONING at Big Huge Games.

INNOVATION UNVEILED



LOS ANGELES CONVENTION CENTER

JUNE 5-7, 2012

E3 is the world's most important annual gathering
for the video game industry.

REGISTER NOW at E3EXPO.COM



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.

Produced by
IDG
WORLD EXPO

© 2012 Entertainment Software Association



TOOLS REPORT FROM THE SHOW FLOOR:

GDC 12

THIS YEAR'S GAME DEVELOPERS CONFERENCE SHOW FLOOR WAS PACKED WITH TOOLS AND MIDDLEWARE DEVELOPERS COURTING TRIPLE A DEVELOPERS AND INDIES ON SHOESTRING BUDGETS ALIKE. WE INVESTIGATED THE NEW PRODUCTS ON DISPLAY TO GIVE YOU AN INTRO TO THIS YEAR'S TOOL SPREAD.

Google Native Client

GOOGLE INC.

WWW.GONACL.COM

/// Flash and JavaScript are nice and all, but there's nothing like C or C++ for sheer speed, especially for games. That's what Google is banking on with its emerging Native Client (NaCl) browser technology, which could bring some power to web games.

Native Client is currently built into the Google Chrome web browser, and aims to let developers build web applications that are secure, sandboxed, and compatible with multiple operating systems like existing web apps, but speedy enough to handle taxing graphics, sound, and media playback functions that you typically need to do with native code. Imagine being able to embed a C/C++ application in a web page like you would a Flash application, and you have the basic idea.

Porting your game to Native Client means your game is playable on any platform capable of running Google Chrome (well, except Android): Mac, Windows, Linux,

and eventually Google Chrome OS. Google would like you to sell your apps or games through the Chrome web store, but since Native Client is an open source project, other browser developers could develop NaCl plug-ins that let you use your apps there, too.

2011 indie hit BASTION is probably the most notable game built for Native Client at the moment. While BASTION isn't terribly demanding compared to a heavy-hitter like CRYISIS 2, a world where web games have the complexity of BASTION at the base level, and Native Client starts to make good sense.

Adobe Flash Player 11.2 and AIR 3.2

ADOBE SYSTEMS INC.

GAMING.ADOBE.COM

/// Adobe wants 2012 to be big for Flash and games, and hopes the new versions of Flash Player and Adobe AIR (11.2 and 3.2, respectively) might be the ones to bring Flash in games to a higher performance level.

The biggest addition to the Flash/AIR set is Stage 3D, which



Unreal Engine running in Flash.

is what Adobe is calling a new set of 2D/3D rendering APIs that can take advantage of GPU hardware acceleration to boost performance by a claimed 1,000%. In other words, Flash should now be fast enough that Epic Games' announcement last October about building Unreal Engine 3 to support Flash makes a lot more sense. If that's the kind of tech we can start seeing Flash game developers work with for 2012, this may significantly change the environment for mobile and social games, especially.

Prior to GDC 2012, Adobe also announced that it would be resuming work on Alchemy, the

Adobe Labs project from 2008 that allows Flash developers to execute secure C/C++ code within Flash at runtime. If Alchemy materializes in 2012, it could prove to be a compelling challenger to Google's Native Client platform.

Autodesk Gameware

AUTODESK

GAMEWARE.AUTODESK.COM

/// You probably know Autodesk best for its widely used content creation tools, such as 3ds Max and Maya. This year, Autodesk



is going big with its middleware applications, of which Cognition, Population, and Scaleform were the most notable at GDC 2012.

Cognition and Population are the newest members of the Autodesk Gameware family; Autodesk acquired GRIP Entertainment, developer of GRIP Character Control System and GRIP Digital Extra System in November 2011, and those two applications became Cognition and Population, respectively. Cognition is a visual artificial intelligence tool that lets designers build and debug AI behavior trees, and Population is used to help developers quickly produce and direct “digital extras”—background NPCs—in Unreal Editor to make game worlds feel more immersive.

Scaleform, Autodesk's Flash-based user interface development tool, is nothing new to many game developers, but its 4.1 update (due in Spring) will make it a bit friendlier to mobile game developers. Autodesk noted that developer GlobZ used Scaleform to port a Flash game called TWINSPIN over to iOS wholesale. The developer says Scaleform helped it stay under the App Store's size limit, bump framerates up from 25 frames per second to 60, and iterate faster by speeding up the compiling process.

Also new to Scaleform 4.1 are a revamped system profiling tool (AMP) for tracking performance issues, and full ActionScript 3 compatibility, which was announced at GDC 2011, but not fully implemented until version 4.1.



FMOD Studio

FIRELIGHT TECHNOLOGIES

WWW.FMOD.ORG

/// Firelight Technologies is hoping that its new FMOD Studio, an overhaul of FMOD Designer 2010, will be your go-to off-the-shelf audio creation tool for high-end,

memory-efficient sound features.

Firelight's big talking point for FMOD Studio is the revamped user interface, which wraps the music editing, event editing, and mixing desk features into a package designed to be familiar to any audio professional used to working with professional digital audio workstations.

On the show floor, Firelight had one sound designer demonstrating how a team's audio specialist could build a complex soundscape on his own. With the event editing tool, he could stitch together a sound environment for a submarine and test how it would change as the character walks through different areas, or as the submarine's engine speed or depth changes, and so on. When it comes time to integrate that audio into the game itself, Firelight says all the coding team needs is the name of the sound event and the parameters it relies on.

Unfortunately, FMOD Studio is still very much in development, so you'll have to wait a bit before you can work it into our projects; Firelight was demonstrating a pre-alpha build, but the full suite should be shipping in Q2 2012.

Hansoft 6.7

HANSOFT AB

WWW.HANSOFT.SE

/// Hansoft came to GDC to show off Hansoft 6.7, the latest version of its project management system (last updated in January 2012), which is built around integrating multiple production methodologies.

With Hansoft 6.7, though, Hansoft is making a play for smaller teams as well, by offering a start-up license for teams of nine people or smaller—presumably so independent or student teams will get hooked on Hansoft and pay for the license if they get a larger budget to work with.

Hansoft says one of its major selling points is the relatively friendly user interface, which Hansoft will actually be changing significantly in the next update, though none of the representatives commented on what would change. The next update will also bring a

native iPhone app, though there's no official word about an iPad-optimized version besides “We're hoping to make one.”

Simplygon 4.0

DONYA LABS

WWW.DONYALABS.COM

/// No artist likes to spend her time building level of detail (LOD) models out of her fully textured, delicately modeled work of art, especially when there are a million other models that need building. That's how Donya Labs aims to sell Simplygon, its suite of automated LOD-building tools that cut polygons, rebuild low-poly replacement meshes, and retexure materials to try to save you time.

New to Simplygon 4.0 is BoneLOD, a new tool that can take skinned, rigged meshes and simplify them by removing bones. You can use BoneLOD to remove an artist-specified number of bones for an LOD, or simply have it remove bones that don't affect the model's skins.

Compared to some of the other tools in this roundup, Simplygon is much more specific, but the company promises big results—Donya Labs estimates that automatic LOD generation can save as many as four or five hours of work per asset. Simplygon won't leave any polys unturned, either. The company demonstrated a flyby of an in-game scene before and after running Simplygon, and the “after” clip pulled off about two to three times the frame rate with one-third of the polygons.

xaitMap and xaitControl

XAITMENT

WWW.XAITMENT.COM

/// German artificial intelligence middleware developer xaitment (pronounced “excitement”) showed off xaitMap and xaitControl, two new products it hopes will make your games smarter.

XaitMap is a pathfinding development tool that allows designers to script and tweak

a game character's pathfinding and movement routines. On the show floor, xaitment reps were demonstrating the newly announced xaitMap Unity plug-in, which integrates directly into the Unity game engine. Within a few minutes, the demonstrator had defined a map's static geometry, selected a path for a patrolling NPC to follow, and tested the NPC's behavior as its path got obstructed by doors opening and closing and bridges raising and lowering.

“Pathfinding is the entry-level point,” said Mike Walsh, xaitment CEO. “We want to make intelligent character behavior. Give him game logic. We want to make your characters smarter.”

This is where they pitch xaitControl, xaitment's behavioral modeling tool which lets designers set up state machines to dictate an NPC's behavior in a visual flow chart interface, introduce varying levels of probability to randomize behaviors between NPCs, and build nested state machines to make behaviors even more individual and complicated. Designers can also make changes to the AI behaviors, test them without recompiling, and view a very detailed debugger report, which the company says should make fine-tuning your AI much easier.

modo 601

LUXOLOGY

WWW.LUXOLOGY.COM

/// Loyal modo users will undoubtedly appreciate Luxology's version 601 update to its 3D modeling and animation suite—there are a whole bunch of new features that the company hopes will give it a competitive edge in the next round of the eternal 3D modeling software wars.

First up is the new set of tools for retopology modeling. At the show, an artist demonstrated the Topology Pen, a combination of multiple tools that he used to “weld” points and edges together very quickly to reduce the polygon count. He started with a very detailed model imported from ZBrush that weighed in at about

three million polygons, and with a few minutes of flicking his pen here and there, he ended up at a svelte 300. You'll have to wait for our full review to see how well those claims stack up to our testing, but it certainly looks promising.

modo 601 also includes an update to the replicator feature, which was first introduced in modo 501. modo 501's replicator tool let you replicate multiple instances of a single object around a plane, but you could only randomize the scale and rotation, so if you were using it to scatter some leaves around a scene it wouldn't necessarily look natural. In modo 601, you can build replicated objects with slightly different animations and tweak individual instances of a replicated object slightly without having to create a separate mesh, letting you build more realistic replicated objects.

