

# INDIANA JONES ADVENTURE WORLD

THE LEADING GAME INDUSTRY MAGAZINE VOL. 19 NO. 5  
MAY 2012 INSIDE: REINVENTING MMORPG COMBAT  
HOW TO PLAN WORLDWIDE GAME JAM IN THREE WEEKS

# gd

**technology survey 2012**

WHAT SOCIAL + MOBILE DEVS WANT

**deep dive into TERA**

free-targeting combat system

GAME DEVELOPER MAGAZINE





## The best ideas evolve.

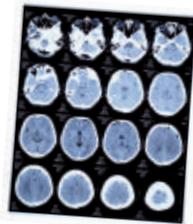
Great ideas don't just happen.  
They evolve. Your own development  
teams think and work fast.

**Don't miss a breakthrough.**  
**Version *everything* with Perforce.**

Software and firmware. Digital assets  
and games. Websites and documents.  
More than 5,500 organizations and  
400,000 users trust Perforce for  
enterprise version management.

**Try it now. Download the free  
20-user, non-expiring Perforce Server  
from [perforce.com/try20](https://perforce.com/try20)**

Or request an evaluation license  
for any number of users.





INDIAN JONES ADVENTURE WORLD

gd

## CONTENTS.0512

VOLUME 19 NUMBER 5

### POST MORTEM

**24 INDIANA JONES ADVENTURE WORLD**  
How do you make a social game that can bring elements from THE LEGEND OF ZELDA: OCARINA OF TIME or TOMB RAIDER to a relatively casual audience? Answer: lots of testing, lots of iterating, and lots of *Indiana Jones*. Zynga Boston lead designer Seth Sivak shows us what went right and wrong with Zynga's casual-core-bridging INDIANA JONES ADVENTURE WORLD. *By Seth Sivak*

### FEATURES

- 7 2012 SOCIAL/MOBILE TECHNOLOGY SURVEY**  
Our big technology survey for 2012 focuses on the engines, tools, middleware, platforms, and services that mobile and social game developers are working with—as well as the tech they wish they were working with. *By Mark Deloura*
- 16 TERA: EVOLVING MMORPG COMBAT**  
If your MMORPG's players are spending more time staring at cooldown meters than the monsters they're fighting, you're doing it wrong. Bluehole Studios's Seungmo Koo explains how they pulled off TERA's truly real-time "free-targeting" combat system without breaking their servers. *By Seungmo Koo*
- 32 WHAT WOULD MOLYDEUX? GAME JAM POSTMORTEM**  
During a single weekend, over 900 developers across 30-plus cities worldwide made more than 300 games as part of the "What Would Molydeux?" game jam. Read about what went right and what went wrong with organizing a massively popular game jam in just under three weeks. *By Brandon Sheffield*

GAME DEVELOPER MAGAZINE

### DEPARTMENTS

- 2 GAME PLAN** *By Brandon Sheffield* [EDITORIAL]  
Make Games for Yourself
- 4 HEADS UP DISPLAY** *By Staff* [NEWS]  
Old fan-favorite franchises get revived on Kickstarter, and Game Developer staff changes
- 35 TOOL BOX** *By Matt Link* [REVIEW]  
Pixologic's ZBrush 4R3
- 39 THE INNER PRODUCT** *By Slawomir Nikiel* [PROGRAMMING]  
Cells for Cell Phones
- 44 DESIGN OF THE TIMES** *By Jason VandenBerghe* [DESIGN]  
The Five Domains of Play
- 47 GDC NEWS** *By Staff* [NEWS]  
GDC Vault debuts classic videos, GDC 2012 lectures
- 48 PIXEL PUSHER** *By Steve Theodore* [ART]  
Tech Tao for Tiny Teams
- 50 GOOD JOB** *By Patrick Miller* [CAREER]  
Q&A with Matt Sughrue, who went where, and new studios
- 51 THE BUSINESS** *By David Ederly* [BUSINESS]  
The Magic of Free-to-Play
- 52 AURAL FIXATION** *By Damian Kastbauer* [SOUND]  
Vroom Vroom
- 53 EDUCATED PLAY** *By Patrick Miller* [EDUCATION]  
ONE AND ONE STORY
- 56 ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]  
Secret Info Inside Here



# MAKE GAMES FOR YOURSELF

## ...AND NOBODY ELSE

A lot of people say that if you want to make a popular game, you need to listen to focus groups, carefully monitor metrics, and focus on the mainstream. I say: bullshit. Scaling small and being true to yourself can win you free marketing and make you rich, if you do it right. Even better, your games will be way more interesting!

### PERSONAL POWER

» Let's look at some examples. **SUPERBROTHERS: SWORD & SWORCERY** EP for iOS is single-player, has a singular aesthetic, and launched at \$4.99, which is expensive for the App Store. It was a recipe for commercial failure in a world where the birds are angry, and the top games are free. Still, the game wound up selling 300,000 units in the first 6 months, because the game was absolutely true to its creators' vision. They made it for themselves.

**MINECRAFT** is another easy one. Notch made the game that he wanted to play. Granted, he knew he was making a sandbox for other people to fool around in, but he didn't do any market research first. In making **MINECRAFT** for himself, Notch created a massive self-perpetuating hype machine, because when people like your game, with the proliferation of social media, they can't help but talk about it.

Something that resonates with a small group of people will expand to their friends, and then their friends, and eventually to their parents and grandparents, who would never have otherwise thought of playing one of these games.

The reason this works so well is because people want to identify with cool things, and they want other people to think that they're cool for thinking that this "cool thing" is cool. This little corner of the world they've discovered is something they now identify with, and they'll want their friends to like it too. When they share it around,

they've already put the weight of their appreciation behind it.

### OKAY, LET'S DO IT

» How do you emulate these successes? Contrary to popular belief, you shouldn't emulate the actual products. Instead, pay attention to the thought process that goes into making them, beginning with the initial idea.

There is something that you like more than anyone else you know does. Maybe it's Apple II-era platformers. Maybe it's fractals. Maybe it's dubstep, god forbid. Find it, and dive right into it.

What qualifies as a niche, then? "Sports" is too vague. The Olympics gets a bit closer, but if you take, say, the Hurdle event in isolation, you're starting to get somewhere. Now you need to find a visual or gameplay hook that really appeals to you (and hasn't been done to death).

A good example of this is **OWOP**, which was a massively popular Hurdling game for browsers. It had stupidly difficult controls, but was hilarious to watch in action, so people played and talked about it religiously. The game has since gone on to App Store success, and is a great example of a good niche game.

Once you've established your niche and tone of gameplay, determine the targets you want to hit, and never deviate from them. If the mandate is "everything blows up," then make everything blow up, even your UI. Rules like this can help you scale small. Throw out everything that doesn't fit your vision.

You may worry that people won't latch on to your idea. But none of us is unique, as much as we might like to think so. There's almost certainly someone else out there that likes the things you do. If you make a game for yourself, you're also making it for them. Nobody expects you to make a game that targets their weird special interest, so if yours matches theirs, they will sing you

praises to the ends of the earth.

Nathan Vella, president of **SWORD & SWORCERY** developer Capy Games said this quite well. He said, "I believe that when you're targeting everyone, you're really targeting no-one. You're not making it for anyone specific, so your target group is no-one."

People can feel when a product is genuine, and there's nothing more genuine than something you've made for yourself. That feeling of "I can't believe someone made this" is what gets you instant success on aggregator communities like Reddit, which are huge drivers of content.

Your method of delivery is important too, though. If your game is hard to find, none of what I just said applies. Consider **BLOODYCHECKERS**, which is an Xbox Live Indie Game. Players explore a massive first person dungeon, with loot, items, and experience points, as they battle the denizens of a haunted castle—all by playing a bizarre version of checkers. If this game were on Steam, the creator would be a millionaire. You have to go where the people are.

If you're developing a game for yourself, you can make something smaller for less money and only take a minimal risk. But the payoff can be huge. You might think this doesn't apply to you if you're working on a big team, and you might be partially right. But the principle of digging deep can be applied to one or two features just as nicely. If your open world game has a really deep crafting system, for example, someone out there will play it just for that.

And who says you should be anonymously toiling away on that big, bloated team anyway? If you have an idea, just get out there and make it. So find your passion, see it through, and don't let the bastards get you down. I'm taking my own advice, and I'll live or die by it.

—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](mailto:gamedeveloper@halldata.com)  
[www.gdmag.com/contactus](http://www.gdmag.com/contactus)

### EDITORIAL

#### PUBLISHER

Simon Carless e: [scarless@gdmag.com](mailto:scarless@gdmag.com)

#### EDITOR-IN-CHIEF

Brandon Sheffield e: [bsheffield@gdmag.com](mailto:bsheffield@gdmag.com)

#### EDITOR

Patrick Miller e: [pmiller@gdmag.com](mailto:pmiller@gdmag.com)

#### MANAGER, PRODUCTION

Dan Mallory e: [dmallory@gdmag.com](mailto:dmallory@gdmag.com)

#### ART DIRECTOR

Joseph Mitch e: [jmitch@gdmag.com](mailto:jmitch@gdmag.com)

#### CONTRIBUTING WRITERS

Seungmo Koo, Mark Deloura, Seth Sivak, Matt Link, Slawomir Nikiel, Steve Theodore, Jason Vandenbergh, Damian Kastbauer, David Ederly, Matthew Burns

#### 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](mailto:amurawski@ubm.com)  
t: 415.947.6227

#### MEDIA ACCOUNT MANAGER

Jennifer Sulik e: [jennifer.sulik@ubm.com](mailto:jennifer.sulik@ubm.com)  
t: 415.947.6227

#### GLOBAL ACCOUNT MANAGER, RECRUITMENT

Gina Gross e: [ggross@ubm.com](mailto:ggross@ubm.com)  
t: 415.947.6241

#### GLOBAL ACCOUNT MANAGER, EDUCATION

Rafael Vallin e: [rvallin@ubm.com](mailto:rvallin@ubm.com)  
t: 415.947.6223

### ADVERTISING PRODUCTION

#### PRODUCTION MANAGER

Pete C. Scibilia e: [peter.scibilia@ubm.com](mailto:peter.scibilia@ubm.com)  
t: 516-562-5134

### REPRINTS

WRIGHT'S MEDIA  
Jason Pampell e: [jpampell@wrightsmedia.com](mailto:jpampell@wrightsmedia.com)  
t: 877-652-5295

### AUDIENCE DEVELOPMENT

#### AUDIENCE DEVELOPMENT MANAGER

Nancy Grant e: [nancy.grant@ubm.com](mailto:nancy.grant@ubm.com)

#### LIST RENTAL

Peter Candito  
Specialist Marketing Services  
t: 631-787-3008 x 3020  
e: [petercan@SMS-Inc.com](mailto:petercan@SMS-Inc.com)  
[ubm.sms-inc.com](http://ubm.sms-inc.com)





# MAGIC PIXEL GAMES

We craft original games.

We love what we do and we want your help.

## WE'RE HIRING!



ENGINEERS and ARTISTS

Located in Los Angeles, CA

VISIT US AT [WWW.MAGICPIXELGAMES.COM](http://WWW.MAGICPIXELGAMES.COM)

©2012 MAGIC PIXEL GAMES, ALL RIGHTS RESERVED



# OLD IS NEW AGAIN

DEVELOPERS USE CROWDFUNDING TO FUND FAN-FAVORITE FRANCHISE REVIVALS

For many players and developers, the biggest news to come out of the industry over the last few months hasn't been about a new studio or a major triple-A title, but about a series of plucky developers hoping to bring back some of their favorite long-forgotten IPs. After Double Fine Productions pulled in \$3,336,471 from Kickstarter supporters for a new adventure game, several other developers decided to try their hands at getting the crowd to fund their favorite pet projects. Here are some of the properties that might be revived.



LEISURE SUIT LARRY

## LEISURE SUIT LARRY IN THE LAND OF THE LOUNGE LIZARDS: 25TH ANNIVERSARY EDITION

[www.kickstarter.com/projects/1451923705/make-leisure-suit-larry-come-again](http://www.kickstarter.com/projects/1451923705/make-leisure-suit-larry-come-again)

### WHO IS MAKING IT:

Replay Games, plus several members of the original Leisure Suit Larry team, including series creator Al Lowe, Josh Mandel, Sabine Duvall, and Leslie Balfour.

### TOP REWARD:

For \$50,000+, you can choose either to have your name replace Sierra Entertainment founder

Ken William's name for the in-game password "Ken Sent Me" or hang out with Al Lowe and writer Josh Mandel for a VIP weekend in Las Vegas.



## SHERLOCK HOLMES: CONSULTING DETECTIVE

[www.kickstarter.com/projects/1920171553/sherlock-holmes-consulting-detective-adventure-mys](http://www.kickstarter.com/projects/1920171553/sherlock-holmes-consulting-detective-adventure-mys)

### WHO IS MAKING IT:

Zojo, a mobile game dev

studio headed by David Marsh (former SHERLOCK HOLMES: CONSULTING DETECTIVE developer).

### TOP REWARD:

For \$600, you can write an article (in your best Victorian prose) for the in-game newspaper, and you'll receive a credit as a game character and writer. You'll also receive an original SHERLOCK HOLMES: CONSULTING DETECTIVE board game and WEST END ADVENTURES expansion game, as well as the CONSULTING DETECTIVE board game.

## SHADOWRUN RETURNS

[www.kickstarter.com/projects/1613260297/](http://www.kickstarter.com/projects/1613260297/)

### shadowrun-returns

### WHO IS MAKING IT:

Harebrained Schemes LLC: Jordan Weisman (SHADOWRUN creator), FASA Shadowrun tabletop developers Mitch Gitelman and Mike Mulvihill, and several former SHADOWRUN authors and developers (Michael Stackpole, Mike Mulvihill, Tom Dowd, Malik Toms, Mel Odom, Jason Hardy, Stephen Kenson).

### TOP REWARD:

For \$10,000+, you will get your own personalized Doc-Wagon card, a custom in-game player character built in your image, and Mike Mulvihill will come to your town and

run a tabletop game of SHADOWRUN for you and five friends (with snacks).

## WASTELAND 2

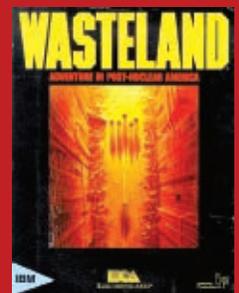
[www.kickstarter.com/projects/inxile/wasteland-2](http://www.kickstarter.com/projects/inxile/wasteland-2)

### WHO IS MAKING IT:

Brian Fargo (executive producer for WASTELAND and FALLOUT), original WASTELAND designers Alan Pavlish, Michael Stackpole, and Ken St. Andre, music by Mark Morgan (FALLOUT 1+2 composer), story by Jason Anderson (FALLOUT cocreator), and concept art by Andree Wallin.

### TOP REWARD:

Donate \$10,000+, and you'll get an invite to an exclusive private party with the WASTELAND 2 team, an in-game shrine in your honor, and 50 copies of the game, among other things.



Miller (left) and Mallory.

*Game Developer* magazine is going through some changes. While the vision will remain largely the same, the staff is shifting rather dramatically.

Brandon Sheffield, editor-in-chief of the magazine, will be moving to part-time, after over seven years with *Game Developer*. He will remain editor-in-chief for the next several months to guide the vision of the magazine, but is departing in a larger sense to spend more time on game development and writing projects of his own through his new company Necrosoft Games ([www.necrosoftgames.com](http://www.necrosoftgames.com)), among other vehicles.

"I've been working on games on the side for the last six years or so," Sheffield said, "but after a large project got canceled, I figured it was time for me to try to tackle this on my own, full time. I don't have a huge safety net, but I needed more creative space, and to tackle a wider variety of projects, not just in games. At the same time, I couldn't just leave the magazine outright, so I will be helping everyone during the transition."

To keep the magazine running smoothly, *Game Developer* has hired new editor Patrick Miller, who was previously an associate editor for *PCWorld* magazine. "I hired Patrick because he's got a love for the game industry and a fire in his belly!" said Sheffield. "The fact that he came from the tech side of journalism and not the consumer side was another big plus, and having also worked with him as an editor on my personal game site, Insert Credit, I'm confident that he can move the magazine in a good direction."

Lastly, production editor Jade Kraus has moved on to new things in Austin, Texas. Her shoes will be filled by Dan Mallory, former senior production coordinator for publications like *PC Gamer*, *Maximum PC*, *Official Xbox Magazine*, and *PlayStation: The Official Magazine*. Mallory will be *Game Developer's* new manager of production.

"I knew Dan was the right guy for the job pretty quickly," Sheffield said, "He's got an extensive magazine background and has worked on a number of game publications. We knew he could hit the ground running and fit into our pipeline without missing a beat."

"*Game Developer* recently revamped its subscription model, has a new site, and an iPad app," Sheffield adds. "I feel confident that I can slowly exit the magazine on a high note. I hope everyone will continue to support it!"

All the editors can still be contacted collectively at [editors@gdmag.com](mailto:editors@gdmag.com). 

**GAME DEVELOPER EDITOR-IN-CHIEF BRANDON SHEFFIELD  
ANNOUNCES GRADUAL DEPARTURE, NEW HIRES**

# CHANGING OF THE GUARD



## 2K GAMES BREATHES NEW LIFE INTO XCOM WITH UNREAL ENGINE 3 TECHNOLOGY

Nearly two decades after the original *XCOM* captivated PC gamers around the world, publisher 2K Games and developer Firaxis plan to introduce the critically acclaimed franchise to an entirely new generation. According to Jake Solomon, lead designer on the new title, game industry legend Sid Meier is working closely with the team to bring this unique blend of tactical strategy and action to life, which marks the first time Meier's studio is leveraging Epic Games' Unreal Engine 3 technology.

"Honestly, it was a thrill to work with Unreal Engine 3," said Solomon. "In-house, we've always used our own tech because of the games we've made before. This was the first game where it made sense to look at a licensed engine. Once we explored the tech and saw what it was capable of, it has just been such a blessing."

Set in a retro-future 1960s Earth, *XCOM: Enemy Unknown* is a turn-based strategy game that pits the player and their troops against an alien invasion of Sectoids. Every decision a player makes is crucial during gameplay, as once a soldier dies they cannot be revived. In addition, the player's research undertaken during the course of the game has a direct impact on repelling the aliens—and there is a real possibility of the aliens succeeding at the game's end. To achieve a sense of life-or-death consequences, Firaxis designed a series of cinematics to add another layer of storytelling to the game.

"We used Matinee pretty heavily in *XCOM*," said Solomon. "That's our foundation for all of the cinematics you see in the game. UE3 was essential in allowing our artists to create content in a mature tool chain like

that. Our programmers were also able to tap into this tech; it certainly helped with the development of three SKUs across PC, PS3 and 360. The ability to build off of a mature engine like UE3 has made it pretty easy."

Solomon said that due to the gameplay style, his team wasn't able to prototype in a day like many shooter teams, but the process was still faster than if they had grown their own tech. Firaxis was also able to seek guidance from other internal game studios.

"So many of the people within our publishing label have UE3 experience, so we have resources within our label that we can reach out to," said Solomon. "Industry-wide, you can't throw a rock without hitting somebody who has UE3 experience. We also used the Unreal Developers Network, which has been a huge help."

From a visual standpoint, the new game taps into the full capabilities of Unreal Engine 3. To achieve the look and feel they wanted, Firaxis created a global landscape for players to engage in, complete with recognizable landmarks. Solomon said that all of the maps were hand-built, including the more regionalized maps, and these encompass both outdoor and indoor maps. Additionally, the player can play in urban areas as well as wilderness areas, where they shoot down UFOs.

"From a technical standpoint, I'm most proud of our destructible environments," said Solomon. "That's something that you still don't see a lot of. But it's really fun to use in our game, since it's a tactile experience. For example, blowing out a wall completely changes the tactics involved in a level and this creates a dynamic battlefield to play in."

Also figuring into this gameplay experience are new aliens conjured by the Unreal Engine 3 toolset from the imagination of Firaxis. Solomon said enemies like the Berserker can literally run through walls. This opens up interesting immersion gameplay, coupling varying

Sectoids with weapons that can change the playing field in an instant. Solomon said UE3 enabled his team to create complex levels and craft intelligent enemies that fluidly run on AI that constantly adjusts to the dynamic surroundings.