Tools to watch for in 2012

The most impactful tools and services aren't always the ones with the biggest booth. Here are some hidden gems from the floor.

FABRIC ENGINE

FABRIC ENGINE, INC.
WWW.FABRIC-ENGINE.COM

/// Perhaps the best way to think of Fabric Engine is as a tool for building tools. At its heart, it's a browser plug-in that allows developers working with JavaScript (support for Ruby and Python coming later this year) to build applications that can take advantage of multi-core CPUs and GPUs and integrate C++ code libraries from within a browser window. Instead of rigging a model in Maya, motion-capturing in MotionBuilder, cleaning it up back in Maya, and exporting it into your game's runtime, Fabric Engine thinks you should be using its

platform to build your own tools around your game engine itself to avoid the back-and-forth between several different third-party tools.

LIVE DRIVER

IMAGE METRICS
WWW.IMAGE-METRICS.COM

/// Imagine that as part of your game's new character creation process, your player would use his PC's built-in webcam to automatically map his basic facial features, track his movements, and animate those facial movements with his in-game avatar. That's basically what Live Driver does. Trash talking on Xbox LIVE will never be the same.

FLEX 13

OPTITRACK
WWW.OPTITRACK.COM


/// OptiTrack wants to bring motion capture to the masses with the Flex 13, its new motion capture camera that can capture 120 frames per second at a 1.3-megapixel

resolution for only \$1,000 per camera. A thousand dollars might not be cheap for a shoestring indie budget, but it's low enough that small studios might decide to invest in their own motion capture rig instead of buying time in a high-end studio or trying to do without.

GRAPHICS PERFORMANCE ANALYZERS 2012

INTEL

[HTTP://SOFTWARE.INTEL.COM/EN-US/ARTICLES/GPA-FAQ](http://SOFTWARE.INTEL.COM/EN-US/ARTICLES/GPA-FAQ)

/// If you're building games for Intel hardware, you're probably already used to the Graphics Performance Analyzers (GPA) toolset, which lets you read live performance reports while your game is running so you can find your performance bottlenecks. New and notable to GPA 2012 is an additional set of diagnostic tools for Atom-powered Android smartphones, which could be a shot in the arm for Android developers. 

VFS
25
YEARS

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.

The LA Times named VFS a top school most favored by game industry recruiters.

Find out more.
vfs.com/enemies

"VFS prepared me very well for the volume and type of work that I do, and to produce the kind of gameplay that I can be proud of."

DAVID BOWRING, GAME DESIGN GRADUATE
GAMEPLAY DESIGNER, *SAINTS ROW: THE THIRD*

VFS STUDENT WORK BY BRENDAN BOYD

gd

FRONT LINE AWARDS 2011

CONGRATULATIONS TO THE 14TH ANNUAL FRONT LINE AWARD WINNERS

THE FRONT LINE AWARDS HONOR AND RECOGNIZE THOSE COMPANIES WHOSE STATE OF THE ART TOOLS ENABLE FASTER AND MORE EFFICIENT GAME CREATION TO ADVANCE THE GAME INDUSTRY. VOTED UPON BY THE READERS OF GAME DEVELOPER AND GAMA SUTRA, THE FOURTEENTH ANNUAL FRONT LINE AWARDS CELEBRATE THIS YEAR'S BEST TOOLS.

2011 Award Categories and Winners

HALL OF FAME

XNA GAME STUDIO
[MICROSOFT]

MIDDLEWARE

HAVOK PHYSICS
[HAVOK]

ENGINE

UNREAL ENGINE 3
[EPIC GAMES]

PROGRAMMING

LUA

AUDIO

PRO TOOLS
[AVID]

NETWORKING

GAMESPY
[GAMESPY TECHNOLOGY]

ART

3DS MAX
[AUTODESK]

VISIT GDMAG.COM/FRONTLINEAWARDS FOR FULL LIST OF FINALISTS



FIGURE 1 This shot from FORZA 4 highlights how well image based lighting can tie specular, diffuse, and ambient lighting together.



SEE THE LIGHT

AN INTRODUCTION TO IMAGE-BASED LIGHTING

Nostalgia can be a real minefield in the digital age. It's humbling to thumb through the old portfolio and take a critical look at some of your favorites from back in the day. Being proud of what you did in spite of the technical limitations is the tao of the game artist. But let's be honest: some of that stuff looks a little creaky 5, 10, or [gulp] 15 years out.

Some things still hold up, even after a decade—a well-painted texture or a cleanly built model (polycounts aside) can still look sweet. Tech-heavy elements such as shaders, on the other hand, don't age so gracefully. The worst offender is probably lighting, which has seen a quiet revolution over

the last four or five years. The old-fashioned yellow-blue-red-three-point light rig looks pretty dated in the modern era of spherical harmonic lightmaps and real-time ambient occlusion.

In the last couple of years, a new gizmo has started cropping up in the lighting artist's gaffer

box. Image-based lighting, (IBL) has a respectable history in offline computer graphics, but it's just starting to show up in game engines today. Unreal's DX 11 Samaritan demo and CRYISIS 2 both use IBL techniques, and now that they've been battle tested—with such pretty results, as you can see from the accompanying figures there's a high likelihood it's going to show up on an LCD screen near you pretty soon.

As the name implies, IBL uses images (or, more precisely HDR environment maps) to light a scene. The idea was originally pioneered in the '90s by researchers interested in compositing virtual objects into real

photographs. Getting the geometry in place wasn't too hard—but making the rendered images match the ambience of the scene was very tricky.

The secret turned out to be those silver balls you occasionally find sitting on a pillar in someone's garden (they seem to be particularly popular with the folks who buy from *SkyMall*.) By photographing one of these balls in the midst of the scene, you can extract a spherical reflection map. Take the same photograph several times at different exposures, and you can create an HDR map that captures the intensity and color of incoming light from the whole scene.

What distinguishes IBL from conventional environment mapping, is what you do with the information in environment textures. In the familiar reflection map, every point on your surface will be reflected out into the reflection map to find the



FIGURE 2 The key to image based lighting is the ability to control the glossiness of materials. Note how the semi-gloss materials in the middle still show progressively less defined reflections.



FIGURE 3 In the conventionally lit version of this image (top), all shadowed areas receive the same flat lighting. Note how the window frames and doors have no visible relief. With IBL (bottom) even areas in shadow have directional cues—and more variety of shades and colors.

trying it out

/// If your engine doesn't already support IBL, you can try it for free by downloading either free version of either the UDK or CryEngine. There's also the Marmoset Toolbag from Monkey Studios, developers of DARKEST OF DAYS, which is built around an IBL lighting preview engine. Mental Ray supports IBL as well, but the practicalities are different enough that the results don't show you much about how the technique works in real time.

correct color. The classical reflection look up, however, only works for mirror-like surfaces though. This is one of the reasons so many seminal CG creations, like the early Terminators or the water creature in *The Abyss*, were highly reflective. The special sauce in image-based lighting is the way you grab colors out of the environment. If you blur the environment, the chrome-like appearance starts to become more and more like the kind of soft, spherical harmonic lighting you see in most games with modern graphics (see FIGURE 1).

Simply by dialing the precision of the environment lookup, you can produce appearances from classic CG shiny down to realistically varied diffuse lighting with a soft, GI-like feel. Because both the soft lighting and specular highlights are coming from the same images, the final result has a solid, physical feel that's hard to achieve with more conventional techniques.

HDR

» There are two main ingredients to the magic IBL formula. The first is high dynamic range (HDR) support, which you need to get realistically strong highlights and bloom. There's no need for a silver ball anymore—any method of creating an environment cubemap, including hand painting, will do, so long as the cubemap is an HDR image. Lack of HDR support was what kept IBL out of real-time graphics until Shader 3.0 debuted in DX 9. The technique didn't really become practical until DX 10 offered support for 16 bits per channel lighting. Now, with DX 11 offering HDR compression as well as 32 bits per channel lighting, near-future PC hardware and the mythical next-gen consoles will be even better positioned to use image-based lighting.

The second ingredient is finding the right way to get the right level of blurriness in the environment lookup. Fortunately, the details are a problem for the graphics engineers—but it's helpful to know the basics. Doing the blurring entirely in the graphics hardware at runtime can be costly; you need to grab a lot of pixels from

the hardware and then blend them in a shader. For the shinier, more specular end of things, you can get some of the blurring from lower mips. The very soft diffuse lighting, on the other hand, is better calculated offline as a separate pre-blurred texture (see FIGURE 2).

ART SIDE

» Ultimately, the technical side of IBL is just the combination of HDR environment maps and a fast method for doing blurry lookups. The result can be interestingly complex for such a simple and comparatively cheap technique.

The most obvious benefit IBL offers the lighting artist is visual complexity. If you're used to working with a single ambient light value, whether from a lightmap or (ouch!) a fixed shader parameter, you'll find that IBL provides a much richer and more complex ambience. For one thing, it's directional—if your environment has a brightly lit sky, IBL objects will get a nice top lighting that feels like global illumination (see FIGURE 3). IBL also ties rendered objects into the scene in a way that's reminiscent of radiosity, as the colors in the reflection map are used to light your foreground objects.

With all those colors flying around, IBL is more realistic than a boring old ambient coefficient. But more than that, it's just more interesting to look at; it adds more color and interest to areas in shadow that usually get the short shrift. Moreover, because it's an image-based technique, you can get into the environment map textures and edit them for special effects.

For example, you can add color filters in Photoshop to subtly or not-so-subtly bias the colors in your map. This is great for creating moodier color schemes, and it's also useful for pulling out the forms in your models more effectively. Adding a bit of color contrast to different faces of your environment map can make it easier to differentiate the walls of an interior and provide your players some subtle navigational cues at the same time (see FIGURE 4).

More than anything else, IBL shines literally—in specular lighting,

Instead of generic round photo highlights, the environment map produces highlights of any shape and size (note the rectangular highlights from the lit windows in the Unreal DX 11 demo shot). The shaped highlights help bring out the contours in your objects, and moreover, gracefully provide both sharp, mirror-like reflections and softer, more abstract lighting on surfaces such as plastic, enamel, or ceramics. These semi-gloss surfaces frequently seem lifeless in conventional renderings, but the variety of colors and shapes in IBL reflections can help. Finally, since IBL maps provide light from every direction, glancing reflections are a natural side effect. The kind of rim lighting that many games try to fake with a fresnel effect arises naturally from the technique in a way that makes the standard edge glow seem very flaky.

Because specular is so central to the IBL works, a good IBL system should provide per-pixel control over glossiness in addition to typical specular masking. Nothing sells the contrast of different materials better than seeing the varying size and sharpness of highlights as they travel over a surface. That's always the case, even for plain vanilla Phong shading, but it's truer than ever when working with IBL.

THERE'S THE RUB!

>> Of course, veteran Pixel Pusher readers are waiting for the inevitable "of course..." The one where we have to list the downsides to the technological miracle of the month. In the case of IBL, there's really only one important limitation, but unfortunately it's pretty significant. Like a skybox or a reflection map, the lighting in an IBL setup is always infinitely far away. Your IBL light texture might have bright light on one side and a shadow on the other, but there's no way to move your object closer or farther from the light source. The effect will vary by angle but never by distance. This is why IBL may look like radiosity for a given point in space, but it doesn't behave like it—you can't "step into the light, no matter how far you walk.

Depending on your budget, you

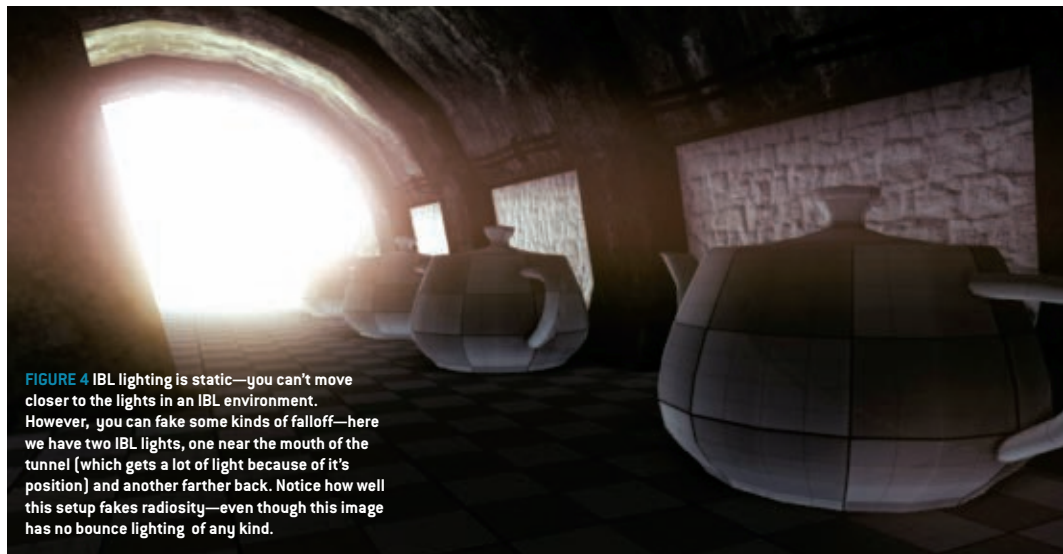


FIGURE 4 IBL lighting is static—you can't move closer to the lights in an IBL environment. However, you can fake some kinds of falloff—here we have two IBL lights, one near the mouth of the tunnel (which gets a lot of light because of its position) and another farther back. Notice how well this setup fakes radiosity—even though this image has no bounce lighting of any kind.

might be able to work around this limitation by blending between different IBL lighting samples. The interpolation is basically a straight crossfade, so it only works when the features in the different IBL textures are lined up nicely. For example, you could not use IBL blending to have a character walk down a corridor with a succession of bright lights, since they would appear to fade up and down rather than to move as the character walked. On the other hand, the trick works nicely if one sample is near the mouth of a cave and another is taken from the back—the cross fade feels more or less like the recession of the bright cave opening. (see [FIGURE 4](#))

There's no doubt that the static nature of IBL makes for an interesting set of problems, though they're also pretty familiar to anybody who's had to deal with things like environment-specific reflection maps. There's a definite memory/fidelity tradeoff, but the results can be very worthwhile. Perhaps just as importantly, seeing familiar models and textures in the richer and more complex ambience of an IBL render makes you start noticing (and resenting) some of the limitations of our more conventional rendering techniques. After natural glancing specular and soft shapely highlights, it's hard to go back to rimlight shaders and big

round Phong spots. It's good to be reminded of the limitation of our standard shading models so we don't become complacent.

Of course, the way our portfolios turn over so quickly, and with all the pretty stuff those art school kids crank out, it's pretty hard to get complacent in any case. 🙄

STEVE THEODORE has been pushing pixels for more than a dozen years. His credits include *MECH COMMANDER*, *HALF-LIFE*, *TEAM FORTRESS*, *COUNTER-STRIKE*, and *HALO 3*. He's been a modeler, animator, and technical artist, as well as a frequent speaker at industry conferences. He's currently the technical art director at Seattle's *Undead Labs*.



MORE THAN ZERO

THE TROUBLE WITH ZERO-SUM MECHANICS IN COMPETITIVE MULTIPLAYER GAMES

Zero-sum game mechanics seem to be the default choice when designing competitive multiplayer games. Most first-person shooters, real-time strategy games, and fighting games, are built around a core my-loss-is-your-gain multiplayer model. However, there are many, many problems with this type of gameplay—particularly for the losing player. Zero-sum mechanics are, at best, a powerful yet dangerous tool and, at worst, a wrongheaded approach to game design that turns away many potential players.

DEFINING A ZERO-SUM GAME MECHANIC

» A zero-sum game is one in which the gains of any one player are balanced out by the losses of all other players, such as winning a pot of chips after a hand of poker. Using strict game theory terminology, many competitive games are not actually zero-sum. Scoring a field goal in football, for example, does not take three points away from the other team.

However, more loosely speaking, the phrase “zero-sum mechanics” means that hurting one’s opponent is as equally valuable as helping oneself. In a typical RTS like *STARCRAFT*, an early-game rush strategy, which aims to destroy the enemy’s economy as soon as possible, is just as viable as a boom strategy, which focuses on building up one’s own economy. It doesn’t matter how low-tech your army is if you can quickly wipe out your opponent’s worker units before his high-tech production kicks in.

Whenever a game rewards the player equally for hindering the enemy as for strengthening herself, the game has a zero-sum mechanic. Most team sports (basketball, soccer, football, etc.) share this characteristic; the defense, which prevents the opposition from scoring, is just as important as the offense, which does the scoring.

Competitive games are firmly rooted in this soil. Fighting games balance protecting your own health and taking away the health of the opponent. Strategy games encourage countering an enemy’s plans and developing your own.

Shooters combine killing as many enemies as possible while also fulfilling some parallel goal, such as capturing a flag or checkpoint.

THE ZERO PROBLEM

» The problem with zero-sum mechanics is that they require a negative experience for someone. No one enjoys watching his character get annihilated by a devastating combo in *STREET FIGHTER*, seeing his buildings crumble in *AGE OF EMPIRES*, or dying and respawning over and over again in *TEAM FORTRESS*. One player’s pleasure results from another player’s pain.

Competitive games do not require that another player must suffer. Indeed, competitive games are even possible without players being able to affect one another at all—consider parallel sports like golf or bowling, for example, or online games with asynchronous leaderboards like *BEJWELED BLITZ* or *BURNOUT PARADISE*. Ultimately, it is the designer that decides how players will interact with each other during play.

The most important distinction is whether a player can lose his current progress or if he can only lose the ability to continue progressing. If the player can lose his current progress, the game mechanics have a zero-sum feel because losing your progress is usually a painful experience (and often a sure route to a loss). In contrast, one of the defining traits of the Eurogame movement (epitomized by games like *Ticket to Ride* and *Settlers of Catan*) is eschewing such direct, zero-sum

player conflict in favor of limited, indirect interaction that will not destroy a player’s progress.

Take worker placement Eurogames, such as *Agricola* and *Caylus*, for example. Players take turns choosing exclusive abilities, and the competition emerges from players jockeying for position to determine who gets to grab the best jobs first. If you know your opponent needs food, you can choose the food job for yourself to seriously damage this opponent’s fortunes. However, this tactic is qualitatively different from actually destroying an enemy’s farms and killing his villagers in *AGE OF EMPIRES*. In the former case, the setback may only be temporary; in the latter, the player suffers a heavy emotional loss and has little chance of recovery. In fact, a player who spends too much time trying to disrupt his opponents in a game like *Agricola* can often dig his own hole as each precious action has significant opportunity costs. In contrast, damaging an opponent early on in an RTS has little downside; wiping out another player’s economy can actually buy valuable time to grow one’s own much larger.