"I'm extremely happy with the new 3D base that we've created in this game," said Solomon. "Being able to show all that level of detail when you're zooming in to see the different soldiers and all the units, that's something that my team just did an awesome job with, technically."

And that level of detail, and deep gameplay experience, was brought to life using Unreal Engine 3. Soldiers new and old will be able to engage in battle against a challenging alien invasion in this new take on squad-based tactical action in *XCOM: Enemy Unknown*.

[WWW.UNREAL.COM](http://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 five times along with entry into the Hall of Fame. UE3 has won three 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 on Twitter.

### UPCOMING EPIC ATTENDED EVENTS

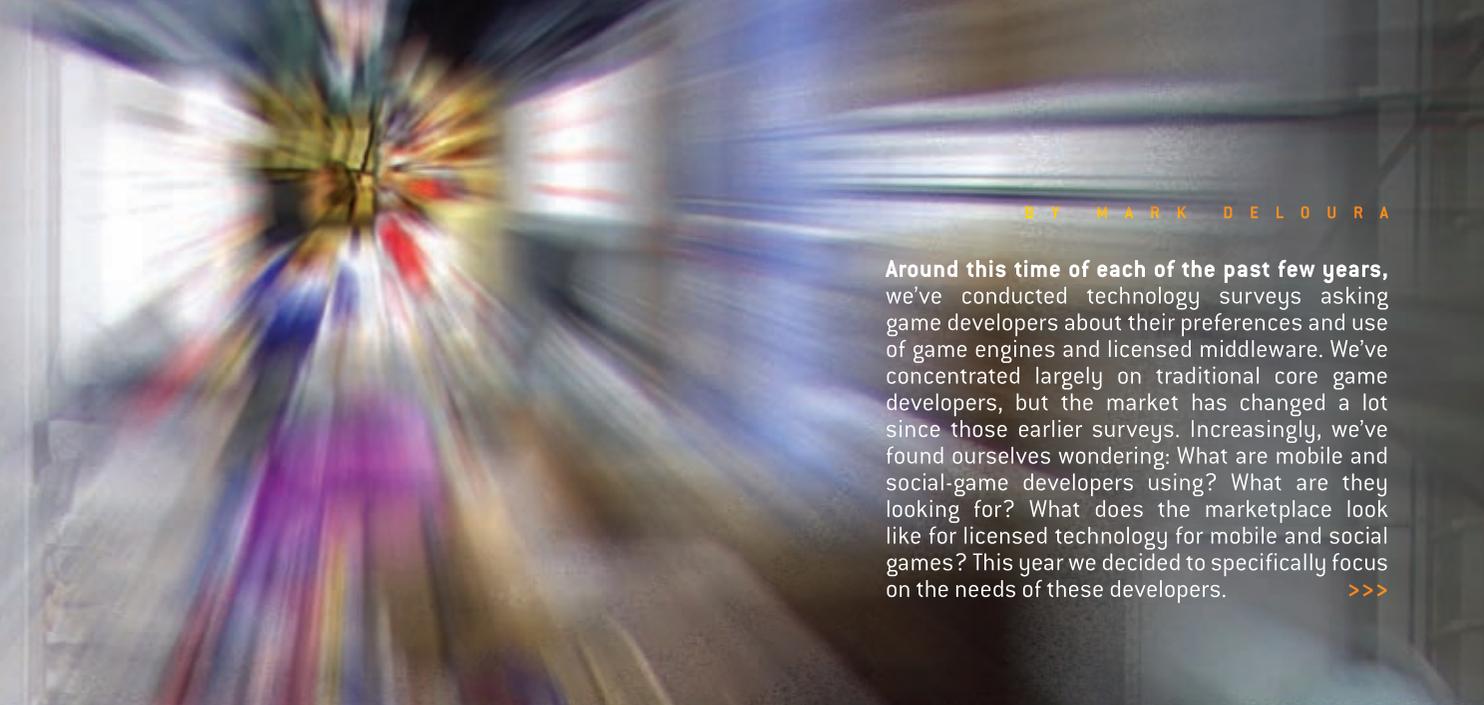
**E3 Expo**  
Los Angeles, CA  
June 5-7, 2012

**GDC Europe**  
Cologne, Germany  
August 13-15, 2012

**gamescom**  
Cologne, Germany  
August 15-19, 2012



Please email [licensing@epicgames.com](mailto:licensing@epicgames.com) for appointments



BY MARK DELOURA

Around this time of each of the past few years, we've conducted technology surveys asking game developers about their preferences and use of game engines and licensed middleware. We've concentrated largely on traditional core game developers, but the market has changed a lot since those earlier surveys. Increasingly, we've found ourselves wondering: What are mobile and social-game developers using? What are they looking for? What does the marketplace look like for licensed technology for mobile and social games? This year we decided to specifically focus on the needs of these developers. >>>

# SOCIAL & MOBILE TECHNOLOGY SURVEY 2012

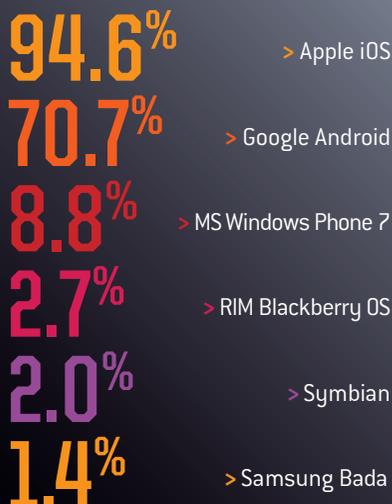
THE TECH THAT MOBILE AND SOCIAL  
GAME DEVELOPERS ARE USING AND  
THE TECH THEY'RE WISHING FOR

# SOCIAL & MOBILE TECHNOLOGY

## who took the survey?

✚ For the purposes of this analysis, we're breaking responders into two groups: mobile developers and social developers. In many cases there's an overlap, but for most topics, breaking down the replies into these two groups gives us useful insights. We classify mobile developers as those working on a mobile smartphone platform (not traditional handhelds such as Sony's PSP or the Nintendo 3DS). Of our mobile-game survey responders, 94.6% of them are working on Apple iOS titles, 70.7% for Google Android, and a smaller number for other devices:

### top platforms for mobile game developers



### top platforms for social game developers



Clearly, iOS is the primary platform for mobile developers. We found that nearly all mobile developers targeting alternate mobile phone platforms are also targeting iOS (93.7%). The majority of our social game responders are working on Facebook apps. In fact, the variety of platforms social-game developers report developing for is rather small—just Facebook, Google+, or a dedicated website. Of those working on games

intended for a dedicated website, 66.7% are also targeting Facebook with this game, while 100% of developers targeting Google+ are also targeting Facebook. Just 19% of Facebook developers are targeting Google+.

We should note that this survey doesn't cover PC-native free-to-play games, another interesting sector of the game industry with a different set of technology requirements. The technology used for the game client on these titles is more similar to triple-A PC-based titles than mobile or social games, although the server backend is similar to what is used for networked mobile and social games.

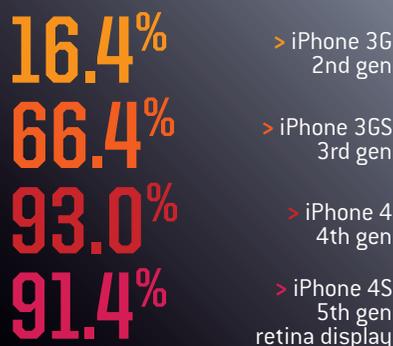
## OS versions and platforms

✚ One of the first challenges when considering making a mobile game is deciding which devices to support. For iOS, the number of choices is relatively small, but each tier of devices represents a large audience, so choosing how far back your OS support goes is very important. We found that the most popular shelf for support tends toward the introduction of Apple's Game Center, in OS 4.1.

### supported iOS versions



### supported iPhone devices



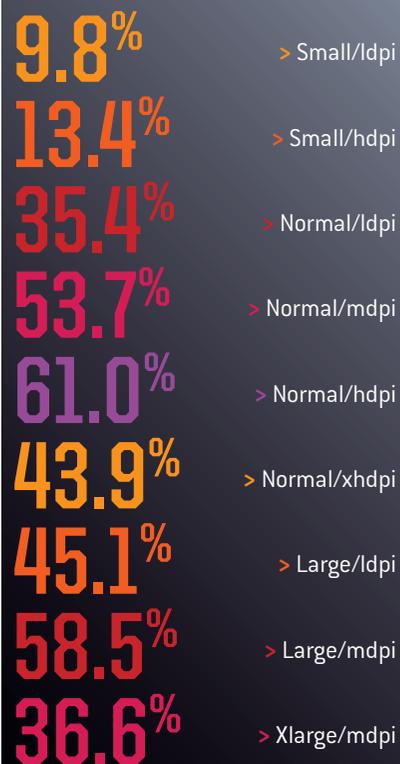
Choosing which hardware to support is similarly difficult. We found that many developers are still supporting devices back to the iPhone 3GS (third generation) and iPad 1, but the support for each

of these is gradually softening. Relatively few developers are still supporting the iPhone 3G (second generation). One developer justified supporting the 3GS with "We'll still release things that work on the 3GS, but the iPhone 4 and up means we're dealing with people who likely spend more money on mobile apps."

### supported iPad devices



### android display support

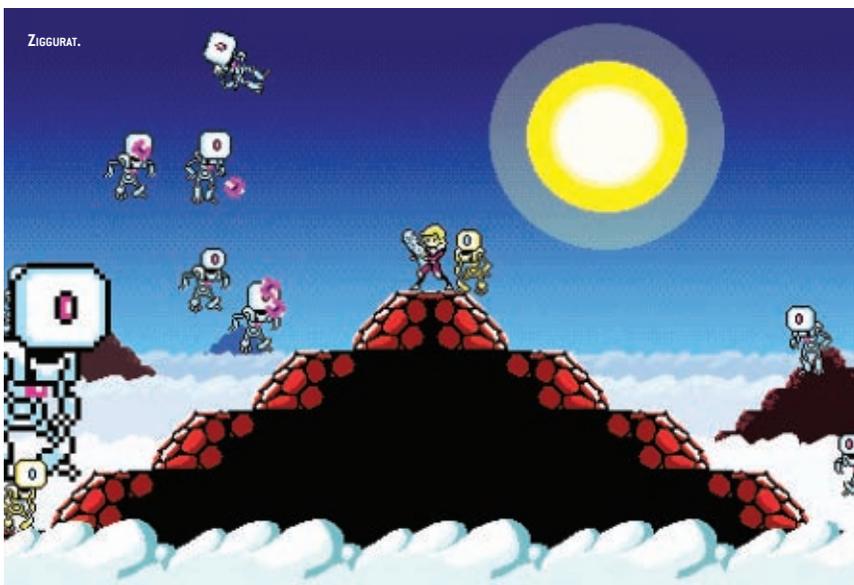


For Android devices, the preferences for OS support were less clear from our survey, but a deeper concern was the diversity of display devices. The Android developer web site contains relatively current information about the distribution of display devices accessing the Google Play market (see <http://developer.android.com/resources/dashboard/screens.html>). At the time of this article, it shows 66.3%

# SURVEY 2012



ZooWorld 2.



# SOCIAL & MOBILE TECHNOLOGY

of devices being actively used have a "normal" size screen with "high" display density (indicated as normal/hdpi); 18.5% of devices have a "normal" screen with "medium" density (normal/mdpi), and each other display formats fall below 5%. Yet, the distribution of support from our survey responders is substantially wider.

Several responders indicated, "In practice, there are almost no 'small' screens any more," and another noted that due to the variety of Android devices, but large installed base for relatively few, there's "really only interest in supporting the Kindle." A few developers commented that they are "using Unity3D to manage this."

Given the challenges from the variety of display sizes and densities available for both iOS and Android, we wondered whether developers were rendering to a default size and scaling in some way to fit the particular device: 29.6% of iOS developers and 33.3% of Android developers indicated that they are. However, developers noted that it varies depending on the type of game: "Some of our games scale, some use native resolution." Said one developer. Another said, "We try to start at iPhone 4 retina resolution and then scale down for other iPhones." Yet a third noted "We do 25% reduction in some low performance situations." Several people mentioned that they plan to use multiple sizes of assets, though the current app store size limitations can bite them one said, "We will eventually use two sets of assets with appropriate resolution, but will start by scaling up." And once again a few people said, "We use Unity to manage this."

## mobile game technology

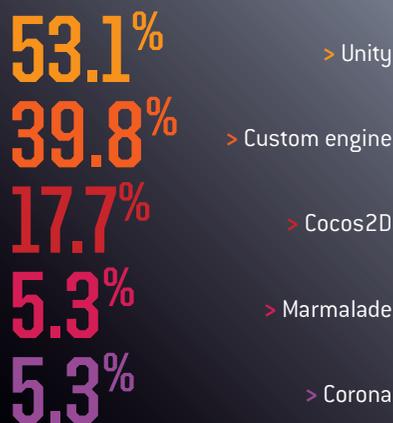
✚ The meat of our survey centered on the game engines and libraries developers are using for mobile games. What was most apparent in the results was incredible enthusiasm for Unity3D. 53.1% of developers commented that they are using Unity in some way, even higher than the number developing a completely custom game engine.

As for the technologies developers are looking at for future projects, once again over 50% of the responses pointed to using Unity or moving to it. Unreal garnered second place for future use with 14% of developers, and Cocos2D was also popular, with 10% of developers suggesting they'd like to use it. Moai, Game Salad, Game Maker, Corona, and Shiva3D also made multiple appearances in the list of potential future technologies.

We were curious about the value of cross-platform engines to mobile developers: Would mobile game developers be interested in working with technologies that also ran on other nonmobile devices? As it turns out,

interest in compatibility with devices other than smartphones falls off rapidly.

### top mobile game engines/ platforms



### cross-platform value (out of 5)



When it comes to determining which engine features are most important for a developer, it's no surprise that the most valuable feature is rapid development time (4.45/5). "A big decider for me is whether or not the engine can 'get out of my way' by enabling me to accomplish my ideas without feeling like I have to jump through hoops to do it," said one developer.

Of course, more and more mobile games are hooking into the player's social networks as well. While Apple has Game Center, there hasn't been much uptake of any Google+ related social networking for Android games. We found that 72% of mobile developers choose to use Facebook connectivity for their games. 65% point to integration with Game Center, and 44% use a player's Twitter network if they so choose. The most prominent independent game-focused network, GREE's OpenFeint, came in at a respectable 29%.

Middleware library developers are increasingly targeting mobile game devs, but by and large,

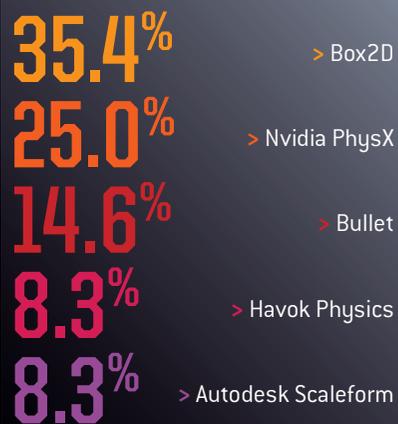
developers seem most interested in physics solutions—the top four middleware packages are all physics-related.

Several developers commented on their use of middleware libraries that are integrated with Unity, such as Beast, Umbra, FMOD, and Allegorithmic Substance. When asked what other libraries developers would like, there was a mix of ideas, but the two most common replies were an "Easy-to-use, small footprint, cross platform UI layout tool" to help with the various screen layouts (Scaleform GFX for Unity was asked for), and a "server-side scalable and customizable solution for mobile and social games" (we'll talk more about this later).

### top engine features (out of 5)



### top middleware libraries



## social game technology

✚ When it comes to technologies used for social games, it seems that the client-side technology options for social games are relatively few, but the distribution is certainly interesting:

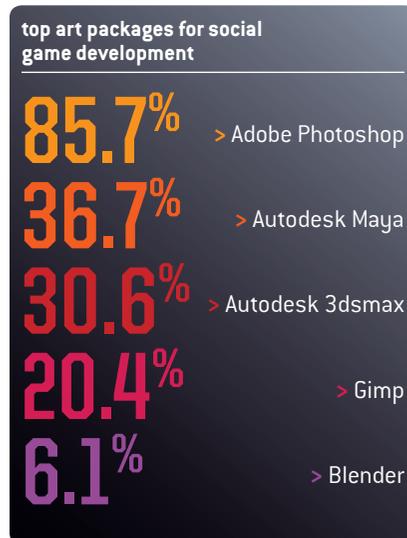


59.2% of developers pointed to use of Flash, 51.0% to Unity3D, and 46.9% to HTML5/Javascript. Of course there is some substantial overlap, and one has to assume that the majority of social games are using HTML5/Javascript in some form. What really stands out is the impressive support for Unity by the social-game responders. We haven't seen many Unity-based social games yet, but it seems like we may soon. As for libraries, Box2DJS and as3isolib were noted. One thing that was remarkably curious was the small uptake of Flash Stage3D; only 6.1% of developers claimed that they are working with Stage3D for 3D graphics in the browser. The Stage3D technology is interesting and powerful, but it may be that without a strong toolset like Unity's, game developers won't make much use of it.

Next, we wanted to know which browsers and art tools are being used by social game developers. Chrome and Photoshop came out on top.



libraries really what developers are interested in most? No. "Backend networks, payment systems, wrappers for platform APIs are where most of our tech emphasis is at. An 'engine' is a commodity," said one developer. A handful of folks commented that they really hope for server-side scalable and customizable tech for social and mobile games," or something like "FARMVILLE in a box, in the cloud. We could use it to prototype stuff quicker."

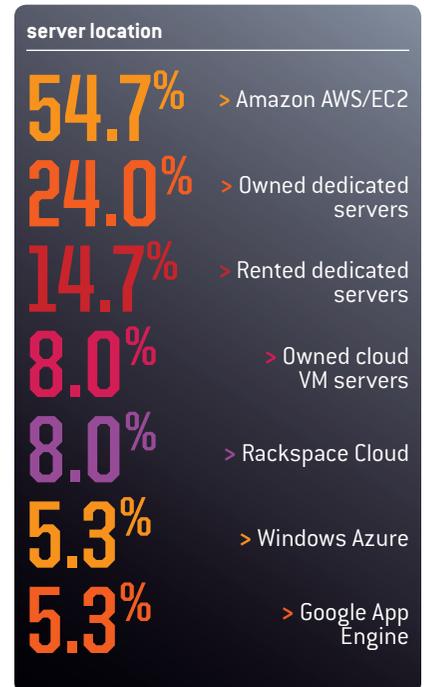


### server technology

In previous sections developers commented on a desire for improved server technologies. We asked survey responders to detail their experiences on their games. Impressively, Amazon Web Services, and EC2, are used by over 50% of all developers who

responded to the survey.

But what technologies are people using on those servers? Particular Amazon technologies (such as DynamoDB) did not rise to the top on this question, strangely. Amazon's server scalability is quite popular, but it seems that game developers are more comfortable with traditional web services such as MySQL and Apache HTTP Server.