Balancing a RTS game to not reward destroying another player’s economic base as soon as possible is extremely hard. Indeed, RTS games suffer heavily from a dominance of zero-sum mechanics, which encourage the rush. Many players adopt “no-rushing” house rules to manually rebalance the gameplay away from destructive raids and toward building up for the endgame.

Further, many RTS games end

with a whimper instead of a bang because the end goal is usually wiping out the enemy’s forces, which means that the outcome is obvious halfway through the match. In *Ticket to Ride*, during which players race to complete routes before running out of pieces, the dramatic tension is not a consistently rising slope but an arc that rises and then falls. In contrast, the dramatic tension of *STARCRAFT* is an arc which rises and falls, and—unfortunately—the downward side of this arc is simply a sequence of painful events for the loser.

However, zero-sum mechanics need not be endemic to the RTS genre. Consider economic games, like the *Anno* series, or *Railroad Tycoon*, or even *M.U.L.E.*, in which the primary goal is the acquisition of wealth; because the players are in a race to see who grows the fastest, the games need not encourage (or even allow) players to attack one another.

Even military RTS games can use alternative competitive mechanics. *WARCRAFT 3* introduced the “creep”—neutral characters who occupy the central area of skirmish maps and who players race to kill for the rewards and experience points. I see no reason why a new RTS couldn’t take this mechanic a step further and make the game focus solely on killing creeps faster than your opponent.

REMOVING THE NEGATIVES

» Many competitive games solve the zero-sum problem by severely limiting interaction, so that players can only affect each other under certain circumstances. In *MARIO KART*, for example, racers can only



▼▼ One of the defining traits of the Eurogame movement (epitomized by games like *Ticket to Ride* and *Settlers of Catan*) is eschewing such direct, zero-sum player conflict in favor of limited, indirect interaction that will not destroy a player's progress.▼▼

shoot one another after picking up limited-use shells from certain locations; even then, players will only get the most powerful shells if they are trailing in the race. Even in a cutthroat RTS, a player can only attack after first building a barracks, training troops, and finally moving them into position. If you are careful with how you allow your players to interact with each other, you can minimize the negative emotions they experience.

We can see this by comparing two games with similar themes and rules and examining how dramatically different they feel depending on what sort of interaction is allowed. For example, *TRAVIAN* and *EMPIRES & ALLIES* are similar asynchronous strategy games played over months of real time, and are centered on developing your military and attacking your enemies. However, an important difference separates these two games, when you consider what happens when players invade each other's cities.

In *TRAVIAN*, attacks are strictly zero-sum; resources captured by the attacker are taken from the defender's stockpile. In

EMPIRES & ALLIES, however, combat is actually positive-sum; the resources captured by the attacker are conjured from nothing. Furthermore, while units that die in *TRAVIAN* are removed from the game, defending units in *EMPIRES* always stay alive, even after a defeat.

EMPIRES quietly belies players' expectations for combat (that a victory requires a defeat) and this design choice pays off by making the game more accessible and less emotionally draining. In contrast, *TRAVIAN* uses the traditional approach that one player's gain requires another player's loss; accordingly, this design choice creates a nasty world full of brutish players with short tempers.

Many designers instinctively assume that conflict must be zero-sum, but this prejudice may be keeping their games from reaching a larger audience. The emotions players experience during a game are real enough, so a mechanic that requires at least some players to suffer should be used carefully.

ADDING THE POSITIVES

» Sometimes, alternate solutions are blindingly simple. In the board

game *7 Wonders*, players compete along multiple axes by earning victory points for science, civics, buildings, wealth, and military. The default way to implement military in such a game would be to allow players who invest in an army to attack other players' units, buildings, or resources. *7 Wonders*, however, employs a very different approach.

The game is split into three epochs, and at the end of each epoch, players with the largest armies receive positive points while the other players receive negative points. Furthermore, the total point distribution is actually positive-sum, so that losing combat does not hurt a player as much as winning combat helps. Since a strong military cannot interfere with your opponent's ability to progress and win with their preferred axis, the military strategy does not drown out all the others and is appropriately balanced.

Indeed, the spirit of positive-sum gameplay can benefit other aspects of game design. *PUZZLE QUEST*, for example, avoids a manual save system by ensuring every combat is positive-sum; players can

never lose an item during combat and will always gain at least a little gold and experience from each battle. Thus, a player is always better off after combat, whether a win or a loss, so the game can constantly auto-save into a single slot. This feature, which would be hardcore if paired with a traditional zero-sum design, instead removes the need for a load/save system, which can be a barrier to entry for new players, thereby expanding the game's potential reach.

Ultimately, zero-sum mechanics are still a powerful tool for game designers as they can unlock primal emotions. Sometimes, allowing players to destroy each other is exactly what a game needs. However, not all conflict need be zero-sum, especially since that design choice has significant disadvantages. Losers need not suffer so that winners can triumph. 🎮

SOREN JOHNSON was the co-designer of *CIVILIZATION 3* and the lead designer of *CIVILIZATION 4*. He is a member of the GDC Advisory Board, and his thoughts on game design can be found at www.designer-notes.com.

save the date

GDC 13

GAME DEVELOPERS CONFERENCE®

SAN FRANCISCO, CA /// MARCH 25-29, 2013

EXPO DATES: MARCH 27-29, 2013

www.GDCONF.com





ME OF LITTLE FAITH

SURE CHANGE IS CONSTANT...BUT WHAT IF WHAT'S CHANGING IS THE RULES?

WHAT HAPPENS WHEN THE CONSOLE CYCLE AS WE KNOW IT SOON COME TO AN END? That was the theme of one of my favorite talks at this year's GDC—"When The Consoles Die, What Comes Next?" by Ben Cousins. Though I disagreed with the main conclusion, he provoked a line of thinking that has shaken my faith in the current model of platform innovation. Cousins asserted that the "good enough" gaming capabilities provided by phones and tablets were an example of Clayton Christensen's model of "Disruptive Innovation," and would grow to replace all dedicated-purpose gaming platforms (see Figure 1).

/// I agree to a point that this model describes the effect at work in the market. I also agree that mobile devices meet gaming needs for most would-be portable console buyers, so that market is in danger. I disagree that mobile devices will be able to equal even the current generation of home consoles in performance—screenshots of tablet games may look close to console titles, but play those games for a few minutes and you'll see they're limited compared to any triple-A game.

So it seems I don't disagree with the "what" or "how" in Cousins' argument, just the "when." And it was in looking at the question of "when" that I realized Christensen's model breaks, and in the process came to lose faith in some of my fundamental beliefs about the rules that govern how the industry functions. In order to explain myself, I have to first explain a pair of other theories, both coined by Geoffrey Moore.

Moore is known for his "Chasm Theory" (see Figure 2), in which he built upon the Technology Adoption Life Cycle developed by Everett Rogers that explains how technologies come to market. Rogers's life cycle describes how innovators create new technologies, then early adopters pick them up, and then they finally move into the mainstream. Moore observed that there is a "chasm" between early adoption and mainstream acceptance, and many innovations (like the Apple Newton and the Laserdisc) never made the leap.

The second theory is from Moore's book *Inside the Tornado*, where he describes a strategy for crossing the chasm with a deep vertical integration of a technology that "just works" by a single vendor. The triumph of the iPod over other early MP3 players would be a textbook example.

According to Moore, once the mainstream has adopted a technology, the vertically integrated model begins to lose its luster. As consumers demand choice in features, price, performance, colors, and so on, the market for a technology will shift to a horizontally structured open market in which numerous vendors compete. PCs are a classic example, as are automobiles.

In this model, an open market innovates, then a single vertically, integrated solution emerges to push one of these innovations into the mainstream, and then the horizontal vendors re-emerge to spin the now-mainstream product into a few different versions to cater to the demand for variety. One can see variations of this model in the rise of 3D games (from the PC, to the PlayStation, to basically all platforms) or Internet connected games (from the PC, to the Xbox, then back to evolve in the PC again in areas like social).

While my day job lies to the "horizontal-favoring" side of the market, I personally believe the model to be symbiotic—both sides serve equally important roles. Lately though, I'm not as steadfast in my faith. For one thing, as technologies grow more complex, the "flip to horizontal" process takes longer. At the same time, consumers have an insatiable appetite for novelty and the vertically-integrated players seem to be delivering it in waves.

This may seem good for consumers at first, but in the long run an imbalanced symbiotic relationship can hurt everybody. In this case, it means that the closed verticals (console manufacturers) dictate how developers innovate. Imagine trying to convince Sony or Microsoft to try a Free-To-Play game without the model first proving that it works on the PC. Ultimately, I worry that a single vendor innovating—rather than a horizontal market of many vendors—may innovate slowly or take a detour too far in the wrong direction.

Back to Ben Cousins' talk: There are two factors that could disrupt the game industry's current pattern. The first is whether the industry will continue the same innovation back-and-forth between open and closed markets at the same rate we've seen over the past two decades. This will determine the slope of the "sustaining innovation" in Figure 1.

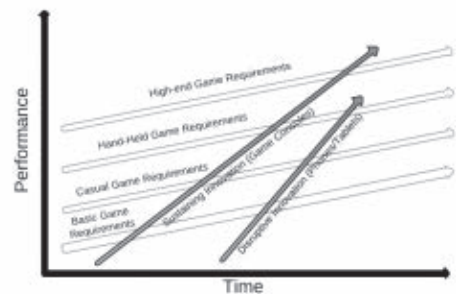


FIGURE 1: Disruptive Innovation and Dedicated Gaming Platforms.

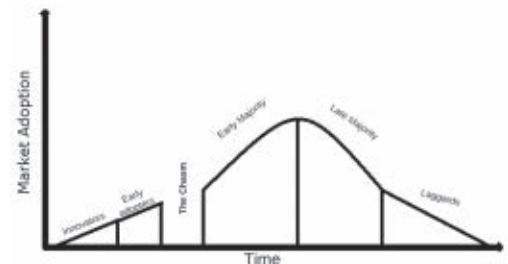


FIGURE 2: Technology Adoption Life Cycle and the "chasm".

The second point is whether developers will be free to build fantastic games that consumers want to buy. This will determine the size of the market at each level of requirements shown in figure 1, as well as whether the "game requirements" line themselves move up in large steps. I believe strongly that developers have the ability to do so, but only if not locked down by rules from the platforms they develop on. I also still believe that the market will continue to evolve through the open/closed symbiosis... though I sometimes have my fingers crossed. 🙏

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.



13 WAYS TO BE A BETTER LEAD PROGRAMMER

TIPS FOR HIRING, TRAINING, AND LEADING YOUR PROGRAMMING TEAM

Long, long ago, back when my coworkers and I were working on an Xbox game, we had a brilliant programmer working on our rendering. Every day, he would come in early, say "Hello", put on his headphones, and would not say another word except Bye on his way out. All was well until one Friday afternoon, when he said "Bye," followed by "I'm leaving the company," and we never saw him again. Nobody had a clue how the code worked, we couldn't even trace his whereabouts, and eventually all his code was thrown out and the renderer was rewritten. (True story.) This is just one of many war stories that could have been avoided with the right foresight. Read on for thirteen tips about hiring, training, and managing other programmers that just might save your game from missing your ship date.

HIRE THE RIGHT PEOPLE

Building an effective team starts with hiring the right people. They need to form a solid team, together with you as the lead.

1: Be on the lookout for young talent.

Years ago, when we were still a startup, we had no choice but to hire recent graduates because we didn't have the funds or the company image to attract senior talent from the industry. This wasn't really a problem, though. When we combined them with a handful of industry veterans, they worked out very well. Young people are often flexible and eager to learn new things, and they adapt and merge easily into a company. Even though we can now attract senior talent with greater ease, we've begun to doubt whether hiring expensive seniors is more beneficial than investing in young talent.

2: Communication skills are just as important as technical skills.

A candidate with great technical skills can't help you if they cannot cooperate with your team members, share your knowledge, or communicate properly with artists and designers. Above anything, I believe game development is about teamwork, and I value open-mindedness, communication skills, and teamwork as much as I value technical expertise.



3: Keep your standards up.

Sometimes you will need to hire a lot of people very quickly. It is tempting to be less picky because you need warm bodies to fill those empty seats. Every hire is intended as a long-term investment. Having people in the team that are not up to par is going to cost the company much more in the long run than it will benefit you in the short term.

4: An interview is not a tribute to your awesomeness.

As an interviewer, you are playing a home game—the interview is

in your company, you're asking the questions, and you're sitting across from someone who wants to impress you enough to offer them a job. Needless to say, you're in a pretty powerful situation, and it may be tempting to ask difficult questions and showcase all your knowledge.

Not only can this be intimidating, it doesn't actually help you find out whether the candidate is competent enough to meet your standards. I personally realized I was asking way too difficult questions during my the interview process. Once I toned down the difficulty of the questions so the candidates felt more at ease, I found that I didn't have to ask a hard question to find out whether a potential hire is competent enough. For example, I started one interview with a simple question ("Can you tell me something about smart pointers?") and the interviewee discussed all different kinds of smart pointers (including pros and cons and implementation details). That was all the information I needed to tell that he was excellent with C++. For all candidates I interviewed, the top talent was spotted within five minutes, and it had very little to do with the difficulty level of the question—it was always about the response.

5: What you see is what you get.

In the beginning when I was conducting interviews with candidates, I thought that peculiar behavior of those candidates during the interviews would probably be the result of being in an interview. After all, people get nervous for interviews and say silly things. I thought those silly things would be much less apparent once they were hired. I was wrong. In practice, I have found the interviews actually do give a good impression of someone's personality.

SHAPE THE TEAM

6: Your company only gets one first impression.

A new employee is like a blank sheet of paper—you only have one opportunity to write on it. Their first weeks are your opportunity to teach employees the company's culture and workflow without prejudice. Also, find the right person from your team to mentor your new hire properly, and you can make sure they get off to a good start with all the information they need to do their job. For example, if you teach a new employee your company's writing style guide on the first day, he'll be able to pick up the new style faster than he would if you sprung it on him a few weeks down the line.

7: Performance reviews are key

Many of the candidates that you

hire will be diamonds in the rough, or they need to adapt to the new company they work for. In my opinion, the most effective way to improve someone's performance is by having periodic performance reviews. This is especially true if the performance score is tied to a raise. This is just basic human behavior: If there are little or no consequences to either good or bad behavior, people will hardly change.

8: When in doubt, don't move forward.

When someone's contract expires, be absolutely sure that you want to continue with this person. Ask yourself: "In a different company, would I hire this person again if he applied?" This question keeps the discussion clean and objective. If the answer is anything less than an emphatic "Yes!" end the working relationship. It's a hard decision, but it's very important to keep the team level up.

GUIDE THE TEAM

9: Keep one step ahead of your team.

"Bring me a stone," a man said to his servant. When the servant returned, the man said: "This is the wrong stone. Bring me another stone." This is a classic example of leading from behind—and it's a good way to waste your team's time. It's your job to show your team your vision and end goal. The specifics of how to reach that end goal, however, need to be figured out together with the team, especially for tasks that take several weeks to complete. I start by making sure I have a good global idea of the task at hand, even if it takes hours, or days, to get there. I need to have a clear big-picture image of how our current task relates to the other big-picture parts of our technology. Next, I try to come up with a basic starting idea for completing the task and throw that idea to the team, and we work together to improve it. That's great! The initial idea is only meant as a starting point for a healthy discussion that keeps the

end goal clear.

The end result is always a superior solution that is supported by the team members and keeps the technology coherent. All you need to do from that point onward is keep regular contact with the team. There may be small adjustments to the idea, but it takes little time and effort because you all started out on the same page. The same can be applied to more territories, for example in the case of code reviews. Code reviews happen when someone already finished a job—a job that he or she takes pride in. Pointing out the mistakes afterward does not feel very rewarding for anyone. Try communicating the important code decisions early on. I have found pair programming to be far superior to code reviews, as I will get to later as well.

10: Think about how you spend your time.

I divide my tasks into three major categories: overhead (meetings and updating plans), coaching, and programming. I personally try to aim roughly for a 40/30/30 division of my time across these. Here's why:

Overhead is important. As a lead programmer, everybody wants a piece of your time. You will be asked to attend every meeting in the company, and your day will become very fragmented, so make sure you're only in the meetings you absolutely have to attend. I have seen many managers running from meeting to meeting, with no time for the coaching and programming/art/design aspects of their job. That's why you need to keep your eye on the ball yourself. Your primary responsibility is (probably) not to sit in meetings all day, and if you're never around to make the calls on important issues, your team members will get frustrated.

You need to be programming. Personally speaking, to be a successful lead programmer, I am convinced that you need to spend a serious amount of time in programming yourself. How

can you give advice to the team without good insight into the work at hand? I see a general pattern that managers lose their credibility once they become too disconnected from the actual work at hand. Besides, technology is shifting so fast, you need to keep up with the knowledge of your team members!

Your team needs your coaching. The effectiveness of your team degrades easily if you're not around to make the important decisions.

11: Aim for collective code ownership.

How often have you had one person who knew everything about a large part of the code? What happens when they get sick or leave? Producers need the freedom to shift tasks among people. For almost any task, two or three people in the team should be able to pick the tasks up—not just for the safety of the project, but also because you should have multiple people thinking with each other about how to approach problems. That's where shared ownership really shines.

Of course, I'm not saying anything new—extreme programming is all about those issues. I am a big fan of pair programming and we use this extensively, particularly when new core systems are being designed. Even when people leave the company, they don't leave a big gap behind. There's also more room for the producers to plan our time, and far more team collaboration. The main problem with pair programming is that you need to find people who are good communicators and open to these kind of practices—which can only work if your workflow matches your hiring policy.


12: Create a safe and open environment.

I had the pleasure of attending Siggraph 2008, where Ed Catmull did the keynote speech. He talked about leadership at Pixar, and it was extremely inspiring. He

talked about creating an open environment, and how important it was to share ideas and make sure that the environment was safe for everyone, so that nobody would hold back. Make sure people are comfortable, and that they are sharing ideas, even if the ideas aren't finished. If people aren't harshly judged by intermediate results, they open up and you get the chance to think together about the problem.

13: Avoid meeting rooms.

Any time a meeting is planned, there's overhead involved. I habitually refill my cup of tea before a meeting, and then I find my notebook and walk to the meeting room, sit down, and wait for people. If there's a meeting planned in 10 minutes, I fill up the time until the meeting starts. So I avoid meeting rooms whenever I can. If possible, I solve issues at someone's desk. I fight to have the daily Scrum standup by the team's desks that people only need to stand up and sit down again. Also, a personal signature of mine is to have a big whiteboard on the wall. Maybe it's because I come from a family of teachers, but somehow there's nothing like standing with a couple of technicians in front of a whiteboard, drawing schemes together. You don't have to meet in the meeting room!