The server technology comments emphasized the difficulty of building a scalable game service in the cloud: "I just want to write game logic on the server—I don't want to be concerned with scaling

Lastly, we wondered if social-game developers were looking for any libraries or frameworks that are not yet available. As the complexity of social games grows, middleware and engine providers will certainly gain in popularity. But are client-side

save the date

GDC 13

GAME DEVELOPERS CONFERENCE®

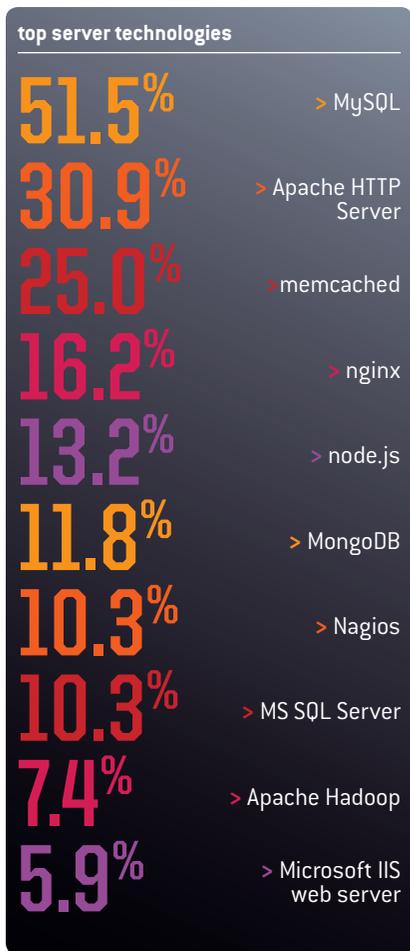
SAN FRANCISCO, CA /// MARCH 25-29, 2013

EXPO DATES: MARCH 27-29, 2013

[www.GDCONF.com](http://www.GDCONF.com)



databases, figuring out how many servers I need, etc.” Said one developer, “This is by far the hardest part of development right now. There seriously needs to be an entirely new category of commodity Internet infrastructure designed for scaling games”, said another. Fortunately, there are a growing number of potential solutions, such as MuchDifferent’s UnityPark Suite for those using Unity, PlayerScale’s Player.ID for Flash titles, and Electrotank’s EUP. This is also an opportunity for publishers, who can help developers with these tricky areas. Recently announced Kerosene Games has plans to assist developers with technology, including analytics, social plumbing, and cross-promotion capabilities.



One survey responder was aggravated about the potential costs associated with cloud computing: “... it’s very difficult to estimate costs ahead of time. If my game goes viral and I end up with a few million daily active users who each send a tiny packet of data once an hour to some cloud backend, what will I be charged?” This is a tricky problem, especially for those new to cloud computing. There’s a clear need for more accurate estimates and real-time reporting. Raveld is an example of one service that

is seeking to help address this problem.

## open source

For traditional core games, using open-source technologies can be challenging. Some open source licenses require release of all related source code, which is a problem for big retail games, and even more generous open source licenses leave developers wondering if they’re exposing themselves to potential patent problems. Typically, there is more leniency around using open source technology for a game’s tools than for the game code itself, but the policy varies from company to company. In the social and mobile spaces, which are moving at such a rapid pace and tend toward use of web services, is open source looked upon any more generously?



In general, our survey responders approve of use of open-source technologies, although there’s still a strong preference toward using open source in the tools versus in the game itself. Developers generally commented favorably on open source, although some expressed concern

that “open source tools are not that good in most situations at least in games. There are great databases and web development tools.” Typically developers favored “open source platform solutions, such as ad-serving, web CMS, and so on,” but were more cautious about game code.

Many developers expressed concern about particular open source license agreements. The GPL and LGPL require release of associated source code in many instances, so they can be difficult to work with for a commercial game. “I like the liberal licenses and try to stay away from the GPL and its ilk. If I can’t at least link to something in my proprietary binary, it’s of practically no value to me.” Preferred license agreements include the MIT license, BSD, and zlib.

## mobile game technology

Although not necessarily technology focused, we had a few other questions for our survey responders as well.

### Analytics

Most of the survey responders have rolled their own analytics solutions: 57.4% of social developers have and 47.5% of mobile developers. For social game developers, Kontagent (20.4%) was the most popular licensed solution, and although there were frustrations about Kontagent’s price tag, “Kontagent has what we need,” said one responder. Mobile game developers flocked toward Flurry (29.3%), and no other analytics solution garnered more than a 10% share. A variety of other solutions were mentioned including Mixpanel, Apsalar, Kissmetrics, Claritics, Localytics, Swrve, and simply using Google Analytics.



## Monetization

We were curious to find out the popular methods for making money, and how it differs between platforms. As we suspected, in-app payments turned out the most popular, among 81.8% of all responders. Among mobile developers, having a paid app was the next most popular monetization solution (52.3%), while using in-game ads was the most popular among social game developers (58.2%). As noted by one developer, "We tried different schemes on different platforms [free with in-app payments on Android, paid on iOS] and made dramatically more with free and in-app payments. So now we are free everywhere."

## Stores

There are a number of alternate app stores available. Are game developers finding them useful? Understandably, we found that the most popular app stores were the Apple App Store (94.4%) and Google Play (58.3%), with the percentages largely reflecting the distribution of developers working on iOS and Android games. But what other app stores are developers using? 32.4% reported putting their games up on the Amazon AppStore, but apart from that, very few developers are utilizing alternate stores. Common comments included sentiments such as "Honestly, only Apple and Amazon are worthwhile from our research and tests," and "Fragmentation of app stores is a very bad thing for developers, particularly indie developers." And yet, some see opportunity: "The Android Market is still in a growing stage, but surely developers that take the opportunity to get fame there will get good results in the future as there aren't that many apps compared to the App Store." In general, though, most developers' opinions were summed up



by this comment: "So far, they have almost no value to us. We've flirted with a few in the past but they cannot bring a significant number of users to make them worthwhile [yet]."

## Publishers

We asked mobile and social developers about their thoughts on using a publisher or distributor. As competition increases in these spaces and the costs for development and marketing increase, it may become more common for developers to partner up. Are they doing so already? Responders said no, by and large. Only 17.7% of survey responders claim to be working with a publisher or distributor currently. "We're hoping to self-publish and test the waters," said one. "We do contracts to

earn money for our main project," said another. Yet, 57.4% of the responders indicated they may look to work with a publisher/distributor in the future. "A publisher could be very well worth the cost in terms of aiding us with publicity and distribution reach," one developer noted. Some encourage caution: "It comes down to the contract detail. Keeping my IP and creative rights would be number one." It will be interesting to see how the funding landscape evolves for mobile and social games over the next few years.

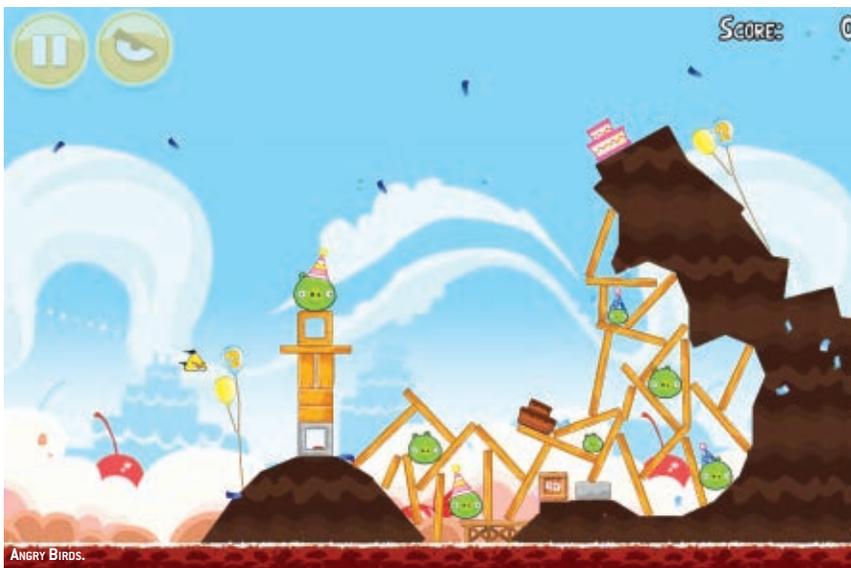
## parting thoughts

There are some clear winners in our analysis of the social and mobile technology scene. iOS and Facebook are the dominant platforms. A majority of game developers are using Amazon's cloud services to manage the scalability of their servers, and developers prefer MySQL for the backend database. Unity is the clear favorite for game engines—developers who aren't using Unity now are looking at it seriously. Social game developers are using Google Chrome a lot during development—admittedly, developers of browser-based games are using *all* the browsers a lot, though Chrome is the clear favorite. Server technology is challenging for game developers, and many developers wish more solutions were available, so they can focus on creating their game.

Yet, as one of our survey responders points out, "It's an amazing time to be a mobile developer. Creating games is more accessible than ever, and we hope to continue to learn, grow, and most importantly have fun while we develop." Another commented, "I would only like to say that there has never been a time when game developing has been as accessible as now. I'm not a programmer, but when I discovered all the tools that are available now, I realized I had no excuse not to try to make a game." On the flip side, one developer responded, "I'm sick of this overhyped bullshit—I'm throwing away my life on this and still not making any money."

The wisest response and best summary for our survey was perhaps this: "The mobile landscape changes every six months. Ask me all this stuff again in August, and I'll probably give you different answers." Of this we can be certain. 

**MARK DELOURA** is a video game industry technologist, and creator of the Game Programming Gems series of technical books. He has worked in technology, strategic, and business leadership positions at Sony, Nintendo, Google, THQ, and Ubisoft. Follow Mark on Twitter: @markdeloura.



# 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](http://E3EXPO.COM)**



entertainment  
software  
association

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](http://www.E3Expo.com) for registration guidelines.

Produced by  
**IDG**  
WORLD EXPO

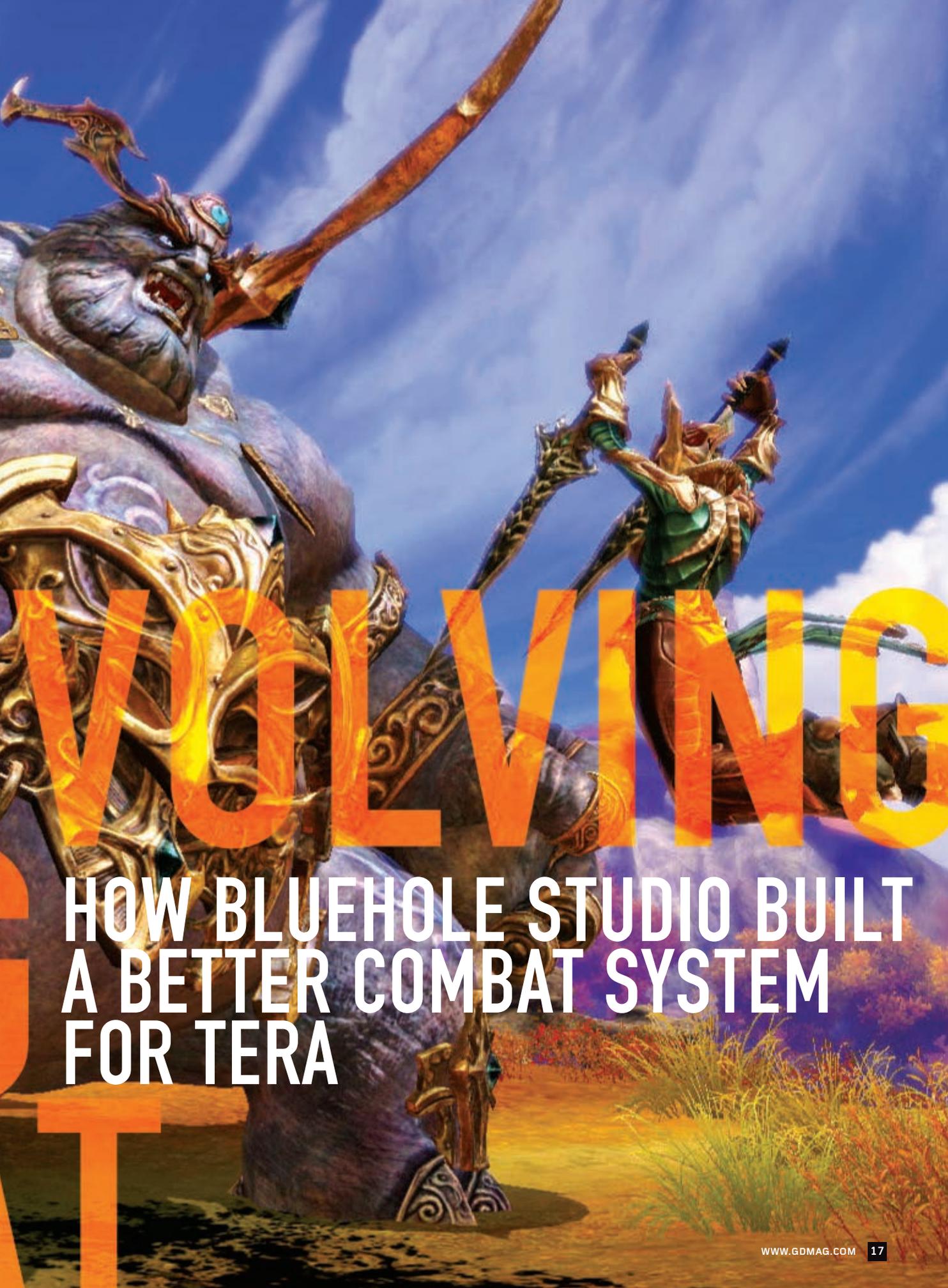
© 2012 Entertainment Software Association



SEUNGMO KOO

BY NOW, MMORPGS AS WE KNOW THEM ARE WELL OVER TEN YEARS OLD. A TYPICAL MMORPG PLAYER SPENDS OVER HALF THEIR GAMEPLAY TIME IN COMBAT. THAT MEANS THAT OF THE TIME OUR TYPICAL MMO PLAYER HAS SPENT OVER THE LAST TEN-PLUS YEARS, OVER HALF OF THAT HAS BEEN PLAYING IN A RATHER BORING COMBAT MODEL—TABBING FROM TARGET TO TARGET AND BUTTON-MASHING THEIR VARIOUS ATTACKS. THAT SIMPLY ISN'T VERY ENGAGING. THAT'S WHY WHEN WE STARTED DEVELOPING TERA, WE SET OUT TO MAKE MMO COMBAT EXCITING. HERE'S HOW WE IMPLEMENTED TERA'S "FREE-TARGETING" SYSTEM IN A WAY THAT DIDN'T REQUIRE A GIGANTIC FARM OF SERVERS.

# MMORPG COMBAT



# WOLVING

HOW BLUEHOLE STUDIO BUILT  
A BETTER COMBAT SYSTEM  
FOR TERA



FIGURE 2.1



FIGURE 2.2

FIGURE 1: Every creature in TERA is made of cylinders that you can attack.

## WHY WE MADE AN ACTION-HEAVY MMORPG

 We incorporated action-heavy fights in TERA because we wanted to differentiate our game from the conventional MMORPG's tedious combat. TERA players don't just trigger skills in a serial order with a set target. Instead, they have to dodge and block enemy attacks by moving their character away while taking the enemy's movement into account, and position themselves just right to aim their attacks while adjusting for their direction and distance. We don't even need to calculate dodging or

parrying statistics, as other MMOs do, because these factors are determined by the player's real-time reaction to enemy attacks. This kind of "free-targeting combat" is natural in console and single-player PC games because it's an effective way to support action-heavy battles, but early MMOs couldn't pull it off because they had too many latency-related problems.

Our free-targeting combat system uses area-of-effect attacks that have continuous timing information, so we assess multiple times whether each attack hits or misses, not just once per attack. Many MMORPGs already

incorporate partial elements of a free-targeting combat system—such as special attacks with cooldowns—but have not extended free targeting throughout their combat.

We also extended the concept of free targeting to include other interactions, most notably healing. When players have to direct and aim their heals and buffs, they're more excited and feel like the game itself is more skill-based. Our understanding is that Western gamers in particular gravitate toward gameplay that rewards player skill and feels less repetitive than existing offerings.

as conventional free-targeting combat, the processing load for the servers would become severe. The calculations necessary to assess whether each attack hits or misses would drastically increase CPU usage. Instead, we imitate a physics engine on the server by searching spaces for weapon range and checking collisions. Any MMORPG desiring to implement free-targeting combat mechanics should make those calculations server-side for security and combatant (player and monster) handling. If calculations are performed on the client-side, you run the risk of your clients being hacked to alter the calculations, and it could be nearly impossible to synchronize combatant information from every client in real-time.

In other words, for free-targeting combat to work, the server will need to process the battle calculations very rapidly. If the server has even a slight processing delay during a free-targeting battle, the resulting latency will frustrate players. To implement free-targeting battles in an MMORPG, we had to address the technical challenges of dealing with increased latency from server-side CPU load.

## MANAGING THE LOAD

 In a conventional MMORPG combat-targeting system, each attack calculates the distance and direction to a single target. It is a one-on-one attack and a low-impact calculation for server load.

The process for free-targeting combat, on the other hand, involves determining weapon range and the corresponding target data for each time frame, then resolving the attack using the collected data. If we went about the same way

```
<Skill id="10101" name="figure 2 skill sample">
  ...
  <TargetingSequence>
    <Targeting time="500" collisionVolume="value, value, ..."
    target="enemyMonster" >
      <SkillEffect damage="200" aggro="value" heal="0" ... >
        ...
      </SkillEffect>
    </Targeting>
    <Targeting time="650" collisionVolume="value, value, ..."
    target="enemyMonster" >
      <SkillEffect damage="150" aggro="value" heal="0" ... >
        ...
      </SkillEffect>
    </Targeting>
  </TargetingSequence>
  ...
</Skill>
```

**SAMPLE ATTACK SKILL DATASHEET TABLE 1**



## DEFINING THE DATA STRUCTURE FOR FREE-TARGETING COMBAT

 The server has no physics engine, because we felt it would be too CPU-intensive to compute physics. Instead, the world of TERA is geared toward space calculations and collision checks. Every creature [PC, monster, stationary object, and so on] is made up of cylinders that you can attack as shown in Figure 1. Each attack contains a timed collision volume (the collision volume expressed over time).

Figures 2.1 and 2.2 show the range of time-marked collisions as they change over time. By checking the collision between the target's cylinders and this timed collision volume, we determine whether the attack hits.

The XML datasheet example shown in Table 1 is the attack corresponding to Figures 2.1 and 2.2. The attribute time is the timing (in milliseconds) for the attack landing, and consequently the time to check the collision. The collision volume in Figure 2.1 and Figure 2.2 is used at 500ms and 650ms after the attack starts. The attribute `collisionVolume` treats the range with considerable detail. [For convenience, these attributes are generated by our skill-authoring tool.]

## TWO-STAGE PROCESSING FOR FREE-TARGETING ATTACKS

 Each free-targeting attack has several targeting times to determine the collision volume hits (We're using an attack as the example here, but it's worth remembering that heals and buffs work the same way). Depending on the type of attack, there may be anywhere from two to more than a dozen targeting times. Each targeting time is composed of a two-stage process; the front end and the back end. The front-end stage gathers targets that collide with the collision volume of the attacker's skill by searching the specific area of the world. The back-end stage applies effects such as damage to the gathered targets.

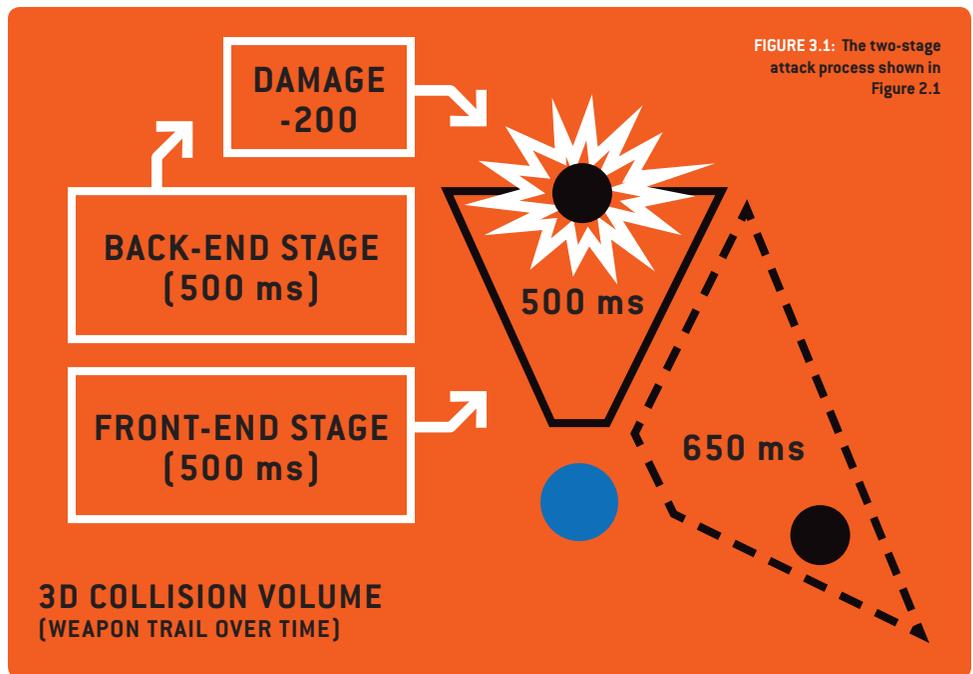


FIGURE 3.1: The two-stage attack process shown in Figure 2.1

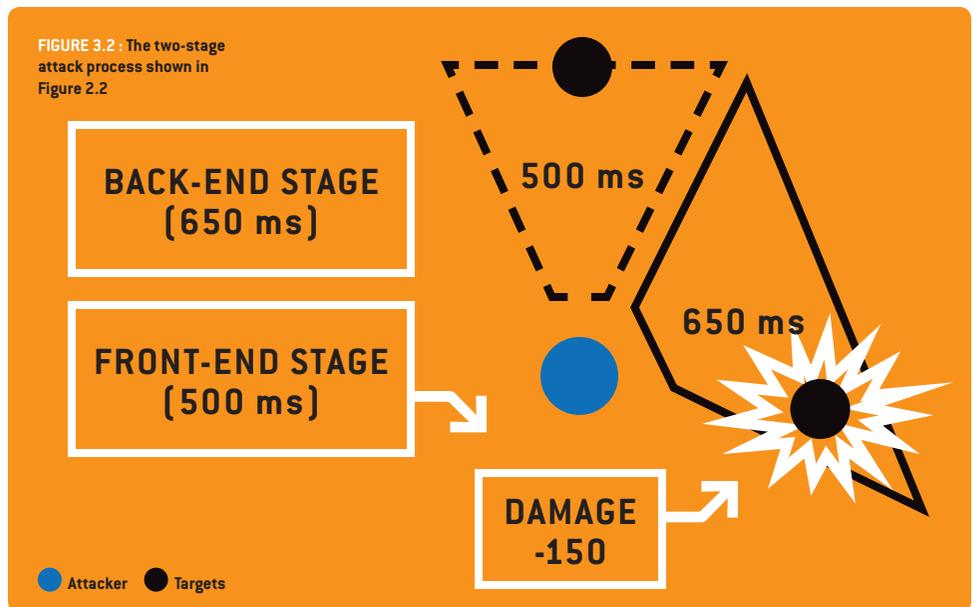


FIGURE 3.2: The two-stage attack process shown in Figure 2.2

Figure 3.1 and 3.2 show an example of this two-stage process. This figure is a simplified top-view diagram corresponding to the attack skill in Table 1 and Figures 2.1 and 2.2. This attack skill has two targeting times: 500ms and 650ms. At the first targeting time (500ms), an attacker collects one target whose cylinders intersect with the collision volume (front-end stage), then causes 200 damage to the target (back-end

stage). After that, the attacker deals 150 damage to another target in the same way at the next targeting time (650ms). The front-end stage corresponds closely to reading the targets' properties, while the back-end stage updates those properties.

While processing, the front-end stage makes frequent space searches to gather targets for a collision check, and the back-end stage requires targets

to update their properties frequently. If these operations use locks (our synchronization mechanism which enforces limits on access to a resource in a multithread environment), the lock-contention—which drastically increases blocking or CPU usage (up to 100%) on the server—would become severe, and we wouldn't be able to use our free-targeting combat throughout the game. Therefore, we use lock-free

## LISTING 1: CLUSTER UPDATE QUEUE (C++ PSEUDO CODE WITH WINDOWS API)

```
struct ClusterUpdateQueue
{
    ClusterUpdateTask* m_TaskQueue[MAX_TASK]; // Circular queue
    __int64 m_PushedCount; // Total pushed task count

    void PushBack( ClusterUpdateTask* task )
    {
        __int64 index = InterlockedIncrement64(&m_PushedCount) & 1
        ;
        // (#)
        InterlockedExchangePointer( (void**)&m_TaskQueue[ index &
        MAX_TASK ], (void*) task );

        // In case of blocking at (#), calling GetTask() will return
        NULL (It's OK)
        // this makes the loop in ClusterTask() escape.
    }

    void RemoveTask( __int64 index )
    {
        ClusterUpdateTask* prevTask = (ClusterUpdateTask*)
        InterlockedExchangePointer(
        (void**)&m_TaskQueue[ index & MAX_TASK ], (void*)NULL );
        delete prevTask ;
    }

    ClusterUpdateTask* GetTask( __int64 index )
    {
        if ( index < m_PushedCount )
            return m_TaskQueue[ index & (MAX_TASK-1) ] ;
        else
            return NULL ;
    }

    // performs tasks in m_TaskQueue
    void ClusterTask() ; // called in the worker thread loop.
    (implementation in Listing 3)
};
```

methods in our programming to reduce the lock-contention.

### LOCK-FREE IMPLEMENTATION FOR TWO-STAGE PROCESSING

 First, we adopt the symmetric worker thread pattern [1] to remove locks effectively in two-stage processing. This pattern has one single thread-pool that consists of as many symmetric worker threads as the number of CPU cores; no thread exclusively handles a particular content or area. Any worker thread can perform any kind of task. Worker threads process all tasks using nonblocking methods; there are no blocks among worker threads utilizing lock-free implementation [2]. This makes worker threads fully utilized; that is to say, it is not necessary that we create more worker threads than the number of cores. This pattern gives us direct benefits because the number and speed of the CPU

cores has a big influence: More cores mean more threads, and faster cores mean faster threads.

Let's look at our worker thread loop as follows:

```
void WorkerThreadRun()
{
    while ( 1 )
    {
        IoCompletionPortTask() ;
        // A
        TimerTask() ; // B
        ClusterTask() ; // C
    }
}
```

The worker thread is an infinite loop throughout server uptime. The number of worker thread loops is the same as the number of CPU cores. Because we input all requests through Windows I/O Completion Port (IOCP) on the server, packet handling from clients is processed in `IoCompletionPortTask()`, which calls `GetQueuedCompletionStatus()`

(Windows API) with a timeout value from 0ms to 16ms according to the turnaround time (processing time for A to C) of the thread loop. In our worker thread loop, `TimerTask()` performs scheduled tasks in time by checking the current system time via `GetTickCount64()` (Windows API). The `ClusterTask()` exists for the effective area search due to the nature of free-targeting combat. We will explain this in detail in the next section.

If there is a lock that blocks two-stage processing within the worker thread loop functions, it will postpone packet handling and scheduled tasks. That's why we designed a lock-free mechanism for the two-stage processing so the worker thread loop never stalls.

### FRONT-END STAGE IMPLEMENTATION

 We divide TERA's seamless world into squares called clusters. In any circumstance, every creature must be attached to a specific cluster; each cluster is a container that holds pointers for creatures. Figure 4 displays creatures on the clusters in a diagram form. When a creature moves from cluster 32 to cluster 33 [C à C'], for example, cluster 32 removes the creature while cluster 33 adds it. In a similar way, a creature can be added to or removed from a cluster when it spawns or dies.

It's worth noting that each worker thread has its own thread-local storage (also called thread-specific storage [3]), which maintains information about clusters. In other words, the cluster information is replicated entirely to each thread-local storage (the number of worker threads determines the copy count of the cluster information). Figure 5 represents an example of this structure.

The cluster update queue, shown in Figure 5, serves as a task dispatcher for updating cluster information in consecutive order. The cluster update queue is a global variable shared by all worker threads. Listing 1 shows its implementation in the form of a lock-free circular queue for best performance.

When an action requires updating clusters (such as, when a monster moves from one cluster to another), a worker thread adds a task (see Listing 2) at the back of the `ClusterUpdateTask` queue. All worker threads will eventually process this task independently so that their private thread-local storage information is updated. Because all worker threads process the same queue using the same algorithm, the per-thread cluster information stored in thread-local storage should be identical.

The cluster update tasks that are posted to the queue provide delta information, so a task might say "remove object 28 from cluster 32 and add to cluster 33." Since these tasks must be processed by all of the worker threads to ensure the thread-local storage information stays synchronized, the tasks each have a reference count. If there are eight worker threads, the reference count will be eight. As each thread completes the task, it decrements the count, and the last worker thread will then delete the update task. As you can imagine, we use a lock-free function (in this case, the Windows function `InterlockedDecrement`) to ensure that multiple threads can safely decrement the reference count at the same time.

When each worker thread invokes `ClusterTask()`, it performs all queued tasks in the cluster update queue to update its replicated cluster information (see "Pop and Execute Cluster Task" in Figure 5). All worker threads have an index that points to the position of the most recent executed task on the cluster update queue. It is a global variable, local to a thread. The code below shows how to declare the index, ["Current Index" in Figure 5] in the form of the thread-local variable on the Windows system.

```
__declspec(thread) __int64
TLS_CurrentTaskIndex ;
```

Whenever one cluster update task is executed in `ClusterTask()`, each worker thread increases its own `TLS_CurrentTaskIndex` by one. When this cluster update task finishes throughout all worker threads, the last-performed worker thread—the



**LISTING 2: CLUSTER UPDATE TASK (C++ PSEUDO CODE WITH WINDOWS API)**

```

class ClusterUpdateTask
{
public:
ClusterUpdateTask (): m_RefCount(0) { }

bool UpdateCluster()
{
    [... ...] // Code for Updating Cluster Information

    // Is this final worker thread?
    long refCount = InterlockedDecrement(&m_RefCount) ;
    if ( 0 == refCount )
        return true ;
    else
        return false ;
}

void AddRef( long refCount )
{
    InterlockedAdd(&m_RefCount, refCount) ;
}

private:
volatile long m_RefCount ;
[... ...] // Members for delta-information for cluster update
}
    
```

**LISTING 3: PERFORMING TASKS FOR UPDATE CLUSTER INFO (C++ PSEUDO CODE WITH WINDOWS API)**

```

void ClusterUpdateQueue::ClusterTask()
{
    while ( 1 )
    {
        ClusterUpdateTask* task = GetTask( TLS_CurrentTaskIndex ) ;
        ;
        if ( NULL == task )
            break ;

        bool final = task->UpdateCluster() ; // execute!

        // remove the task when all worker threads carried out
        // this task
        if ( final )
            RemoveTask( TLS_CurrentTaskIndex ) ;

        ++TLS_CurrentTaskIndex ;
    }
}
    
```

**LISTING 4: TASK WRAPPER FOR LOCK-FREE EXECUTOR (C++ PSEUDO CODE WITH WINDOWS API)**

```

template <typename Arguments>
struct Task
{
    typedef void (Creature::*MemberFunction)( Arguments ) ;
    // return type: "void" only

    Creature* m_Creature ; // creature
    MemberFunction m_Func ; // function pointer to invoke
    Arguments m_Args ; // function argument
}
    
```

thread getting a return value of true when called by `ClusterUpdateTask::UpdateCluster()`—removes it from the cluster update queue. Listing 3 shows the program code for this process.

When searching for creatures in some target clusters, it is possible to get their pointers directly from the thread-local storage without locking the clusters, because each

thread has its own copy of the cluster data. On the other hand, if a creature moves from one cluster to another cluster, all worker threads update the cluster information in their thread-local storage. In other words, the advantage of this mechanism is that it takes more work to read the cluster information than to update it because of the

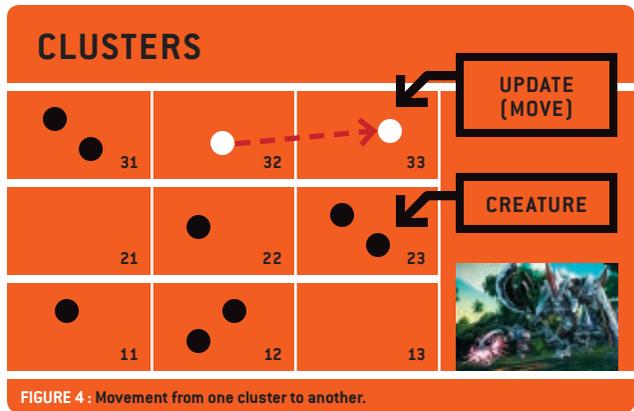


FIGURE 4: Movement from one cluster to another.

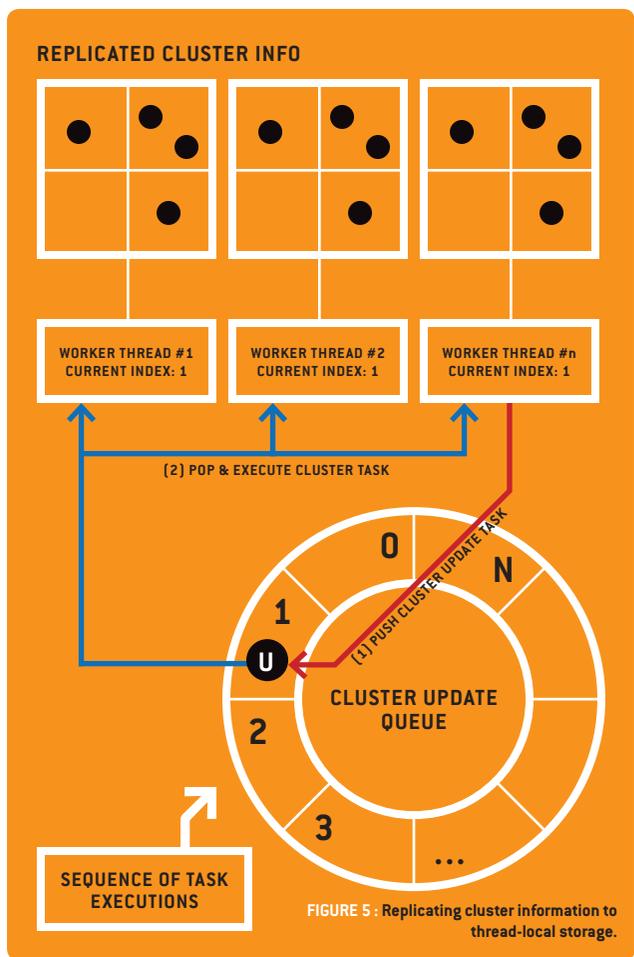


FIGURE 5: Replicating cluster information to thread-local storage.

nature of free-targeting combat.

Overall, the cluster mechanism is very effective for the front-end stage because so many queries will find creatures in a particular space.

**BACK-END STAGE IMPLEMENTATION**

The back-end stage is the process that modifies

creatures gathered in the front-end stage. This process makes frequent update operations.

To reduce the HitPoints (HP) value of a target creature by 200, for example, two possible implementations to deliver damage to an enemy are shown in this code:

## LISTING 5: LOCK-FREE EXECUTOR (C++ PSEUDO CODE WITH WINDOWS API)

```
// list of LockFreeExecutor registered in this worker-thread
_declspec(thread) deque<Lfe*>* TLS_LfeList ;

// indicates the LFE which occupies this worker-thread at this moment
_declspec(thread) Lfe*      TLS_CurrentLfeOccupyingThisThread ;

// Lock-Free Executor (LFE)
struct LockFreeExecutor
{
// Task Queue (we built this queue by modifying the reference [5]) struct LockFreeTaskQueue
{
    void Push(Task* newValue) { ... }
    vector<Task*> Pop() { ... }
    ...
};

// member variables
LockFreeTaskQueue      m_TaskQueue ;
__int64                m_RemainTaskCount ;

LockFreeExecutor () : m_RemainTaskCount(0) {}

// Push a task into TaskQueue, and then Execute tasks if possible void DoTask(Task* task)
{
    if ( InterlockedIncrement64(&m_RemainTaskCount) != 1 )
    {
        // register the task in this LFE
        m_TaskQueue.Push(task) ;
    }
    else
    {
        // register the task in this LFE
        m_TaskQueue.Push(task) ;

        // Does any LFE exist occupying this worker-thread at this moment?
        if ( TLS_CurrentLfeOccupyingThisThread != NULL )
        {
            // just register this LFE in this worker-thread
            TLS_LfeList->push_back(this) ;
        }
        else
        {
            // acquire
            TLS_CurrentLfeOccupyingThisThread = this ;

            // invokes all tasks of this LFE
            Flush() ;

            // invokes all tasks of other LFEs registered in this thread
            while ( !TLS_LfeList->empty() )
            {
                Lfe* lfe = TLS_LfeList->front() ;
                TLS_LfeList->pop_front() ;
                lfe->Flush() ;
            }

            // release
            TLS_CurrentLfeOccupyingThisThread = NULL ;
        }
    }
}

// Execute all tasks registered in TaskQueue of this LFE
void Flush()
{
    int count = 0 ;
    do {
        vector<Task*> taskList = m_TaskQueue.Pop() ;
        count = taskList.size() ;

        for ( int i=0 ; i<count ; ++i )
        {
            Task* currentTask = taskList[i] ;
            currentTask->OnExecute() ;
            delete currentTask ;
        }
    } while ( InterlockedExchangeAdd64(&m_RemainTaskCount, -count) != count ) ;
}
```

Case of using a critical section

```
enemy->EnterCriticalSection()
;
enemy->DoDamage(200) ;
enemy->LeaveCriticalSection()
;
```

Case of using a spin lock

```
enemy->EnterSpinLock() ;
enemy->DoDamage(200) ;
enemy->LeaveSpinLock() ;
```

In this way, whenever we change a property (i.e., HitPoints) of another object (i.e., enemy) in a multi-thread environment, we must use a lock (such as critical section and spin lock) for thread safety. If there are a large number of updates to the shared data—in this case, enemy—at the same time (we call this phenomenon “lock-contention”), it either increases blocking (in case of using a critical section) or increases CPU usage drastically (in case of using a spin lock). This situation causes latency.

For this reason, we built a task dispatcher called Lock-Free Executor to update them locklessly. Every creature has a Lock-Free Executor to guarantee the execution order of its member functions. In this example, the function pointer changing HitPoints with an argument (i.e., 200) is wrapped as a task, and then the Lock-Free Executor of that creature registers this task for subsequent execution. This code is the case of using Lock-Free Executor for higher concurrency:

```
enemy->LockFreeExecutor.
DoTask(&Creature::DoDamage, 200) ;
```

The member function pointer, **&Creature::DoDamage**, with the argument 200 is passed into enemy's Lock-Free Executor by calling **DoTask()**. In fact, this call is wrapped as a Task shown in Listing 4 (**m\_CreatureEnemy, m\_ArgsΔ200, m\_FuncΔ&Creature::DoDamage**), and then this Task is registered to enemy's Lock-Free Executor, which performs its tasks in sequence.

Listing 4 shows something peculiar; it allows the member function to wrap only a void return type. Because the task execution in Lock-Free Executor is asynchronous, we cannot get a return value immediately. To



receive the return value, we should build the programming pattern similar to Active Object [4].

For example, if the monster instance deals 200 damage to player instance and gets a return value [HitPoints] from calling `Player::DoDamageAndGetHitPoints()`, we need a member function to receive the return value—`Monster::ReceiveHitPoints( int returnVal )` in this example.

Look at this. The execution sequence is {1} to {3}.

This way, we get the return value asynchronously.

Listing 5 shows Lock-Free Executor code. Every creature has a Lock-Free Executor for updating its states (properties such as HitPoints and ManaPoints) from other objects. If some other objects want to call a member function that modifies another object's properties, they must call the member function through using `LockFreeExecutor::DoTask()`. It guarantees the execution order of function calls on a creature by utilizing a lock-free queue,

`LockFreeExecutor::LockFreeTaskQueue`, based on nonblocking concurrent queue [5].

Besides using Lock-Free Executor in the back-end stage, we could use it anywhere to guarantee an execution order in performing tasks without a lock—such as changing other objects' properties in `IoCompletionPortTask()` for handling packets and `TimerTask()` for processing scheduled tasks.