Follow your own lead. Now, there are many roads to Rome, and what I presented here is only one such way. I personally enjoy watching chef Gordon Ramsay on TV. He has perhaps one of the most radically different ways to manage a team. He's not just hierarchical—he's downright rude. But the proof of his effective leadership is his 13 Michelin stars. Whatever the flavor, it's your job to make sure that work is done well and on time while minimizing company risks and making people happy. 

JELLE VANN DER BEEK has been in the game industry for 15 years. He is now lead engine and tools programmer at Vanguard Games and the author of memory analysis tool *Heap Inspector* (www.heapinspector.com)



GAME DEVELOPER MAGAZINE

the best of
postmortems,
product reviews,
and standout
columns

NOW AVAILABLE FOR
DIGITAL DOWNLOAD AND
FOR iOS DEVICES.

SUBSCRIBE TODAY!

GDMAG.COM/SUBSCRIBE

DOWNLOAD THE GAME
DEVELOPER APP
bit.ly/gdmag_ios



gd



FOLLOW GAME DEVELOPER



Com2Korea

BEN SHERMAN MOVES FROM 505 GAMES TO COM2US

PICKING UP YOUR LIFE AND MOVING TO ANOTHER COUNTRY IS A DAUNTING PROSPECT, BUT THAT'S EXACTLY WHAT BEN SHERMAN DID, UPON HIS MOVE FROM BEING A US-BASED PUBLISHER PRODUCER AT 505 GAMES TO A MORE HANDS-ON DESIGNER/PRODUCER HYBRID AT KOREAN SMARTPHONE GAME DEVELOPER COM2US

Brandon Sheffield: What has been the best part of working abroad? The worst?

Ben Sherman: The best part of working abroad has been the "abroad" part. Travel, new experiences, consistently novel or unfamiliar stimulation, and meeting people from wholly different backgrounds has made the last half year very pleasant. You end up missing out on things like live-tweeting SXSW or arguing with your peers over whether video games have finally gone from "brain slime" to "art brain slime," but you gain new ways of attacking old problems, interesting perspectives on creative endeavors, new personality-building challenges in getting people to buy into your vision, surprise trips to Thailand, and soju hangovers.

The worst part of working abroad is a low level hum of alienation that is constantly with you. Korea is a very culturally and ethnically homogenous place, so while Seoul is very cosmopolitan and the people are very polite, and—dare I say—glad to have me around, I've more than once had a child cry at the sight of me. Being The Other is hard, bro.

What would you say the main differences are being a producer in the U.S. and in Korea?

Being an American producer/designer in Korea is a bit like having social super powers. I am impervious to (i.e. mostly oblivious of) certain hierarchical mores, so I get more say than a lot of my Korean peers. Often when it is polite to shut up and accept what a superior wants to do, I keep talking because I have not been brought up to realize that I am not supposed to.

There are large cultural differences between the United States and Korea in expectations for creativity and professionalism.

In America, you are expected to have a loud voice and push for your point of view; always being on the border of politeness is okay so long as you are vehemently arguing that your design is superior to the other guy's. Koreans do not often get into passionate rip-roaring arguments over the minutia of a UI design because they have a much stronger we're-all-in-this-together ethic. Everybody's idea should be incorporated if possible. Design by committee is held up as an ideal, which is exactly the opposite of the American ideal of the autocratic creative auteur.

What suggestions would you have for someone who is looking at game dev in another country?

Expect exactly the same sort of creative challenges and professional challenges, but expect the professional challenges to have a very alien slant on them. You come into the office and one of the artists has a yellow exclamation point over his head. "Sweet," you think to yourself, "leg two of the quest." You right click on his face and he says to you (in Korean, duh) "With the intent of making the product better, I challenge you to criticize my work without being allowed to criticize my work because it would not be professionally courteous." This is not an exaggeration. I have seen this happen. I have seen one developer right-click another developer's face in real life and he is my inspiration for becoming fluent in Korean.

I have found it effective in this situation to physically mock up a prototype showing how one concept, while not the perfect solution, is preferable to another concept. Sometimes translation is difficult. Sometimes there are just huge cultural differences. But having something to play and realizing that it is more fun or easier to use is about as close to robust scientific method as you can get. Anybody looking to do game development where they do not have cultural mastery should learn to program a computer. Without shared language, prototyping is the modern equivalent of painting on a cave wall before the hunt.



who went where

Bullfrog and Lionhead Studios co-founder Peter Molyneux has announced that he will leave Lionhead Studios once FABLE: THE JOURNEY is complete for a new game company called 22 Cans, which was recently founded by former Lionhead CTO Tim Rance.

Phil Harrison has left his post as head of Sony Worldwide Studios to join Microsoft as the corporate vice president of its Interactive Entertainment Business in order to oversee its European game development efforts.

Arjan Brussee, co-founder of KILLZONE developer Guerilla Games, has left the company to join DEAD SPACE developer EA subsidiary Visceral Games.

CITYVILLE lead designer Michael McCormick left Zynga for San Francisco-based social games startup Idle Games. Idle Games was started in 2009 by Playdom co-founder Rick Thompson, and recently raised \$10 million in its second round of funding.

new studios

Bungie co-founder Alex Seropian has started a core-focused mobile studio called Industrial Toys, along with Brent Pease (Bungie, DreamWorks Animation) and Tim Harris (Seven Lights).

TOMB RAIDER veterans Anna Marsh and Sarah van Rompaey founded a new studio called Lady Shotgun Games, and plan to announce an iOS game as their debut title in summer 2012.

EA Tiburon staffers Jerry Phaneuf (former technical art director) and Volga Aksoy (former lead software engineer) have started a two-man development studio aimed at the high-end PC gaming market called PixelFoundry. PixelFoundry's first game is a space-based real-time strategy game titled BLACKSPACE.

FABLE creators Dene and Simon Carter, along with fellow Lionhead Studios members John McCormack, Guillaume Portes, and Jeremie Texier, have formed a new UK-based development studio called Another Place Productions.

GAME DEVELOPER MAGAZINE

the best of postmortems,
product reviews, and
standout columns

GET THE
PRINT+DIGITAL

ACCESS BUNDLE FOR ONLY

\$49.95 /YEAR

+ DIGITAL ACCESS
TO BACK ISSUES

+ EXCLUSIVE
INTERACTIVE EXTRAS

INCLUDES:



PRINT
SUBSCRIPTION



DIGITAL + GAME
DEVELOPER APP



BONUS!



BEST OF
POSTMORTEMS
PRINT ISSUE

SUBSCRIBE TODAY!

GDMAG.COM/SUBSCRIBE



the art & business of making games

take
control
of your
future



www.gamasutra.com





KNOWING A THING, OR TWO

CAREER FRAGMENTATION AT THE TURN OF THE GENERATION

The evolution and reinvention of modern game development constantly reminds me of '80s rock band *Ratt*'s classic lyric: "Round and round, what comes around goes around." We're seeing the game audio world continually fragment and consolidate job titles and disciplines throughout the expansion and contraction of development teams. In an effort to understand some of the ways things have changed, let's take a fresh look at how they've also managed to stay the same.

RENAISSANCE, MAN

» Since the first game programmers and electrical engineers began probing circuit boards in an attempt to re-create realism through sound, the task of game audio has often ended up in the hands of a solitary figure wearing many hats: game designer, programmer, artist, and, finally, musician and sound designer. As the scope of games grew, each discipline found different

old days. As this new world grows, we will see room for people to create new specialties focusing on new ways of engaging the player through their mobile device or social networks. And the cycle will continue.

KNOW YOUR ROLE

» So, is the job market diversifying or consolidating? The answer is yes! Now is the perfect time for anyone

that doesn't cultivate your creative needs.

So, should you focus on some obscure art that holds a gravitational pull over your particular pleasure? You could specialize in creature vocalizations, for example, or script preparation, or physics audio implementation, or vehicle recording, or chiptune composition, or interactive music editing, branching dialogue systems design, voice processing, DSP programming, user interface design for audio tools—just to name a few. I know people working within these specific capacities throughout the industry, but you'll never find these jobs posted. The way you get these jobs is by being the one whose name is synonymous with the subject. To get there, you need to dig deep into the well of knowledge.

You'll never know everything, but it's not about knowing it all. It's about being involved in the conversation and contributing to the outcome of something that you care deeply about and satisfying your inner geek in the process. Find what it is that you care deeply about and start playing a role.

FORKS IN THE ROAD

» What happens if you find yourself at the beginning of the long road ahead and, as *Granddaddy* put it in- "The Group Who Couldn't Say," "The sprinklers that come on

at 3 a.m. sound like crowds of people asking you, 'Are you happy what you're doing?'" How do you find out what you want to focus on when you're busy worried about the path not taken?

The first thing you do, is you do. You do something, anything, and everything that will put you in a position to succeed and fail, and then fail some more until you succeed at achieving some just-out-of-sight goal that will help you further down the road. You'll scrub through forums seeking knowledge, rubbing shoulders with your fellow seekers of game-audio insight. There will be many along the way who will offer their story and regale you with tales of battles long since past and if you're wise, you'll listen and learn from some of the best that have gone before you, because yesterday's triumph often comes back around as tomorrow's solution.