These two-stage processes make free-targeting combat possible in TERA by decreasing the servers' CPU usage to one-twentieth of what it was previously. Before the two-stage processing was implemented, we first tested the lock-based algorithm to verify the CPU load and server response. The test showed that the algorithm consumed high CPU loads (up to 100 percent CPU usage with fewer than 3,000 players), which resulted in slow server response times. The lock-based algorithm made free-targeting combat nearly impossible. However, the two-stage processing based on the lock-free

paradigm dramatically decreased our server loads, thereby opening the possibility to free-targeting combat with minimal affect from latency. In live service, the average CPU usage of TERA's world server (Intel Xeon E5630) is 1 percent to 6 percent with anywhere from one to 6,000 real players.

### THE EVOLUTION OF ACTION COMBAT

 We dramatically enhanced the combat of TERA by adopting free-targeting combat. It was complicated to implement and required much more time to code, but we think the player experience is much more fun than other MMO combat systems. We hope that many online game companies follow our lead and utilize free-targeting combat for action-intensive MMORPGs in the future.

The payoff is obvious as soon as core gameplay begins. The player aims, dodges, and repositions from moment to moment, rather than managing a set sequence of attacks. The player

**resources**

- 1 *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Object* Douglas Schmidt et al. John Wiley & Sons, 2000.
- 2 *The Art of Multiprocessor Programming* Maurice Herlihy et al. Morgan Kaufmann, 2008.
- 3 *Thread-Specific Storage for C/C++* Douglas Schmidt et al. C++ Report, 1997.
- 4 *Active Object: An object behavioral pattern for concurrent programming* R. Greg Lavender et al. Pattern languages of program design, 1996.
- 5 *Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms* Maged M. Michael et al. ACM symposium on Principles of distributed computing, 1996.

watches the character and the monster, not cooldown timers or other U.I. elements. This sense of immersion—that sense of freedom, really—is what will make free-targeting combat an evolutionary leap for MMORPGs. 

**SELUNGMO KOO** was server architect for TERA and is currently co-director at Bluehole Studio. Twitter: @sm9krvt



# 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](http://vfs.com/enemies)



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





postmortem

GAME DATA

**INDIANA JONES**  
ADVENTURE  
WORLD

**PUBLISHER**

Zynga Inc.

**DEVELOPER**

Zynga Boston

**RELEASE DATE**

September 9, 2011

**PLATFORMS**

Web (Facebook)

**TOTAL WHIPS PURCHASED**

14

**TOTAL "ACCIDENTS"  
INVOLVING WHIPS**

2

IN THE SUMMER OF 2010, THE SOCIAL GAMES SPACE WAS STARTING TO REALLY PUSH THE BOUNDARIES OF GAMEPLAY, STORY, AND PRODUCTION VALUE. THE NEWLY-MINTED ZYNGA BOSTON STUDIO WANTED TO KEEP THAT INNOVATIVE MOMENTUM GOING WITH INDIANA JONES ADVENTURE WORLD (IJAW). THE PROMISE OF IJAW, INTERNALLY KNOWN AS "QUEST", WAS TO DELIVER A SOCIAL ADVENTURE THAT DISTILLED TRADITIONAL ACTION-ADVENTURE GAMEPLAY INTO SOMETHING APPROACHABLE FOR EVERYONE.



Zynga Boston is made up of industry veterans from Harmonix, Turbine, Insomniac, as well as people from the web and tech industries. The studio was formed from the acquisition of Conduit Labs in August 2010, and when we came on board we were encouraged to pitch a project we were enthusiastic about—a game we could deliver with a high level of polish and innovation. As happens with most teams that strive to build something new, we nailed some elements, but missed on others.

INDIANA JONES ADVENTURE WORLD launched on September

9th, 2011. Over the last seven months, close to 28 million players from over 200 countries have played the game. Since launch, we have released more than 20 new story lines, and our players have whipped over 750 million snakes and recovered 1.8 billion artifacts. We like to think even Dr. Jones would be impressed with such numbers.

#### WHAT WENT RIGHT

##### 1 / Prototyping

Our goal for INDIANA JONES ADVENTURE WORLD was to evolve social games by introducing players to the

types of games that we love to play. We wanted to take action-adventure games like Legend of ZELDA: Ocarina of Time and Tomb Raider and find a way to deliver that experience to an audience that expects to use a single mouse button.

We wanted to introduce social gamers to a casual (but exciting) form of combat. That wouldn't be easy, because one of our core design values was "never punish the player." The very first "beasties" in the game were simply animated obstacles that didn't fight back. We knew we needed to make this interaction more

# INDIANA

# JONES

# ADVENTURE WORLD

SETH SIVAK

# IJAWA JUNGLES



meaningful and interesting, and that meant we had to start building prototypes.

The design team spent weeks toying with mechanics like stealth, avoidance, and timing-based skill. We built prototypes that used real-time movement and others that were turn-based. We tried dozens of different movement rules and an even larger variety of user interface treatments that attempted to explain them all. Eventually, we went broad and tried puzzle-based combat and a few other disparate ideas to see if there was anything that stuck. One good example of this is the “tank beastie.”

IJAW is a turn-based game; each tile that the player crosses counts as a turn. We designed a beastie that would move, rotate, and attack at different rates based on the number of turns taken by the player (the “tank beastie” would turn really slowly but attack for a lot of damage). This meant we needed to create a “tell” that could explain to the player that the beastie was about to move, turn, or attack. We also had a threat range that the player could back out of, which would make the beastie

stop. This was just way too much information, and even when players understood all of it, the gameplay was not terribly fun.

Every prototype was put in front of people in the office and playtested side-by-side. We judged the designs primarily on their ease of use and how easy they were to understand, but also in terms of depth and overall fun. We hoped that combat would be a cornerstone of the game for all our players—even those who had never experienced combat in a game before. We went with a very simple final design that had a handful of basic mechanics, which included critical hits that caused stuns, weapon upgrades that dealt more damage, and dodges. When the player is damaged, he loses a unit of energy (which is used every action in the game). While this went against our core design value of never punishing our players, it allowed us to give them a feeling of judgment and mastery that we couldn’t get otherwise. We like to think that IJAW was one of the first Zynga games with consequences. To our surprise, players never found the damage

confusing, and they embraced it as part of the game.



#### LESSON LEARNED:

**Innovation is not created on paper, so make as many prototypes as possible. Every paper design (even core values) can benefit from the momentum of a good prototype.**

## 2 / Constant Focus Testing

Designers have a love-hate relationship with focus testing. It’s not fun to watch players struggle with the game you have been putting your heart and soul into and then hear them list all the things they hate about it. We decided to take advantage of Zynga’s Player Insights group and run playtests every four to six weeks. These tests were broken down into two specific groups: intensive first-time-user experience testing (FTUE), and unassisted progression testing.

The FTUE focus testing had players try the first few maps of the game with help, and take a survey. A person from the team sat in the room with them to observe body language, and we would screencast their playthrough for the rest of the

design team to watch from outside. We recruited playtesters that fit our target audience, which represented a wide range of players inside of our demographic. For each round of testing, we had fifteen testers come in Monday, Wednesday, and Friday, and we made sure we had designers and engineers on the hook to fix usability issues and confusing bugs found during the test.

Our second set of focus testing was done with large groups who would play unassisted for an hour and then fill out a survey. We used this to get a read on the overall feel, what players remembered about the FTUE, and what they liked most and least. We asked specific questions to gauge where players wanted more gameplay and how they felt about story elements and characters. This data helped us build out more compelling features and tune the difficulty ramp for the combat and puzzles.

This process forced us to reevaluate some assumptions we made as longtime developers and gamers. IJAW was the first Zynga game to include height, and we didn’t realize that we’d have to teach our players about height in an



isometric game, or how to click and drag to move the camera.

**LESSON LEARNED:** Watch focus testers struggle through the game and try to feel what they feel. Test early and test often, even if you think the game is not ready. Act on the feedback quickly while it is still fresh. You can always use another focus test.

### 3 / Design through Iteration

We knew we were not going to design a perfect social adventure game on our first try. With that in mind, we structured the team and features around weekly sprints that let us rapidly iterate the core gameplay and content around the feedback we were getting from focus testing. When we started work on the FTUE, we knew it would become the most important aspect of the game, but we were still trying to find the fun and figure out our core focus. We spent months iterating on the first map and went through several reboots (some bigger than others). During that time we questioned many of our own core design values and ended

up with a FTUE that not only could teach players how to play a social adventure but also made a promise about the innovative story and gameplay to follow.

The first map of the game, known as the Upper Jungle, was changed close to 600 times during the course of development. Many of these changes came from watching the focus testing, but others came from the design team trying to figure out how to create a map that was easy to understand, and which had good flow. Sometimes these changes meant going back to the basics—for example, we changed the level design from a large open map to a series of rooms (which is how dungeons have been built in games for a long time).

**LESSON LEARNED:** Structure the team to rapidly iterate and understand things will never be right on the first try. The best designers are not born that way—they learn through iteration.

### 4 / Technology

Social games have come a long way since the Facebook platform

launched in 2007. The platform and the technology available to early social games had serious limits on graphical quality and performance. We wanted to ensure that our game would move the visual quality bar for social games and maintain a gameplay level that traditional game players have come to expect from a triple-A console title. We also wanted to build a set of tools to make the creation of assets and content quick for content designers and artists.

We chose to build an entirely new engine that could support the kind of large-scale levels and gameplay we hoped to deliver. We wanted the quality of the experience to be paramount to almost everything else, so it became a massive priority that drove both engineering and design decisions. This focus really paid off, as we were able to deliver some of the best frame rates and load times of any social game. The engine, affectionally called "BRO" (Boston Rendering Optimization) has made appearances in other Zynga games like CITYVILLE and CASTLEVILLE.

The game was designed to be content-driven, which meant



# INDIANA JONES



we needed to have tools that could empower designers to work quickly and autonomously, which is commonplace in MMO development, but not as much in social game development. Toward that end, we built a world editor that allowed designers to build maps without the help of engineers and a suite of art tools that allowed our characters to have a wide range of animation. The time we spent making these tools easily paid off in the speed and iteration available to a designer when constructing a new map.

IJAW was built around the idea of consistently delivering content to players, and it would have been almost impossible to do that without a dedicated set of tools so that designers could build their maps freely while engineers focused on feature development.

 **LESSON LEARNED:** Invest in technology and tools. Social games are casual and light, but the technology needed to run them is serious.

## 5 / INDIANA JONES

We initially launched our game as ADVENTURE WORLD, but immediately we were given the chance to bring Indiana Jones to life inside the game. This was a great opportunity for us, but it was also a shift in the core narrative and a big design challenge. Once we were over the initial hump, the license became a big boon for the game and the development team. We were able to quickly get the hang of the new process and start using the characters and canon in new content for our players.

Licenses are tricky, but they can pay off. One thing we noticed almost immediately was that our brand recognition went up. Players were trying the game who might not have been interested before. In the social-games space, where finding and connecting with an audience is competitive, having that brand recognition is a huge advantage.

The *Indiana Jones* license was a particularly good fit because the characters and story are deep, and it has a hardcore following of fans who are hungry for more content

all the time. We built this game to deliver episodic stories to our players each week so that they can go on a quick adventure with their friends. There are very few licenses that would have been a better fit for us.



As fans of *Indiana Jones*, we really enjoy using the characters to tell stories and putting in easter eggs that later end up on the *Indiana Jones* wiki. Interacting with hardcore Indy fans is really satisfying.

 **LESSON LEARNED:** A good license can give a social game instant brand recognition and increased player loyalty.

## WHAT WENT WRONG

### 1 / Innovate Everywhere

The goal for IJAW was to innovate with new gameplay and establish a new genre of social adventure games. As it turns out, that is really hard. We tried to innovate with every mechanic, user interface, and idea in the game with the intention of building something new. This became dangerous because social game players have come to expect a set of



# ADVENTURE REW

postmortem

features that they believe should be part of every game and should work consistently across games.

Our collection system fell victim to this particular problem. For the readers new to social games: A collection system provides a player with a slot machine feel for every action in the game. Each object the player interacts with has a chance of dropping an item from a specific collection, and when the player completes the collection, they receive a reward. We went through several iterations for a collections system that included many different additions to the core feature. Our first try, which was called "Discoveries," was a standard collection system. Ideally, a player would make a Discovery, give it a unique name, and then have the option to share that discovery with friends. That additional functionality increased the Discoveries system's scope, so several of the more interesting parts were put into a secondary tier. Without those additional features, however, the system did not feel very rewarding, and the relationship between the Discoveries and the objects that provided them

were not clear to the player. In the end, we decided to keep the traditional collection mechanic and scrap the Discoveries system completely. Cutting a feature your designers have been working hard on is nothing new to the game development process, but we did this everywhere—even places where we had no reason to do it (quest structure, for example). We ran into issues later on with features like Mastery, where we should have innovated but did not have enough time left to do so.



**LESSON LEARNED:** Focus on a few key areas to innovate and invest heavily in them. Resist the urge to make everything a little bit better and instead make a few things a giant leap better.

## 2 / Single Session Focus

Our focus on gameplay iteration meant we used only a single session to measure how fun the game was. This is a real challenge for social games that need to run for months, or even years. We did not play through a fully working version of the game until we were



almost into the Beta, and this meant most of our multisection features were not tested until it was far too late.

The game was built to be fun for a session, but social games are designed and played like a traditional MMO, so there needs to be a reason to come back tomorrow. In social games this is commonly known as "retention" and it is a notoriously difficult value to move after launching a game. (Some games solve this with an appointment mechanic.) In IJAW we thought we could incentivize players by putting time limits on the adventures.

We never wanted players to feel bogged down or overwhelmed by the number of adventures. We hoped to give players a way to leave an adventure unfinished and have it just wash away like a withered crop would in FARMVILLE. The problem was that we always played through maps in a single session, with infinite energy, to see how fun they were, but we rarely played them for real. This meant we never felt the pinch of an adventure expiring and we continued to design around the assumption that timers were good for the player.

When we actually started to play



# INDIANA JONES



them “at pace” we realize that is was pretty punishing to have these timers on the adventures, but we had always counted on them as a part of the core loop and left them in.

**LESSON LEARNED:** If a game is supposed to run for several months, playtest it appropriately—or at least find a way to approximate that length of time and judge the impact.

### 3 / Not Enough Beta

The Beta in IJAW proved to be too short and too early in development. We were still building core features when we entered into Beta, and it only ran for a short period of time. Also, we did not give ourselves enough time after the Beta to really catch our breath and fix any issues we had noticed before we launched. When your deadlines are creeping up, it is easy to say you can cut a week of Beta. Don't do that.

The goals of the Beta test were to see how the game felt when played over a longer period of time, and how the impact of different players' social graphs would change the player experience. The Beta was open to all Zynga employees and we watched the numbers daily to see how players with many friends and neighbors were progressing compared to those who only had a few. We looked at how often people were playing and how long their sessions were to determine if we were delivering enough gameplay for a satisfying experience, and we set up a forum for players

to give feedback and report issues they were having. The Beta was an awesome resource and experience, but it was just not enough.

**LESSON LEARNED:** Allow players to progress through as much of your game as possible in the Beta. Make sure all core systems are in place before the Beta. Schedule a good buffer between the Beta and launch.

### 4 / Content Structure

The game content was initially designed to be similar to a traditional MMO where one new content pack was released every month or two. However, we learned that social gamers wanted to see something new every few days instead of every few weeks, and they prefer light experiences that they know will start at a specific time and last for a known length of time. In IJAW, players wanted bite-sized content, but our game content and story lines were structured more like entrees.

Before launch, we assumed we would release a big new map each week and a new area (composed of another five to ten maps) every six to eight weeks, but that was not enough—our players wanted to get new content each week that they could complete more quickly. So we shifted our pipeline to deliver more maps each week, and spread each new storyline over a few maps. Unfortunately, several user interfaces and flows in the game

were designed around the large-scale content structure, especially the World Map, which is the central hub for all the adventures in the game. It took weeks of iteration and work to get the World Map to match the way players liked to experience content, and it was very difficult to make changes to the launch content to reflect the new structure.

**LESSON LEARNED:** Content for short-session games should reflect the playstyle of the players. Episodic content should be satisfying in as few sessions as possible.

### 5 / The Base Camp

For months we designed the game to take place completely on maps, similar to a traditional linear game. This was a challenge for players because they would return for a session and need to be re-acclimated to the game and the story they were playing. We wrestled with this idea for a while and ended up deciding we needed a place for the player to call home. This is when we decided to create the Base Camp.

Since the Base Camp was developed late in the design process, it never really managed to feel totally integrated into the game. It was a very different style of gameplay from the maps that the players were used to, which was a turn off for them. The Base Camp's main problem was the “Kitchen Sink” problem; our designers pushed too many things into a

single feature to try and solve as many problems as possible.

We wanted the Base Camp to be a consistent starting point for players, a place where they could decorate, a place where they could show off their trophies, where we could implement an appointment mechanic, and provide a place for friends to visit. We ended up solving a few of these but missed the mark on many others. It took months of iteration after the launch to fully deliver on the Base Camp, but we managed to make it a meaningful piece of gameplay for players.

**LESSON LEARNED:** Give added focus to any feature that is trying too much. Spend extra time mending the seams between different styles of gameplay. Notice red flags early and be willing to correct your course.

### A NEW ADVENTURE

With INDIANA JONES ADVENTURE WORLD, we delivered a game that established a new genre in social games, and we are thankful for the fans who love playing the game every day, the opportunity to deliver content for a beloved license, and the industry recognition IJAW has received. I would make this game again if I had the chance, and I expect there to be an expanding market for social adventure games in the future. The reason this game worked at all was because of the commitment of the team at Zynga Boston and their unwillingness to give up. We went through some very difficult times on this game and the team pushed through together with the goal of creating something that everyone could be proud of. I was lucky enough to be a part of that and I can't thank this team enough. 🙌

**SETH SIVAK** was lead designer on INDIANA JONES ADVENTURE WORLD and was named one of the Game Developer 50 in 2011. Seth joined Conduit Labs (which became Zynga Boston) straight out of the Carnegie Mellon University Entertainment Technology Center as a gameplay engineer. While in graduate school he was part of the team that made WINDS OF ORBIS: AN ACTIVE-ADVENTURE, which was a 2009 IGF student finalist.



## Join the Conversation

GAMES ART/DESIGN FILM/TV PRODUCTION RESEARCH  
PRODUCT DEVELOPMENT EDUCATION STUDENT OTHER

Bring your artistic ability, scientific innovation, and everything in between to inspire and be inspired by the most diverse gathering in computer graphics and interactive techniques.

You Are  
**SIGGRAPH**2012



The **39th** International  
**Conference** and **Exhibition**  
on **Computer Graphics** and  
**Interactive Techniques**

**Conference** 5–9 August 2012  
**Exhibition** 7–9 August 2012  
**Los Angeles** Convention Center



Sponsored by ACMSIGGRAPH

[www.siggraph.org/s2012](http://www.siggraph.org/s2012)



## WHAT WOULD

## MOLYDEUX?

## A POSTMORTEM OF ONE OF THE WORLD'S LARGEST GAME JAMS

Game jams are not a new occurrence. They've been helping teams of developers meet each other and make games together (usually in under 48 hours) for years. But given the ease with which we're all connected now, in under three weeks a few people were recently able to create a global game jam, with more than 900 participants across almost 35 cities making more than 300 games. This is the story of how we did it.

Game jams are inspirational. They get people working together in new ways, and give you the creative push you get from a crunch without the fear of losing your job. You can (and you should) participate in or organize these sorts of jams yourself. This is the story of how we did it.

## THE MOLYJAM

Our jam, called What Would Molydeux?, took place on March 31 and April 1, 2012, and it was a surprise success. It began with an offhand question on Twitter from Double Fine gameplay programmer Anna Kipnis, who wondered why there hadn't been a game jam



based on the tweets of Peter Molyneux (FABLE, POPULOUS, BLACK & WHITE) parody Twitter account @petermolydeux. Four of us (Kipnis, Chris Remo of Idle Thumbs, Patrick Klepek of Giant Bomb, and I) took up the banner, and in a short time the jam expanded beyond the original California Bay Area site to the U.K., to Israel, to Mexico, to

Finland, to Australia, and more. Noncentralized developers jammed on their own in solidarity from their homes.

How the heck did this happen? Why was it so successful? What can we do better next time? This abridged postmortem of our event may be helpful for you when planning your own jams, whether they are global or local.

## WHAT WENT RIGHT

## 1 /// Organized like a jam

Everything came together organically, just like in a game jam itself. The initial four of us started out with no concrete idea of what the end result would be, but others stepped up when they needed to. Jake Rodkin of Telltale

Games came up with great T-shirt designs, as well as the "What Would Molydeux?" jam name. Zane Pickett stepped up and created our web site ([www.whatwouldmolydeux.com](http://www.whatwouldmolydeux.com)), and organized our game submission form. Justin Ignacio of Justin.tv came in to help with our livestreams. When something was needed, someone magically

appeared to help.

We also had a number of generous sponsors. Unity, GameMaker, GameSalad, and Construct 2 offered temp licenses of their engines during the jam. Individual cities got sponsors for food, locations, and more. People stepped up to make sure this event happened.

One of the smarter things we did was set a date early on. Not only did April Fools' Day fit with our Molydeux theme, it was a Sunday, so we had the entire weekend to jam. From the start, we had a hard deadline, which helped us get everything ready.

Everyone in our jam was full of goodwill and positivity, especially the organizers who sacrificed their time and energy for the greater cause. Our jam felt like a crackling ball of positive energy! Takeaway: If you're running a jam yourself, make sure you have a good network, and that people have bought in to your idea.

## 2 /// Clear vision

"Peter Molydeux" signed on to the project immediately, which lent the jam legitimacy. The jam was based on his tweets, so the participants were able to reflect on the theme and what they'd like to do in advance. Molydeux curated some of

his best tweets for those who didn't want to sift through everything, and we also got a Google doc of all his original tweets.

We had a solid vision from start to finish, which helped us get things set up. There was no confusion about goals, no back and forth about theme, and no dissent—the reason new cities joined up was because they had already bought in!

The day of the jam, Molydeux made us an introductory video, with closing words from Molyneux himself. Molyneux even showed up at the London jam, gave an opening speech, and participated in the first day.

The Molyjam encapsulates everything Molyneux himself has actually tried to accomplish over the years. Every event seemed in keeping with the spirit of whimsy and emotional game making. The takeaway here is that you should get buy-in from the organizers, whether it's your company or a group of friends, right away. Make sure everyone's committed, knows the vision, and is prepared to follow it through

## 3 /// Popularity

We got a lot of press. People were astounded that it was happening.

It was its own hype machine. Almost every day the core team of organizers would hear from a new city that wanted to join in.

We fielded dozens of press requests—some from outside the industry. While you may not be able to get this level of press support (we didn't expect it), make sure to let other people know you're jamming. It gives you a sense that you really should finish your project, because you're going to show the end results to an excited audience.

#### 4 /// Livestream

The major success in the San Francisco location was the livestream. The teams provided updates on their progress, which kept the audience engaged—somewhat unusual for a jam. That caught on to other locations. “The internet was very supportive of the game developers as they were working,” said Kipnis. “That I did not expect at all.”

Folks watching the San Francisco stream noticed a guy with an orange hat in the jam. He was a musician named Bill Kiley who became a mini-celebrity to the stream, with folks sharing his soundcloud in the chat. The team also interviewed some of the more famous folks in the crowd, and really created a sense of community. Getting people involved in the process is clearly a boon.

#### 5 /// Diversity

I organized the Oakland/East Bay jam, and I felt it was unique because of its diversity. We had straight folks and gay folks. Black people, White people, Asian people, and Hispanic people. We had a very healthy distribution of men and women, including transgender folks. The ages ranged from under 20 to over 40. Not bad for a group of 40 people. The diversity of the crowd was astounding to me, and if the Oakland jam were a microcosm of the industry, problems with gender, race, and orientation in games would be far fewer and further between.

What was it about this event

that brought so many people together? Oakland is a very diverse place, of course. But more than that, we all cared about making games, and the content of those games. The theme of our jam attracted a diverse crowd, and I'd encourage jam organizers to invite folks who would not normally come to your studio or jam to join in. Diverse backgrounds yield diverse ideas.

#### WHAT WENT WRONG

##### 1 /// Organized like a jam

Not one of the main organizers had ever organized a jam. None of us had even been to one (though Kipnis had participated in Double Fine's internal jams). So the fact that we were suddenly at the helm of an international event was a bit of a surprise. We did the best we could, but it was confusing. People asked questions we didn't know how to answer.

Also, we all had our own jobs to do on top of organizing this event. Kipnis in particular put in long hours, and at one point became ill. As the event exploded, I was driving to Phoenix, and could only communicate via smartphone. Since we couldn't anticipate the event's popularity, we couldn't plan for it, and had to scramble to put things together. San Francisco didn't have a venue until three days before the event, for instance. The lesson here is simple—plan your jam well in advance if you can, and talk to people who have organized jams in the past. They will have a lot of excellent advice!

##### 2 /// Fragmented information

We started out asking folks to join on Twitter, then we switched to a Google doc, but that grew unwieldy. We moved the event to Facebook, but not everyone has a Facebook account, so we also had an Eventbrite sign-up. Once sign-ups ballooned to 225, San Francisco and Oakland had to split. Oakland was originally the home of the whole jam, as we had gotten The MADE to agree to host us, but they can reasonably



Peter Molyneux

!! The day of the jam, Molydeux made us an introductory video, with closing words from Molyneux himself. Molyneux even showed up at the London jam, gave an opening speech, and participated in the first day. !!

hold only 50 people. And since San Francisco needed names in advance for security, Eventbrite became the only sign-up that mattered. This happened late in the game, it was hard to let everyone know about the shift, so many folks from the East Bay went to San Francisco, and many folks from San Francisco had to come to the East Bay because they had RSVPed on Facebook but not Eventbrite.

Frankly, the fact that we got venues with projectors, good internet, public transportation access, and late-night access in under three weeks is amazing. This should not be a problem for jams that are planned in advance.

##### 3 /// Internet connectivity

Some locations had spotty internet connections that couldn't consistently stream, which hindered their ability to feel like part of the community. Make sure your connection can support all your game-serving needs as well as livestreaming, and all the programs and assets you may need to download.

##### 4 /// Upload system

It took us a while to solve the problem of how to get so many games out to the public. Molyjammers made hundreds of games, and some were simply too large for our uploader, which broke very early on in the process.

On top of that, it was getting flooded with requests, since people wanted to play the games. Popularity is a great problem to have, but it killed our web volunteers, and in the end, many games missed out on the initial wave of popularity. The takeaway: Just make sure your deployment plan is clear, if you want to make your games public (which I highly encourage).

##### 5 /// Not really 48 hours

Not many venues let us use their space for the whole 48 hours, which is an important part of some jams—you're supposed to be stuck in one space until you finish your game. That could be good or bad depending on your vision for your event, but if you tell people they have 48 hours to finish, or a week to finish, you need to give them that amount of time. In our case, we were limited by the requirements of our space, as well as our ability to feed people—both of which should have been easy to plan for.

#### JAMMING IN THE NAME OF THE BULLFROG

/// The Molyjam experience was difficult but rewarding, and I encourage others to try to make their own jams, regardless of their scale. Even if nobody knows your jam is happening, going through it can really help you prototype an idea, get a feature together, or meet new developer friends. So get to it! Happy jamming! 🐸

**BRANDON SHEFFIELD** is editor-in-chief of Game Developer magazine, and owner/creative director of game developer and consultancy group Necrosoft Games ([www.necrosoftgames.com](http://www.necrosoftgames.com)).

# GET YOUR GAME ON!

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

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



### DIGITAL COUNSELOR

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

[www.gamecareerguide.com](http://www.gamecareerguide.com)



Download your FREE digital edition of the 2011 game developer Game Career Guide online at: [www.gamecareerguide.com](http://www.gamecareerguide.com)



# GDC Vault

THE BEST ON-DEMAND CONTENT FROM THE GAME DEVELOPERS CONFERENCE SHOWS

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

**STUDIO AND EDUCATION GROUP RATES AVAILABLE!**

For more information visit: [www.gdcvault.com](http://www.gdcvault.com)



# PIXOLOGIC ZBRUSH 4R3

REVIEW BY MATT LINK

It's no secret that ZBrush stands at the forefront when it comes to making high-poly models and 3D illustrations. But competing tools like Autodesk's Mudbox are making significant improvements with every release, so I thought I'd take a look at what Pixologic's ZBrush 4R3 does to stay at the front of the pack.

## MAKING MOUNTAINS INTO MOLEHILLS

» ZBrush has a steep (though relatively short) learning curve, in part because its roots as a 2.5D program means it still carries several idiosyncrasies in its core—none of which are changed in ZBrush 4R3. For example, most 3D applications give you a standard 3D viewport, but ZBrush's viewport is a resolution-dependent document. It may look like you are zooming in to your model, but you are actually scaling the model within the document. To save or export something in ZBrush, unlike other programs, you must make sure you save or export the "Tool" and not the "Document." This "Tool" concept alone is likely to

be the most painful lesson new ZBrush users will learn firsthand—most likely the hard way, when they open a saved "Document" and find it empty. You'll eventually get used to these little oddities, but it's not a pleasant process.

One thing that helps is Pixologic's community support forum, where users can share their art, as well as tips and tricks. Pixologic also supplies invaluable learning material on its ZClassroom site ([www.pixologic.com/classroom](http://www.pixologic.com/classroom)). ZClassroom contains in-depth tutorials, walkthroughs on new features, and other instructional videos.

It's a lot of work at first, but it's worth it. Once you can navigate the viewport and quickly wade through

dozens of menus, you will be creating more detailed work faster and more efficiently than you could ever imagine with standard 3D applications. ZBrush is that good.

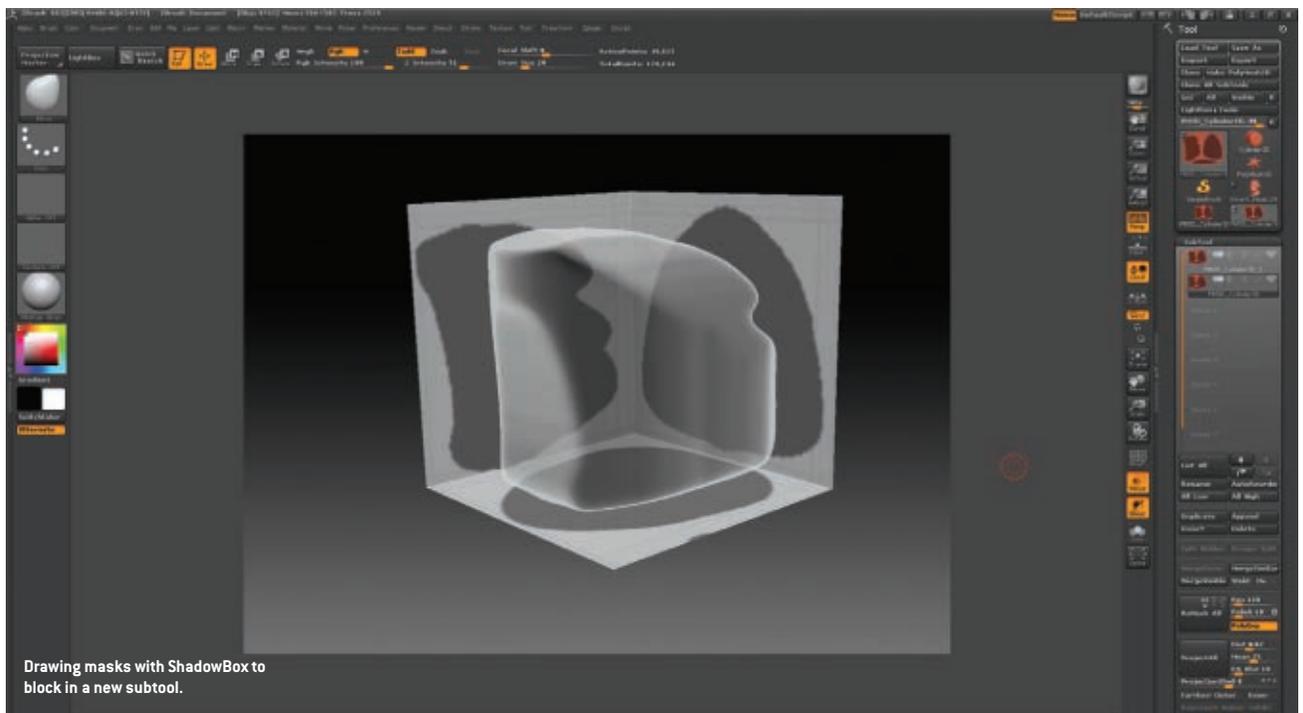
## THE INTERFACE

» ZBrush's interface is completely customizable and comes with a handful of preset modes. A properly customized interface can really speed up your workflow, so it's worth spending the time to tweak it. You can remove buttons for features you rarely use and replace them with those you use more frequently, create your own hotkeys and menus for items you often switch between like brushes or materials, and create useful macros and place their shortcut in

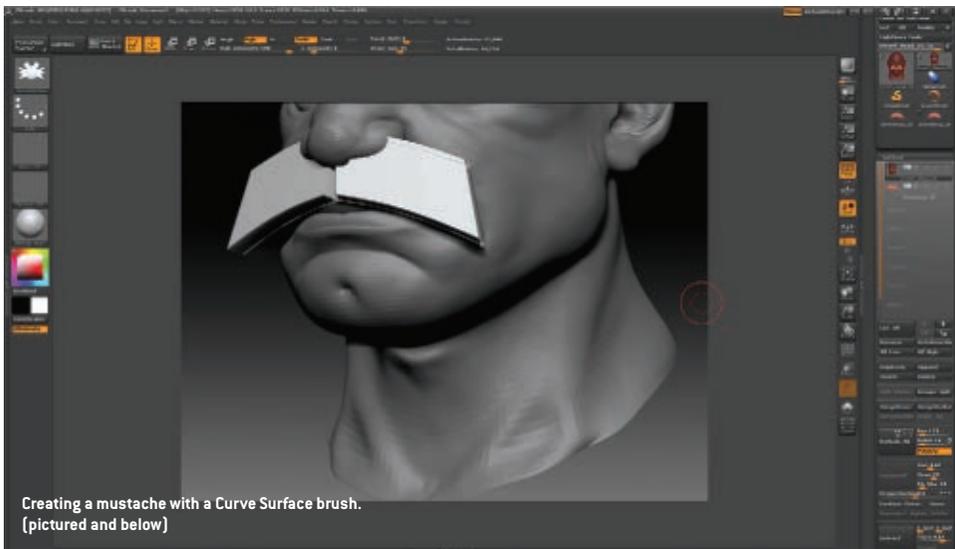
a convenient location. If there is a tool in the interface you would like to know more about, just hold the Control key and mouse over the button, and a pop-up explains what the tool is and how to use it.

With the R3 update, you can now collapse subpalettes in the menus, which was a popular request from previous versions. This helps keep certain absurdly long palettes under control, since you can pick which sections are expanded. It's not perfect, though—since the subtool palette is limited to displaying only eight subtools at a time, you'll usually need to Alt-click the subtool you want to use.

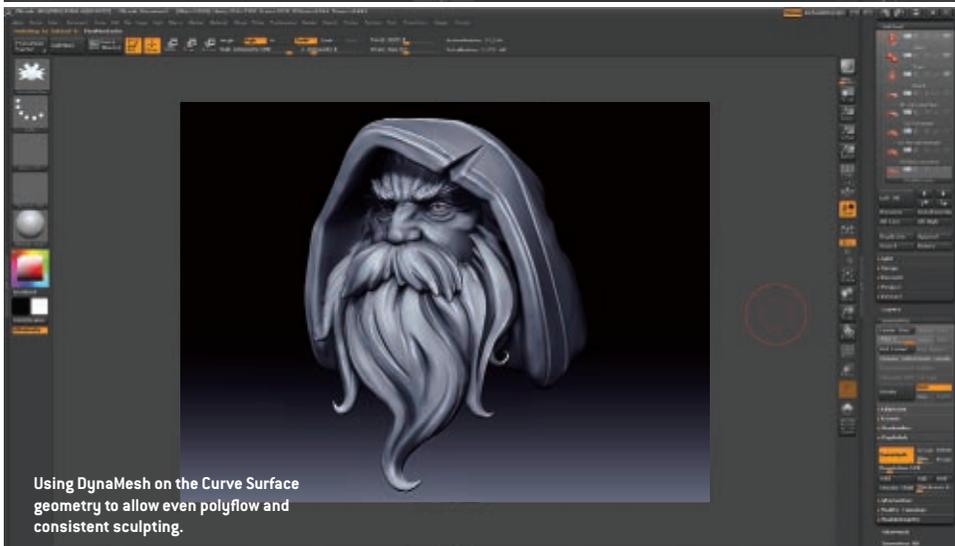
Pixologic also changed the way you assign your own hotkey from a Control-click to a Control-Alt-click



Drawing masks with ShadowBox to block in a new subtool.



Creating a mustache with a Curve Surface brush. (pictured and below)



Using DynaMesh on the Curve Surface geometry to allow even polyflow and consistent sculpting.

## PIXOLOGIC ZBRUSH 4R3

[www.pixologic.com](http://www.pixologic.com)

### PRICE

> \$699

### SYSTEM REQUIREMENTS

- > Windows Vista/Windows 7 32-bit/64-bit (ZBrush 4R3 is a 32-bit application, and may use up to 4 GB of RAM).
- > Core 2 Duo or newer (or equivalent such as AMD Athlon 64 X2 or newer) with optional multithreading or hyperthreading capabilities
- > 1 GB RAM minimum, 6 GB recommended
- > 1280x1024 monitor resolution or higher

### PROS

1. One of the most powerful, feature-packed, polished 3D sculpting applications on the market.
2. Excellent technical and community support with countless tutorials to learn from.
3. Competitive price, no charge for upgrading to new versions.

### CONS

1. Only 32-bit so it can't take advantage of all the RAM workstation computers might have.
2. Occasional stability issues which can lead to crashes and lost work.
3. Fairly steep learning curve which can be intimidating to new users.

with not so much as a mention of the change, which was kind of annoying at first. Unfortunately, this just comes with the territory of a full-featured program.

### NEW IN 4R3

» One of the most notable additions to ZBrush 4R3 is the DynaMesh tool. With DynaMesh activated, you can push and pull the mesh to the point of seeing its polys stretch and distort, and when you Control-click-and-drag in empty viewport space, your mesh will adapt to allow even poly coverage to the entire mesh, giving you a new surface optimized for sculpting. Combine this with the clay brushes and you can iterate very quickly while you explore your sculpting possibilities. This way, your creations evolve as you go. I've seen artists create entire characters, creatures, and trees starting from a simple sphere and DynaMesh. This helps keep you in a creative state rather than feeling limited by your initial base mesh.