THE SEA

» You may find you're the only person who cares about the audio for a freshly minted game idea, and you're ready to take on the world. It hasn't even dawned on the rest of the team that sound could be anything more than squeaky bleeps and rotund bleeps. You're alone in a sea of potential, with yourself to rely on for everything coming out of the speakers.

You strap on the headset mic, vocalize some gnarly waveforms, edit them into shape, and drop some science at the next online team meet-up. With a little convincing, the programmers are onboard to wire things in dynamically so they react to player input in an orchestrated symphony of skronking interactivity. Your advocacy for audio has boosted your teammates' expectations from a "few well-placed sounds" to a "breathing environment of sonic density."

Before you know it, you're investing all of your time toward creating a style guide and design document for the ever-expanding scope of audio. The content lists are growing daily and you wonder if you'll ever find the time to record half of these sounds or design that steampunk machinery—and let's not get started on the systems you need to build to enable them to play back appropriately. If only you knew someone who could help out who knew a thing or two about building a steampunk soundscape, perhaps.

And so the cycle continues. 🎧
"Look out honey, 'cause I'm using technology." — *Iggy and the Stooges*

DAMIAN KASTBAUER is freelance technical sound design miscreant who can be found expanding on game audio at LostChocolateLab.com and on twitter @LostLab.



ways to further specialize in specific facets of the field. Our industry began to grow a broad range of possible specialties enabling passionate people to dig deep and carve out a career focusing on a single piece of the interactive Fabergé egg that is a video game.

But with the emergence of casual/mobile/social gaming came a renaissance of single visionary developers or small teams who had to apply themselves across multiple disciplines—just like the good (or bad)

entertaining a career (or a career change) in games to reach to the heart of their desires and, to quote Joseph Campbell, "Follow your bliss." There is a place for your passion if you believe in yourself and are willing to work at your dream, whether you want to pursue a particular specialty or learn how to do it all. But it's normal to worry. Maybe you're in school wondering whether there will be a job waiting for you after graduation, or you're stuck working a job

GDC 2012 HITS RECORD ATTENDANCE, ANNOUNCES 2013 DATES, AWARD WINNERS

The 2012 Game Developers Conference hosted a record-breaking 22,500 game industry professionals last week at the Moscone Center in San Francisco, a 17 percent increase in attendance over the previous year's event.

Following the success of the show, organizers have announced that GDC 2013 will return to the Moscone Convention Center in San Francisco from Monday, March 25 to Friday, March 29, 2013, with a call for lecture submissions to open this summer.

GDC 2012 also hosted the 14th Annual Independent Games Festival and the 12th Annual Game Developers Choice Awards (GDCAs) on the evening of March 7th. Montreal-based developer Polytron Corporation earned the Seumus McNally award for Best Independent Game at the IGF and the \$30,000 grand prize with its unique perspective-shifting platformer, FEZ. Meanwhile, at the Game Developers Choice Awards, Bethesda Game Studios's epic fantasy adventure game, THE ELDER SCROLLS V: SKYRIM took home Game of the Year.



BEST STUDENT GAME

WAY
[Carnegie Mellon University, Entertainment Technology Center]

TECHNICAL EXCELLENCE

ANTICHAMBER
[Alexander Bruce]

EXCELLENCE IN DESIGN

SPELUNKY
[Mossmouth]

BEST MOBILE GAME

BEAT SNEAK BANDIT
[Simogo]

EXCELLENCE IN VISUAL ART

DEAR ESTHER
[thechineseroom]

EXCELLENCE IN AUDIO

BOTANICULA
[Amanita Design]

AUDIENCE AWARD

FROZEN SYNAPSE
[Mode 7 Games]

MICROSOFT XBLA AWARD

SUPER T.I.M.E. FORCE
[Capy]

NUOVO AWARD

STORYTELLER
[Daniel Benmergui]

SEUMUS MCNALLY GRAND PRIZE

FEZ
[Polytron]



BEST AUDIO

PORTAL 2
[Valve]

BEST DEBUT

BASTION
[Supergiant Games]

BEST NARRATIVE

PORTAL 2
[Valve]

BEST VISUAL ARTS

UNCHARTED 3
[Naughty Dog]

BEST DOWNLOADABLE GAME

BASTION
[Supergiant Games]

BEST GAME DESIGN

PORTAL 2
[Valve]

BEST TECHNOLOGY

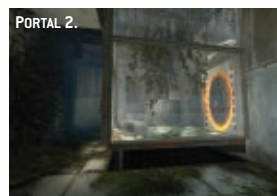
BATTLEFIELD 3
[DICE]

BEST MOBILE HANDHELD GAME

SUPERBROTHERS: SWORD & SWORCERY EP
[Superbrothers and Capy]

INNOVATION AWARD

JOHANN SEBASTIAN JOUST
[Die Gute Fabrik]



2012 PIONEER AWARD

Dave Theurer [MISSILE COMMAND, TEMPEST, and I, ROBOT]

GAME OF THE YEAR

THE ELDER SCROLLS V: SKYRIM
[Bethesda]

LIFETIME ACHIEVEMENT AWARD

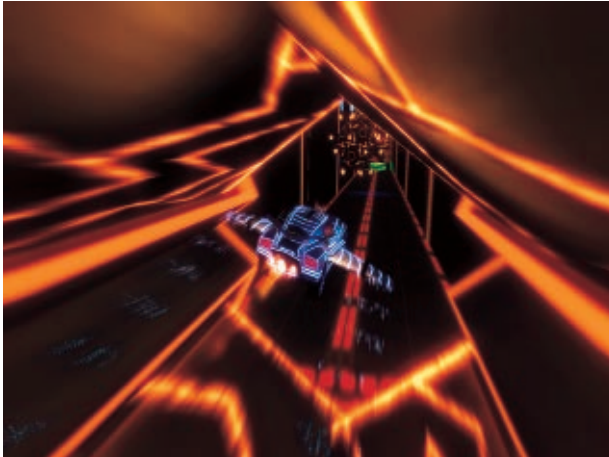
Warren Spector



NITRONIC RUSH

<http://nitronic-rush.com>

NITRONIC RUSH IS A BIT OF AN ODDITY WHEN IT COMES TO DRIVING GAMES. IT'S NOT ABOUT RACING OTHER DRIVERS, BUT SIMPLY ABOUT MAKING IT TO THE FINISH LINE IN ONE PIECE. THE GAME THROWS PLAYERS INTO A VIBRANT, *TRON*-LIKE WORLD WHERE THEY MUST AVOID TRAPS, SPEED BETWEEN OBSTACLES, AND EVEN USE A SET OF DEPLOYABLE WINGS TO FLY ACROSS THE TRACK. THE GAME'S UNIQUE DESIGN AND IMPLEMENTATION EARNED IT AN AWARD AT THIS YEAR'S INDIE GAME CHALLENGE, AND AN HONORABLE MENTION AT IGF 2012. WE CHATTED WITH THE DIGIPEN STUDENT TEAM BEHIND THE PROJECT TO LEARN ABOUT ITS PRODUCTION AND EVENTUAL SUCCESS.



Tom Curtis: *What were some of your biggest influences on the project? I'm guessing Tron: Legacy played a pretty big role?*

Kyle Holdwick (executive producer, gameplay programming, obstacle logic): Overall, we were definitely inspired by a number of '90s arcade racing games like RUSH and HYDRO THUNDER. Visually, we always wanted to go with a futuristic cyber-based style, and we certainly did look at *Tron: Legacy* for reference. Musically, we were influenced by retro game music and the influx of the modern electro house movement.

TC: *The game is surprisingly full-featured for a student project (with achievements, multiple modes, and so forth). What was your strategy for implementing all of these various systems on time?*

Jordan Hemenway (audio director, music composition, sound design, web site development): Scope was definitely a challenge for the team, but we did our best to prioritize between needs and wants. Oftentimes during development it seemed like

there was a monumental list of features waiting to be implemented. Despite this, we made sure the foundation for new features and mechanics was well laid for us to build upon. Thanks to that, we ended up being largely content-driven during the last couple months of development, adding as much music, levels, and polish into the game as we could.

KH: In many ways we took the development of NITRONIC RUSH day by day, allowing our inspirations and mechanics to guide us. This allowed for our design methodology to be very flexible and natural. When we would have a design meeting, we would look at what our playtesters wanted and figure out what we wanted to work on together. This approach allowed for a number of creative decisions to be made by many different team members throughout the entire development cycle of the game. Since our strategy was to be self-driven, motivation was usually high and finishing tasks on time was easier.

TC: *You worked on the game for 17 months, correct? Is that sort of development cycle typical for other student projects at DigiPen?*

JH: Usually a DigiPen game project takes two semesters (fall and spring) with a possible additional semester during the summer to add polish for competitions. Nitronic Rush's development, however, lasted five semesters in total (from May 2010 to November 2011).

We actually started a semester early to build tech for a shared architecture called Superdyne (used in another game Kyle and I worked on called Solstice), and due to teacher encouragement we continued the project through the end of the year. It's definitely atypical for a DigiPen game, but with how ambitious the design was I'm glad we were able to see it to the end.

TC: *What were the biggest challenges during development?*

Andy Kibler (game designer, level design): Physics is by far

one of the hardest elements of a racing game. The car physics have to be solid in order for anything to happen on-screen. Jason Nollan did a great job, and it definitely shows.

JH: In terms of the overall team, I agree that the physics and overall car controls were probably the most challenging problems we had to solve. Otherwise, I'd say that having a polished beginning and ending to every piece of the project was definitely a challenge. Oftentimes we'd get stuck working on the core part of a level, menu, or story mode arc while leaving the loose ends unpolished until later on.