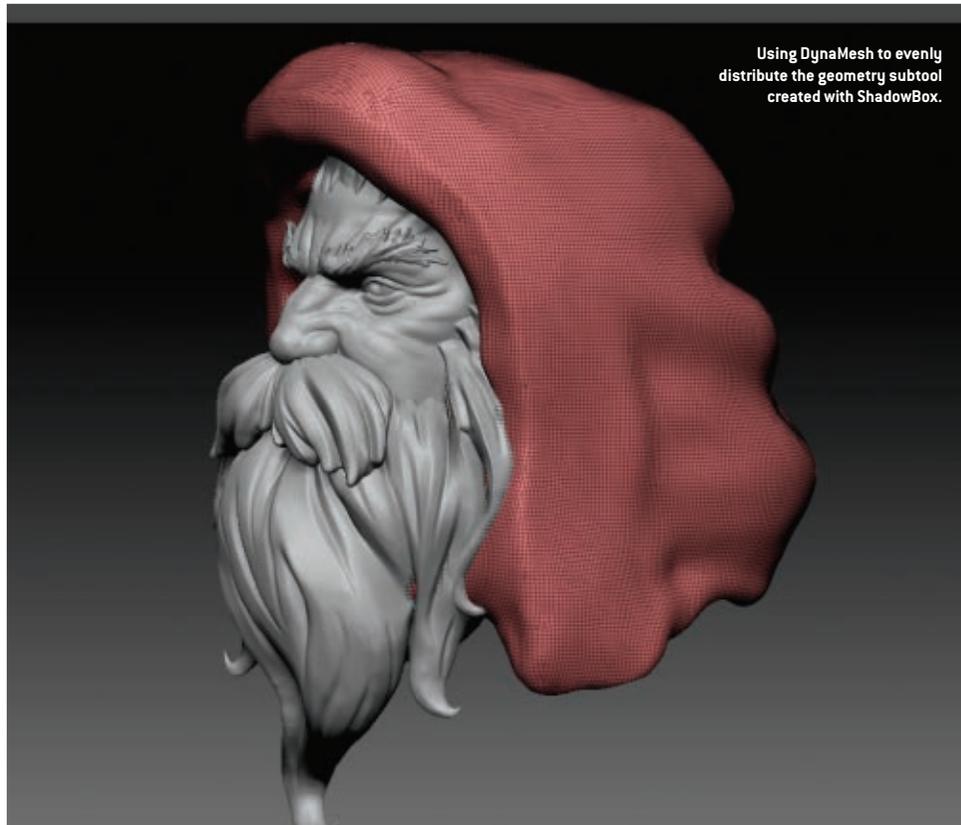
ShadowBox is another new tool that allows you to draw out masks on

any of the three sides of the “shadow box” and create 3D geometry from the intersecting shapes you’ve drawn. This is a useful way to rough out a shape to begin sculpting on or create complex hard surface models that would be more difficult through traditional modeling.

Pixologic has also added several new brushes in ZBrush 4R3, and it’s worth your time to try them out and get a feel for what each of them do—you might find some new favorites. They have even added the clip and trim brushes, which are geared more toward hard surface sculpting. The new stroke type, curve, allows you to draw out a spline to control your stroke. As you draw it out, every time you tap the Alt key, a new point appears on the spline allowing you to change direction and continue to make it more intricate. Using the Slice Curve brush, you use the curve stroke type to draw out shapes that will separate your mesh at the desired spots into different polygroups.

GoZ is another new feature that sends your model to your chosen 3D package, where you can add or edit topology and make desired changes to your mesh, then send it back to ZBrush and continue sculpting on your new improved mesh. GoZ also supports the extended version of Photoshop CS4 or CS5, so you can paint it on your texture and model in Photoshop and then send your model back to ZBrush. Unfortunately, when we tested GoZ, Photoshop ran far too slowly to get any usable work out of it. For reference, we tested ZBrush 4R3 on a PC with an Intel Core i5 CPU, 8GB of RAM, an Nvidia GTX 560 graphics card, and a Western Digital Raptor 15,000 rpm hard drive. When we brought this up to the ZBrush developers, they maintained that the sluggish performance was due to Photoshop’s poor handling of 3D.

Would-be Mudbox converts should appreciate the vector displacement tool, which lets you displace a surface in any direction (including undercuts). It’s not quite groundbreaking—vector displacement has been in Mudbox for several years now—but if you have been tempted to switch to Mudbox for this sole reason,



Using DynaMesh to evenly distribute the geometry subtool created with ShadowBox.

there’s no need. ZBrush has finally implemented a layer system that supports sculpting and polypainting. After sculpting in a layer, you can drag the slider down to lessen the extent of your changes. I find this very useful, as I can oversculpt an area and then pull it back to where I want it. You can also toggle the layer visibility on and off to get a quick before-and-after comparison. Once you have created a layer, you can no longer edit that subtool unless you are recording in another layer. This can become irritating if you are frequently switching between subtools that contain layers and you just want to make quick changes.

Last up is the new FiberMesh, which lets you add fibers to your model and adjust them with sliders. You can then use the new groom brushes to get them into the exact shape and style you want. When you create new fibers, the base will be masked by default so you don’t move it off your model while grooming. Use this with the new MicroMesh best preview rendering feature and the fibers can act as a placeholder

for a specified mesh until it’s time to render. FiberMesh is much more fluid and responsive than I expected it to be, and I’m looking forward to new creative uses for fibers that will hopefully be available for download on Pixologic’s ZBrush Central forums ([www.ZBrushcentral.com](http://www.ZBrushcentral.com)).

#### TOOLS FOR THE JOB

» Thanks to the clever way ZBrush displays polygons, you won’t need a high-end video card to use the program. You will probably want a pen tablet (a Wacom tablet, for example), but it isn’t strictly necessary, though the tablet does let you control ZBrush better (and honestly, without the pressure sensitivity, you might as well be painting a wall with a rock).

#### UP AGAINST THE COMPETITION

» There’s no doubt that ZBrush is very powerful, but it’s not necessarily for everyone. Autodesk’s Mudbox (\$745) is easier to use for someone new to sculpting applications, and it shares certain user interface elements with Maya (the viewport

navigation is the same, which is nice). Mudbox’s layer system far and away beats ZBrush’s. There’s also 3D Coat (\$349), a slightly cheaper competitor that is built around voxel-based sculpting. It has very powerful retopology tools, excellent texturing tools, and a good layer system, but the voxel sculpting still feels a bit clunky and unrefined in comparison to Mudbox and ZBrush.

Ultimately, neither Mudbox nor 3D Coat can hang with ZBrush’s community, support, efficiency, fluid interface, or ability to handle millions of more polygons at a time.

Also, once you purchase a ZBrush license, all subsequent updates are free.

In my opinion, ZBrush is absolutely worth the price, bugs and quirks aside. It’s an industry standard for a good reason. 

**MATT LINK** works at Gearbox Software and has been involved in creating characters for several next-gen games. He is currently working as a character artist on *BORDERLANDS 2* and *ALIENS: COLONIAL MARINES*. email him at [matt.link@gearboxsoftware.com](mailto:matt.link@gearboxsoftware.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](http://GDMAG.COM/SUBSCRIBE)

DOWNLOAD THE GAME  
DEVELOPER APP

[bit.ly/gdmag\\_ios](http://bit.ly/gdmag_ios)



# gd

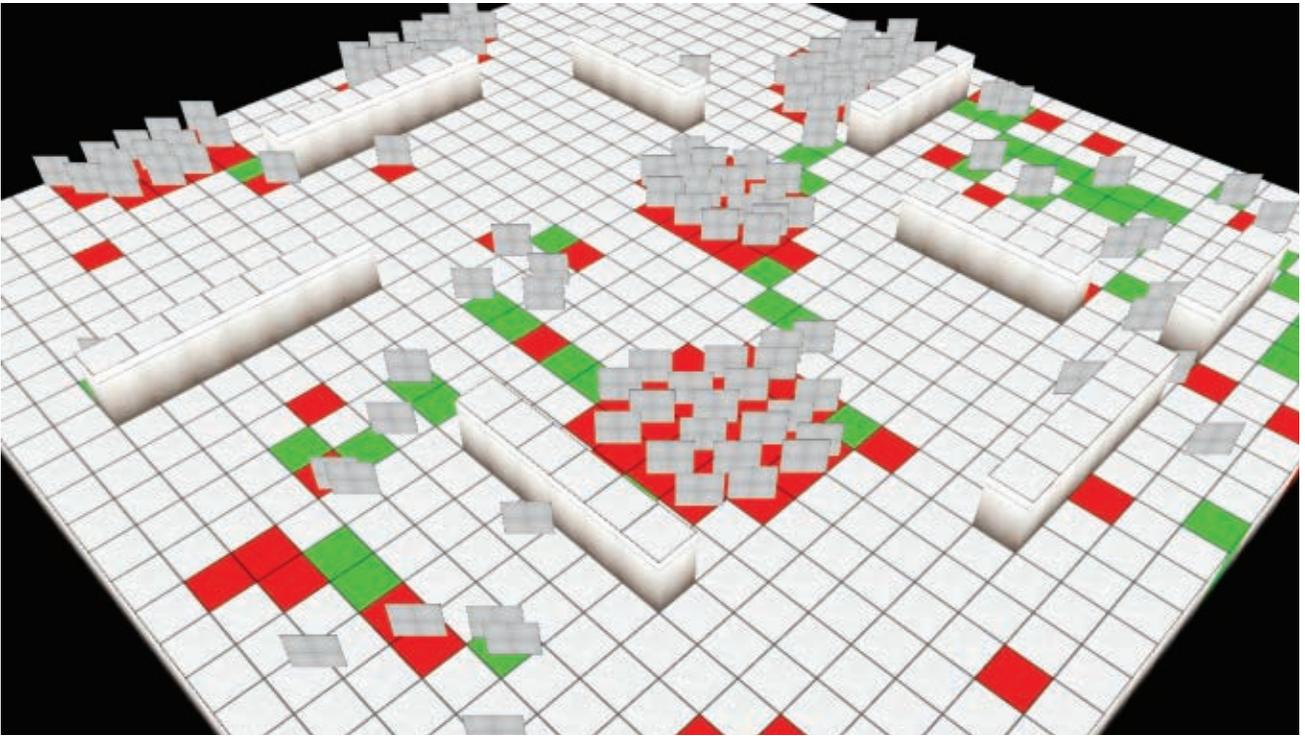


FOLLOW GAME DEVELOPER



# CELLS FOR CELL PHONES

USING CELLULAR AUTOMATA TO SIMULATE CROWDS FOR MOBILE GAMES



Plenty of action films feature scenes that simulate interactions of hundreds or thousands of creatures—a horde of zombies, for instance, or an enemy army massing on the horizon—and when it comes to video games, we want similar experiences. But simulating crowds isn't easy, (especially on low spec-devices) if we want each individual actor in a crowd to look and behave slightly differently from his neighbors. However, with the Cellular Automata model (CA) and a few grouping rules, we can model a crowd of hundreds of locally acting bots that can organize themselves in groups and attack the player—and we can do it on a simple cell phone. We'll show you how we designed a prototype Cellular Automata AI, and see how well it runs on both an Android tablet and an older cell phone.

## CELLULAR AUTOMATA AND FLOCKING

» Cellular Automata methods were developed to simulate and analyze complex interactions through relatively simple rules of intercell interactions. They are discrete dynamic systems that specify behavior completely in terms of local relations. The CA space is represented with a uniform grid that contains a limited amount of information in each cell. Time advances in the simulation in discrete steps, and the transformation rules are usually expressed with a simple

recipe. Given the right recipe, we can model a whole hierarchy of structures and phenomena.

Our simulation procedure is based on a limited region of space that forms a mesh of cells. A number of objects is placed either randomly (with a given distribution) or manually. Each cell is empty or occupied by an object. In the simplest case, the cell is emptied or filled with a selected object with some probability. The final structure is obtained after several iterations, where each cell of the mesh is checked. When we permit interactions between

cells, we can obtain simulations of ecological models.

Cellular Automata can simulate large numbers of entities, but the interaction is always a matter of neighboring cells. Groups existing in nature (such as a flock of birds or a school of fish), however, can be simulated with an elaboration of a particle system that shows each individual member of the group as a particle and uses an augmented behavioral model to simulate each particle's motion.

Essentially, each simulated element of a flock is designed as an independent entity that moves

according to its local perception of the surrounding environment, the laws of simulated physics, and a set of preprogrammed behaviors. The composite motion of the simulated flock of NPCs is the result of the cross-interaction of the relatively simple behaviors of an individual simulated NPC. In our case, we used our crowd simulation to build a simple game about zombies.

## DERIVING THE CA AND FLOCKING ALGORITHMS

» We start building our CA-based game logic by grouping cells of the same characteristics augmented

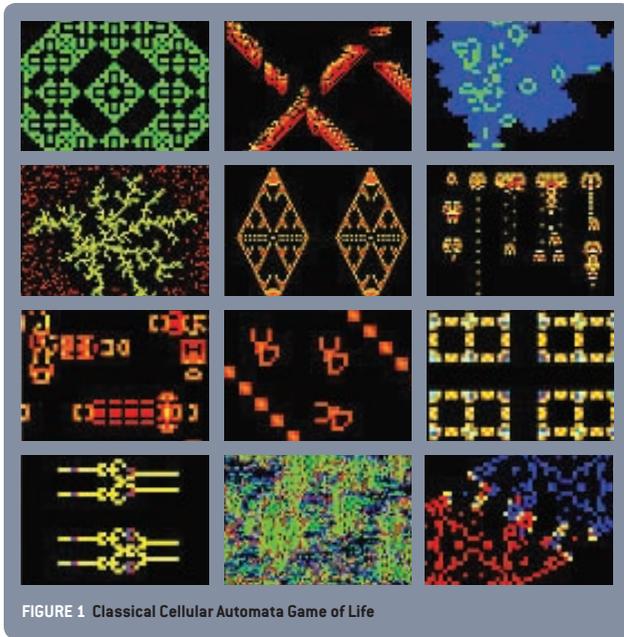


FIGURE 1 Classical Cellular Automata Game of Life

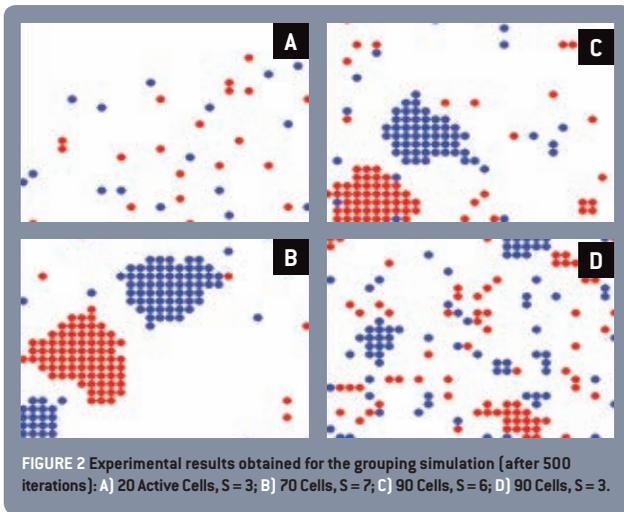


FIGURE 2 Experimental results obtained for the grouping simulation (after 500 iterations): A) 20 Active Cells, S = 3; B) 70 Cells, S = 7; C) 90 Cells, S = 6; D) 90 Cells, S = 3.

with the "infection" capability. The result of this iteration is processed by another cellular automaton equipped with rules that can trigger large groups of NPCs to attack. We use three general rules to govern the simulation:

→ The grouping rule, which is based on the Moore neighborhood (a cell's eight neighboring cells).

→ The infection rule: If there are two zombies in the Moore neighborhood of a human, they infect him and turn him into another zombie.

→ The attack rule: When the group of zombies is sufficiently large, it can attack the player's avatar. In the counting process, a neighborhood similar to the Margolus neighborhood is used.

GROUPING AND INFECTION PROCESSES

» Cellular Automata operates on a two-dimensional grid size n x m (to simplify n = m). Cells can have any number of states s. The border conditions are described as periodical, and we use the Moore neighborhood (excluding the central cell).

The following is our automaton rule. One iteration consists of repeated n² following steps:

- 1. Randomly pick cell c, with less than S neighbors with similar states.
2. In the vicinity of c, randomly choose neighboring cell n.
3. If C's cell state is equal to n's cell state then go to step 6.
4. If cell n is empty (state = 0), determine the growth factor (g): The number of neighboring cells with cell c's state on location n, minus the number of neighboring cells with cell c's state on location c. If g >= 0, then c is "moved" to n (change c's cell state to 0, and n's cell state to c's cell state) and go to step 6.
5. Count the growth factor g by taking the number of cells neighboring c on location n and subtracting the number of neighboring cells to n on location c. If g != 0, then swap cells n and c.
6. If C is "human" and N has G neighbors, then cell C changes its state to N. In other words, C has changed into a zombie. The value of G determines how easy or hard it is to infect a human.

PERFORMING ZOMBIE ATTACKS

» Zombie "attacks" are performed by moving the group of infected NPCs toward the player's avatar. If the player touches a given number of zombies (growth factor G), the player is infected. In order to trigger a zombie attack, the group of zombies should be large enough to successfully perform the assault.

The automaton working on the output of the grouping algorithm from the previous section estimates the zombie group count. First, it defines a neighborhood similar to the Margolus neighborhood, though the number of cells (9) is larger than the original (4) version. The size of zombie groups can be changed according to a given game level [with the parameter S discussed in the next section]. The automaton can set the preferred direction of

movement for group cells.

We tested this algorithm in a prototype test application to experiment with an appropriate s value. s determines the minimum number of neighboring cells counted in the grouping process. The most interesting results were obtained for s=3 (after 500 iterations), as illustrated in FIGURE 2.

The grouping procedure works when the number of active cells is larger than 40. A smaller number of cells results in chaotic behavior (FIG. 2A). For the larger number of cells, the influence of s on grouping process is clearly visible. Smaller values of S result in numerous but smaller flocks of cells (FIG. 2D), while larger values of s (6 or 7) end up in two big groups of each type of cell (FIG. 2B and 2C).

BUILDING THE GAME PROTOTYPE

» Next, we created a set of rules for a prototype game.

→ The main task of the player is to save a given number of humans from infection within a time limit.

→ Zombies can attack the player and his fellow humans and form "flocks" to increase their strength.

→ Individual zombies try to group and only attack when the player shows no apparent activity.

→ When the player's avatar contacts a zombie, the player loses life. Run out of life, and it's game over for the player. When a given number of zombies contact humans, the humans turn into new zombies.

→ The player can refill his life by picking up the "med-aid" items that appear in various places.

→ The avatar has two weapons. The first weapon can be used to destroy other game actors, but has limited ammunition scattered across the game area. The second weapon can turn zombies back into humans, and has unlimited ammo.