TC: *If you were to go back and do one thing differently on this project, what would it be?*

KH: If we were to go back and change some things, we would definitely add both competitive and cooperative multiplayer to the game. We would add more vehicles with a broader range of abilities. We would also add more levels with more of an emphasis on wall riding and flying.

AK: I should have made more levels for the game. I know I made quite a few, but I know I could have made more!

JH: Fleshing out the story quite a bit more would have been really interesting. Everyone had visions of a fairly grandiose story early on in development, but it wasn't until the very end that we created the cutscenes and started laying out the actual story within the game. If we had more time, it would have been great to spend some proper time creating a more elaborate story arc throughout the game's story mode.



Developer/Publisher: Team Nitronic
Release Date: 11/11/11
Platform: Windows PC
Number of Developers: 11
Length of Development: 17 months
Budget: \$0
Lines of Code: 132,551
Fun fact:
 Two members of Team Nitronic are actually twin brothers, Andrew Nollan and Jason Nollan.



FULL SAIL UNIVERSITY®

THE POWER TO CREATE



DEGREE PROGRAMS IN:

Game Art

Game Development

Game Design

© 2012 Full Sail, LLC

3300 University Boulevard • Winter Park, FL

Financial aid available for those who qualify • Career development assistance • Accredited University, ACCSC

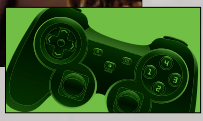
To view detailed information regarding tuition, student outcomes, and related statistics, please visit fullsail.edu/outcomes-and-statistics.



800.226.7625

fullsail.edu

Start Living The Dream!



A.S. Degree in Game Production

Learn to create the future of games with an Associate's Degree in Game Production from The Los Angeles Film School. Your education will give you the knowledge to view every piece of a game artistically, analyze its programming and learn the tools & techniques to create the worlds we play every day.

Learn the Science of Game Production and have the following skill set:

- Create Game Art
- Design Characters, Objects & Environments
- Develop Game Programming Skills
- Discover the Art of Storytelling

Learn from The Los Angeles Film School's experienced industry professionals.

- On-site housing coordinator
- Accredited College, ACCSC, VA-Approved
- Financial Aid & Military Education Benefits (including BAH) available to those who qualify

Scan for more Information



THE
LOS ANGELES[®]
FILM SCHOOL
ANIMATION + AUDIO + FILM + GAMES

Create Your Future Today. Call:

800.406.7485

www.DesignLAFilm.com



*Length of program and start dates are dependent on course of study and degree option. For more information on our programs and their outcomes visit www.lafilm.edu/disclosures. ©2011 The Los Angeles Film School. All rights reserved. The term "The Los Angeles Film School" and The Los Angeles Film School logo are either service marks or registered service marks of The Los Angeles Film School. Accredited by ACCSC

ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
BLIZZARD ENTERTAINMENT	26 & 27	LOS ANGELES FILM SCHOOL	55
E3 EXPO	34	MAGIC PIXEL GAMES.....	14
EPIC GAMES	6	ONLIVE INC	C2
FULL SAIL REAL WORLD EDUCATION	54	RAD GAME TOOLS	C4
INTEL CORPORATION.....	3	VANCOUVER FILM SCHOOL	37

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.



QUARTERLY REPORT

PURSUANT TO SECTION 13 OR 15(D) OF THE SECURITIES EXCHANGE ACT OF 1934

CAUTIONARY STATEMENT

This quarterly report on Form 10-Q contains, or incorporates by reference, certain forward-looking statements within the meaning of the Private Securities Litigation Reform Act of 1995. Such statements consist of any statement other than a recitation of historical fact and include, but are not limited to....

Pssst! Hey! Hey, you still with me? You are? Good! Now that we got all the artists, designers, and other non businesspeople to stop reading this, we can get started for real. Welcome to your secret back-page column, video-game industry executive! And our actual column title is:

KEEPING YOUR DEVS IN LINE

» Today, we're going to discuss developers! You know—those weird, difficult-to-control geeks and nerds who actually create the games that you sell in order to make money and have a business.

First of all, let's take a moment to ponder those engines of profit for our companies and agree with each other that they're all really flippin' annoying! Always complaining about your optional mandatory crunch time, making noise about your brilliant marketing spend decisions, and totally not meeting the perfectly reasonable dates and budgets you keep setting for them.

It's a shame we need them around, but much like the mold cultures that we tolerate in order to eventually have fine wines and aromatic cheeses, so too are game developers a slightly icky but ultimately necessary part of what it takes to rake in cash from your gamer audience.

That said, there are a lot of great techniques you can use to keep your pesky crews happily churning away on stuff to make you rich.

PROMISE THEM REWARDS

» This is an easy one: Everyone will work hard if they're promised a beautiful, shining light at the end of the tunnel. If you can spare the 10 or 15 minutes it takes to talk to your developers once a year or so, remind them how amazing and wonderful things are going to be after the game (whatever game it is they're making at the moment) is done.

Of course, devs like to think they're pretty smart. They can do math and use computers and so forth, so I'm sure at least one of them will think they saw through the vagueness of your words. What do you do if one of your people asks for concrete information on what those rewards will be?

This leads me to...

GAMIFY YOUR GAME DEVELOPERS

» You pay your game designers to create elaborate stages of badges, achievements, and rewards that unlock as players stay loyal to your branded experiences. As it turns out, you can do the very same thing to your own employees. Make 'em stay at your company to unlock tiers of revenue sharing, stock options, and access to the executive bathroom (not yours, of course—keep that one to yourself).

For as smart as these developers think they are, somehow all the techniques of gamification work just as well on them as they do on your customers. Do what they do: Create



a reward schedule with smaller prizes (T-shirts, cupcakes, or plastic goblets) more frequently and larger prizes (their name on a piece of metal somewhere in the studio) less frequently. Do this enough and the real prize everyone thought they were going for in the first place will eventually fall by the wayside, fading into a distant memory of a dream.

ANSWER THEIR QUESTIONS (NOT REALLY)

» To create the impression that you are a caring executive who listens to the concerns of the team, you might open the floor of your company meeting to questions once every couple of years or so. Most people will ask

easy ones—"When's the next company picnic, because I can't wait to hit Jared in the face with a volleyball again, ha, ha, ha," and the like. But once again, some developers have to be unnecessarily combative and unsportsmanlike and will try to throw curveballs at you.

If some smartass works up the nerve to ask a difficult question the next time you do a company all-hands, it pays to be ready with a good answer. Here's a sample question:

"It just feels like we make the same game over and over. What about new IP or incubating cool new game concepts?"

And an answer: "We continually look at our offerings in relation to the market to identify and exploit what we believe could be viable new initiatives to create entertaining new value propositions for our customers. Since costs for creating games have become so high in recent times, our process of due diligence on new intellectual properties must be consistently balanced vis-à-vis our blah blah blah"—you get the idea. You're an executive—I bet you can spout this all day without burning a calorie.

Another question: "Does our company have a coherent strategy? Because from where I'm sitting, I don't see one."

This is where you smile at the guy who asked the question and IMPRINT FACE/VOICE PATTERN INTO SUBDURAL IMPLANT. TRANSMIT ENCRYPTED ID OF SOFT ASSET TO OVERWATCH UNITS. MARK TARGET FOR TERMINATION PROTOCOL ALPHA-7.

Oops, looks like it's about time to wrap up... See you next time!

IN CONCLUSION

» Refer to the section titled Management's Discussion and Analysis of Financial Condition and Results of Operations contained in Part II, Item 7 of our annual report on Form 10-K for the year ended December 31, 2011, for a more complete discussion of our critical accounting policies and estimates. ^(f)

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



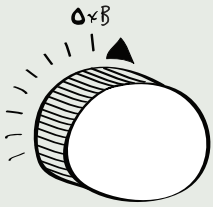
GAME DEVELOPERS CONFERENCE™ EUROPE

COLOGNE, GERMANY
AUGUST 13-15, 2012

2012

WWW.GDCEUROPE.COM

TURN UP THE



WICKED

AUDIO IN YOUR GAME WITH

MILES 9

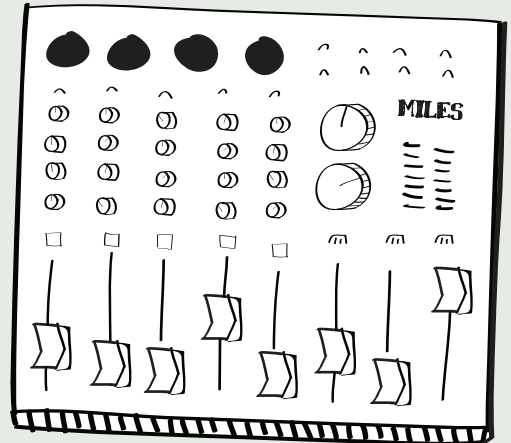
BRAND NEW!



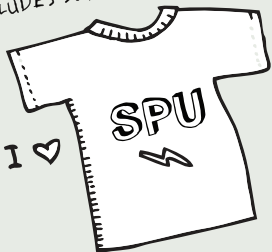
Our multichannel sound system features cool, new

high-level audio tools

+ the world's **FASTEST**



INCLUDES SUPPORT FOR THE PS3



MP3 and Ogg DECODERS.

USING MILES 9 NOT ONLY MAKES YOU

WANT TO TURN IT UP, IT MAKES YOU rad!



www.radgametools.com
(425) 893-4300