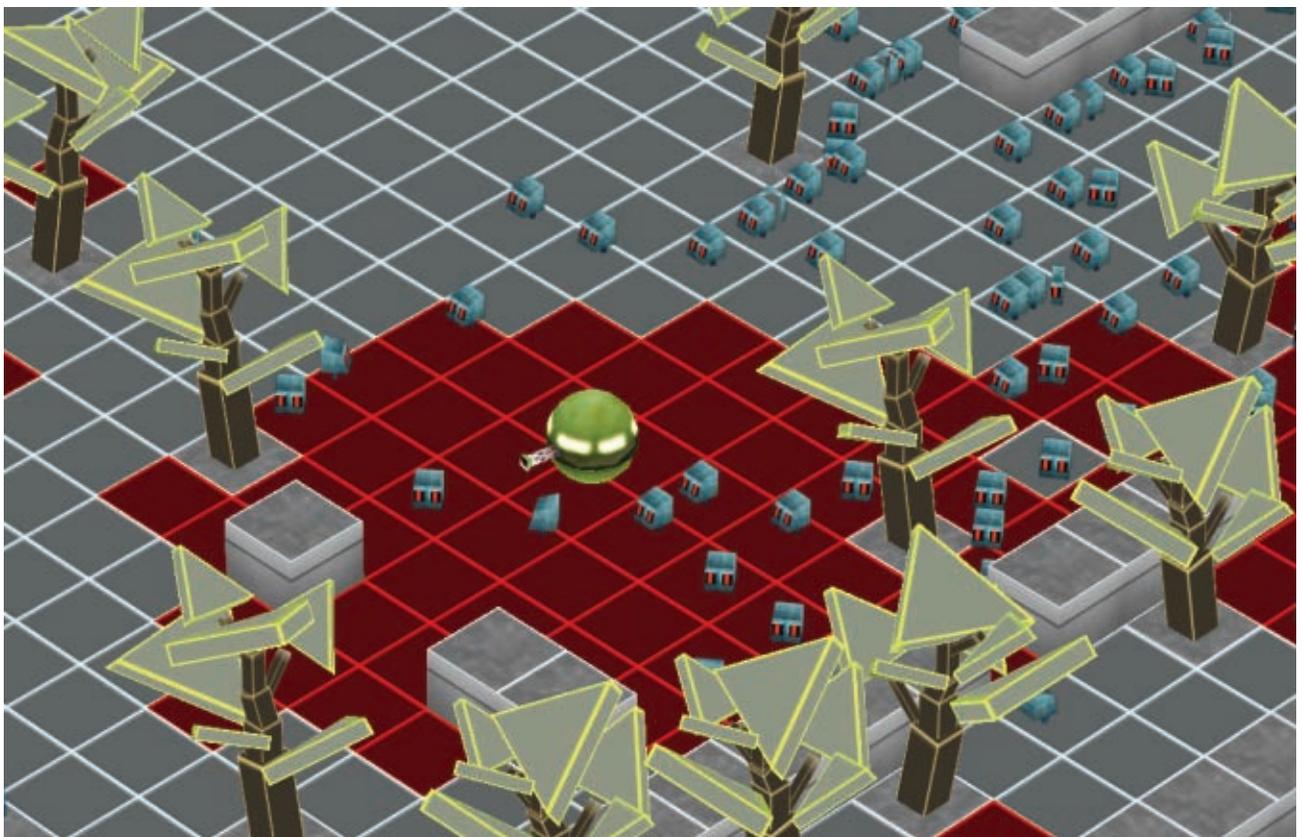
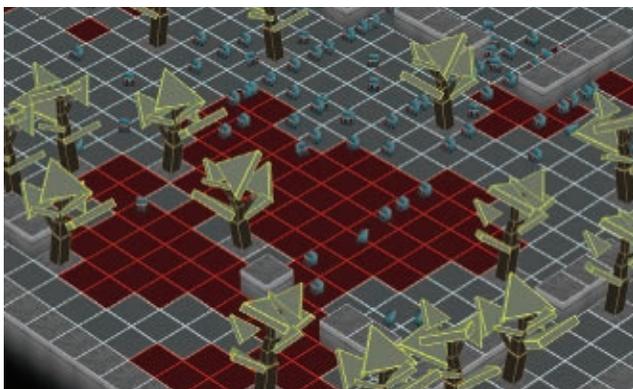
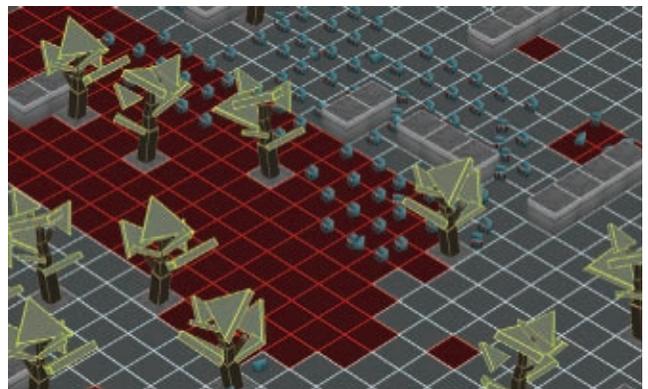


FIGURE 4: The game prototype built for an Android tablet.



## No zombies and humans

Map	30x30	60x60
No.	FPS	FPS
1	56.70	54.60
2	56.34	55.09
3	55.27	55.03
4	57.00	55.10
5	55.90	55.15
AVG.	56.24	54.99

## 20 zombies and 5 humans

Map	30x30	60x60
No.	FPS	FPS
1	45.01	44.99
2	45.13	45.54
3	45.90	45.01
4	45.67	45.49
5	45.78	45.37
AVG.	45.50	45.28

## 40 zombies and 10 humans

Map	30x30	60x60
No.	FPS	FPS
1	34.67	34.86
2	34.55	34.87
3	34.70	34.57
4	34.23	34.01
5	34.17	34.17
AVG.	34.46	34.50

## 80 zombies and 20 humans

Map	30x30	60x60
No.	FPS	FPS
1	24.96	24.39
2	25.01	24.45
3	23.91	24.58
4	24.78	24.10
5	24.59	24.17
AVG.	24.65	24.34

→ When a human is destroyed, the number of humans the player is required to save increases.

→ There is a teleporter in the game area, which teleports the player to a random game spot.

### RUNNING THE SIMULATION

» We built our prototype application in Java and ran it on a Sony-Ericsson K300i cell phone, with CLDC1.1 and MIDP2.0, a 30MHz Java processor (equivalent to Intel Pentium III 540MHz), 512KB stack RAM, and a 128x128 pixel display.

The prototype application managed to sustain very good frame rates during the simulation. It's worth noting that there were no visible differences for different RAM sizes (128KB and 512KB).

### GAME LOGIC

» The game logic based on our straightforward implementation of CA has a few inefficiencies. First, randomly choosing a cell on the CA grid with a large number of empty cells is a waste of processing power. This can be fixed by choosing a cell from a list of non-empty cells instead of choosing from the entire grid. Also, there is a chance that the same cell could be chosen several times during a single iteration (the "Lévy flight"), which could be fixed by attaching an information index with movement-limitation rules to active cells for each iteration.

Another drawback of the prototype implementation is the discreet movement of NPCs on the game arena, which is caused by the straightforward implementation of the CA grid to the game space, which could be fixed by handling action calculations offscreen and animating the moving NPC fluidly onscreen, as it is performed in the Android version (described in the following section).

### ANDROID VERSION

» The Android OS core limits the screen's refresh rate to a maximum of 56 to 60 frames per second (depending on the software version). In order to keep the animation smooth, the CA logic-simulation algorithm needs to be run as a separate thread. Another way to handle this problem is to bind steps of the CA algorithm to the rate of iterations per second. At each game loop cycle, the number of steps is estimated according to the length of time that has passed since the last loop. One second is then split into  $n^2$  iterations.

Jakub Grzesik performed a sample simulation on the Archos G9 Android tablet (<http://www.youtube.com/watch?v=aQqQOMM8aqI>), shown in **FIGURE 4** and on YouTube. The test app was designed with libGDX graphics library, (<http://code.google.com/p/libgdx/>).

Here, we can see the grouping formations and pathfinding processes at work. The cellular automaton sets the target position, and then the A\* algorithm is used to

find the shortest path. (The colors are used for the debugging process.)

We can clearly observe zombies gathering behind obstacles. Sometimes, individual zombies will leave the original group, wander around the area, and eventually join the other group. From the player's point of view, this kind of behavior appears to be an "intelligent" action, such as scouting out the map or deserting a group to improve its odds of survival. This behavior is controlled by the  $s$  parameter (described in earlier sections). As the  $s$  value increases, more individuals will leave their groups. This behavior is also influenced by the individual's proximity to obstacles (walls), which appears to a player as though some individuals try to use the walls for shelter.

### EFFICIENCY ANALYSIS

» The CA algorithm is able to simulate large number of cells on the grid. We tested the algorithm's performance by measuring our prototype Java application on the Sony Ericsson K300i mobile phone with several different map sizes and unit compositions. The results can be seen in Table 1. The Android version, meanwhile, runs at the maximum speed of 56–60 fps, though the Archos G9's memory restrictions won't let us build a map larger than 150x150. With large, crowded levels, the Android version's performance drops down to 8 fps on a 100x100 map with 2000 NPCs and

0.25 fps on a 150x150 map with 3000 NPCs.

### CELL YOU LATER

» By building bots that interact on a local level, we can create massive hordes of "intelligent" bots. Our method lets players interact with hundreds of NPCs, extending the programming possibilities for action, role-playing, and survival games. We plan to further develop this method in order to enrich a player's experience—for example, by using this implementation to make more demanding first-person shooter games.

**NOTES** Test results are part of M.Sc. diploma theses by Michał Jackowski, "Implementation of CA model to game logic in mobile systems, and by Jakub Grzesik titled "Cellular automaton implementation in 3D game logic," University of Zielona Góra, Poland.

### REFERENCES Moore Neighborhood:

([mathworld.wolfram.com/MooreNeighborhood.html](http://mathworld.wolfram.com/MooreNeighborhood.html)) Margolus Neighborhood: (psoup.math.wisc.edu/mcell/rullex\_marg.html) Levý Flight: (www.maths.qmul.ac.uk/ffklages/bee\_wshop/bbees\_checkkin.pdf)

**SLAWOMIR NIKIEL** is currently the professor at the Institute of Control and Computation Engineering, Department of Electrical Technology, Computer Science and Telecommunication, University Of Zielona Góra, Poland. His research interests include virtual reality systems, game programming, and multimedia.



**GAME DEVELOPERS CONFERENCE™ EUROPE**

COLOGNE, GERMANY  
AUGUST 13-15, 2012

**2012**

[WWW.GDCEUROPE.COM](http://WWW.GDCEUROPE.COM)



# THE FIVE DOMAINS OF PLAY

## MAPPING THE FIVE-FACTOR MODEL IN PSYCHOLOGY TO GAME DESIGN

Over the last twenty months, I've been trying to draw correlations between the Big 5 motivational factors and game design elements that cater to those factors by interviewing any game player willing to take a test about their play behavior. In essence, I wanted to translate the work of motivation psychologists into game design—and I managed to draw a few correlations sooner than I had expected.

### OPEN SOURCE PSYCHOLOGY

» Before we get our hands dirty, I want to mention a few things about why the Big 5 system is different from other systems.

For starters, the Big 5 system doesn't come from a single person—it is an international collaboration between dozens (hundreds?) of researchers. Instead of keeping the data driving the discoveries copyrighted and secret, or preventing correlative studies with other systems, the data behind the Big 5 system was subjected to every imaginable cross-analysis—which is ongoing, and will be into the foreseeable future. And while the contributors could have owned the discoveries and charged for their use, they decided to release the entire thing into the public domain. In other words, the Big 5 system is the Linux of motivation psychology.

### BELLS AND CURVES

» When you take the Big 5 test (the best free one I've found:

[www.personal.psu.edu/ffj5j/IPIP/](http://www.personal.psu.edu/ffj5j/IPIP/)), you get a report that shows where you fall in five personality "domains," as well as in the individual character "facets" that make up those domains. Each of these domains is defined as having a standard distribution (read: bell-shaped curve) when applied across humanity. A low score or a high score means that for that particular facet your motivation is a rarity, and a score in the middle means that for that facet your motivation is similar to a majority of the population.

This means we have a statistical baseline for any analysis we'd like to do, which is a Very Good Thing—as we apply this system to game development, we can swap out vague assertions like "most gamers want X" for factual statements like "half the human population has a preference for X, and the other half for Y." Even better, we can begin to accurately measure different player populations. Have you ever wondered whether the "core gaming" population is statistically different in its preferences than the rest of the

world? Well, now we're one detailed study of players away from having an answer.

### TWO SIDES TO EVERY STORY

» Each domain, or "factor," is a two-sided spectrum with a positive motivation on each end.

This may seem obvious at first: Some people are open to new experiences, and others less so, for example. Well, compare that two-sided structure to the commonly-held-in-the-game-biz archetype of the Achievement Player. What is the positive opposite of an Achievement Player?

The developers I have worked with tend to talk about Achievers like this: If your game satisfies Achievers, you'll probably get those players to play your game. If your game doesn't satisfy Achievers, you'll fail to attract those players, and by implication you make less money.

I have started calling this way of looking at players as the "thermometer model" of motivation—you stick a thermometer in the game and you measure its "Achievement-ness."

LITTLEBIGPLANET appeals to players that value harmony.



High is good, low is bad. But from what I have learned so far, that view is completely wrong.

The opposite of an "Achievement player" is a "contentment player:" someone who is perfectly happy to ignore your target goals, difficulty challenges, and medals, and just hang out. Someone who is motivated to be content with their current state and who will buy games that let them act on that motivation.

Remember the bell-shaped curve discussion above? We're talking about 50 percent of humanity being on the "contentment" side of the curve. That's an awful lot of players who are not being discussed in most design meetings, thanks to a simple misunderstanding of how player archetypes work.

### SWIMMING IN THE O.C.E.A.N.

» Let's get down to business. The Big 5 are: **Openness to Experience**, **Conscientiousness**, **Extraversion**, **Agreeableness**, and **Neuroticism**—"O.C.E.A.N."

**Openness to Experience** distinguishes creative, intellectual folk from down-to-earth, pragmatic ones. A high scorer would be **Alice In Wonderland**, who is happy to drink whatever she comes across and follow rabbits into the unknown. Alice finds Wonderland a delight. A low scorer would be **Samwise Gamgee**, who just wants to go home, have a predictable life, and not bother about with wizards quite so much.

**Conscientiousness** deals with our ability to control our impulses and order our world the way we want it. A high scorer would be **Hermione Granger**, who is the best in her class at almost everything, and who is the one you go to when you need to get something difficult accomplished. A low scorer would be **Jeff "The Dude" Lebowski**, whose primary ambition in life is to bowl, and who can turn nearly any simple outing into a near-complete disaster.

**Extraversion** deals with the desire for external stimulation, both social and otherwise. A high scorer would be **Austin Powers**, who is always ready to party, much prefers the company of others to solitude, and is the leader of the pack. A low scorer would be **Edward Scissorhands**, who is happy to do whatever you want, please, just leave him alone, in the dark.

**Agreeableness** deals with cooperation and social harmony. A high scorer would be **Charles Xavier**, who puts the needs of others ahead of his own, believes in the good in people, and who understands how you're feeling better than you do. A low scorer would be **Snake Plissken**, who, if you want him to care about another human being (like the president), you have to inject explosives into his neck that will detonate if that person dies.

**Neuroticism** reflects how strongly one experiences negative (and only negative) emotions. A high scorer in Neuroticism would be **Woody Allen** (the character, not the man), for

whom the world is a panoply of fears, anxieties, angers, and frustrations. A low scorer would be **Obi-Wan Kenobi**, for whom fear, anger, and jealousy lead to the dark side of the Force, and who met his death with a polite salute.

### FACETS WITHIN FACTORS

» Your score in each domain in the Big 5 is actually something like a weighted average of your score in six "facets" that describe specific preferences within that domain. For example, Openness to Experience is composed of Imagination, Artistic Interest, Emotionality, Adventurousness, Intellect, and Liberalism; Conscientiousness includes Self-Efficacy, Organization, Dutifulness, Achievement-Seeking, Self-Control, and Cautiousness; and so on. I won't list them all exhaustively—take the test yourself, read the Wikipedia page ([http://en.wikipedia.org/wiki/Big\\_Five\\_personality\\_traits](http://en.wikipedia.org/wiki/Big_Five_personality_traits)), or just Google "Big 5 facets" to see them all.

As you will see, many of these facets describe polarities that game designers already use. These facets are where the proverbial rubber meets the road for game design.



### THE FIVE DOMAINS OF PLAY

» With the psychology described, we can go looking for game elements that will satisfy these motivations. In order to do this, I have been interviewing gamer players after they take the Big 5 test, with the idea that people with a similar score in specific facets should theoretically prefer similar games or game elements (such as PvP, achievements, grouping, and so on).

My database isn't complete, but I have uncovered strong evidence for many direct associations between the Big 5 facets and game

elements. The notable exception is Neuroticism, which refuses to produce predictable correlations so far. That said, based on the data so far, translating the Big 5 into game elements gives us these five domains of play: **Novelty**, **Challenge**, **Stimulation**, **Harmony**, and **Threat**.

**Novelty** (which maps to **Openness to Experience**) is the presence or lack of new, interesting, dramatic, or beautiful things in the game. A high Novelty game would be **MINECRAFT**, and a low Novelty game would be **FLIGHT SIMULATOR**.

**Challenge** (which maps to **Conscientiousness**) is the part of the game that requires the player to use self-discipline: overcoming obstacles, work, avoiding danger, and (literally) collecting achievements. A high Challenge game would be **SPLINTER CELL**, and a low Challenge game would be **LEGO STAR WARS**.

**Stimulation** (which maps to **Extraversion**) is the part of the game that excites, be that through direct thrill-rides or through social interactions. A high Stimulation game would be **JUST DANCE**, and a low Stimulation game would be **FLOWER**.

**Harmony** (which maps to **Agreeableness**) is the part of the game where the player behaves

in a particular way toward other people or characters. Do you shoot them? Or help them? A high Harmony game would be **LITTLE BIG PLANET**, and a low Harmony game would be **STREET FIGHTER**.

**Threat** (which maps to **Neuroticism**) is the negative tone of the game that can evoke negative emotions in the player, such as addiction, anxiety, anger, or sadness. As I mentioned, Threat is the domain that has so far resisted my efforts to find games that I can predict players will like, so I will save further discussion for when solid data exists for this domain.



The MADDEN series is a good example of a game with a realistic, not fantastic, setting.



**MAPPING FACETS TO GAME DESIGN**

» Now we're ready to correlate the individual facets in the Big 5 factors to a game's capability to satisfy different types of player preferences. Remember; above we're measuring the player, but here we are measuring the game. The idea is that by correlating play preferences to game elements, we can predict what kind of player will enjoy, play, and buy games with those elements.

Let's start with a simple example: just one facet of the **Openness to Experience** factor, which we've described in game terms as **Novelty**. In the Big 5, the facet of "Imagination" reflects a person's preference for their inner, imaginative world over the "real world." I find (so far) that a player's Imagination score often directly maps to their interest in fantastic/imaginative settings (such as SKYRIM or MASS EFFECT) over realistic ones (such as CALL OF DUTY or MADDEN). So, I call this facet of a game "World," and describe it as the game's "offer of fantastic or realistic settings."

**World** is the first of six facets in **Novelty**. The other five are **Predictability** (offer of exploration and discovery mechanics over repetitive or "base-building" game mechanics), **Melodrama** (offer of emotionally evocative narratives), **Artistry** (offer of compelling visuals/audio), **Puzzle** (offer of puzzle-solving play), and **Message** (offer of socially progressive themes).

The **Conscientiousness** factor is described in game terms as **Challenge**. Our game facets are **Difficulty** (offer of difficult-to-accomplish goals), **Achievement** (offer of accomplishment recognition, such as Achievements), **Order** (offer of set-completion mechanics, as well as grid-based play over free-board-play), **Obligation** (offer of guilds and other social obligation structures), **Work** (offer of labor-intensive tasks

or "grinding"), and **Cautiousness** (offer of—precise, calculated play over run-and-gun—said another way, the silenced pistol over the rocket launcher).

The third Big 5 domain is **Extraversion**, which for games we map to **Stimulation**. The game facets are **Expression** (offer of positive socialization opportunities—chat, emotes, and so on), **Crowd** (offer of play with large groups of people), **Role** (offer of leadership roles versus follower roles), **Pace** (offer of a high volume of activities), **Thrill** (offer of high-intensity/exciting action), and **Joy** (offer of strong positive emotions in the player—happiness and delight, for example).

**Agreeableness** is the fourth domain of the FFM, mapped to **Harmony** for a game. The game facets are **Trust** (offer of play that includes or does not include the capacity to be betrayed, especially in a way that feels "outside the rules"), **Integrity** (offer of, or the lack of, the ability to do the above to other players), **Help** (offer of support roles), **Cooperativeness** (offer of direct confrontation with other players—note that while pure PvP is a low Cooperativeness score, team-based PvP is a high one), **Glorify** (offer of publicly viewable medals, scores, character customizations, etc.), and **Compassion** (offer of contexts that trigger and/or require an emotional comprehension of characters).

**Neuroticism** is the last domain, and while I call this domain of play **Threat**, the correlations are less clear. Currently, the facets I have are these: **Tension** (the capacity to instill fear in the player), **Provocation** (the capacity to make a player angry), **Despair** (the presence of "hopeless" in-game contexts), **Humiliation** (the capacity to make the player feel self-conscious), **Compulsion** (the presence of "addictive" game mechanics), and **Danger** (the capacity to hurt the player's

feelings). The issue here is that so far none of these facets have proven to be predictive—I have examples of players with very high and very low Anxiety scores (which maps to Tension, above) who both list RESIDENT EVIL 4 among their favorite games of all time. For now, Threat is still in "preproduction." Now for some homework. If you have read this far, you're clearly interested. Point your browser to <http://www.personal.psu.edu/ffj5j/IPIP/> and take the 300-question version of the Big 5 test. Then, with the mappings above, use those results to deconstruct what your motivations of play might be.

A note of caution: In giving the interviews, I have learned to strongly emphasize that we are trying to map preferences of play to game elements that are satisfying—not which elements players like. The idea of "liking" a game element includes the player's opinion on a lot of nongame things (how much time in the day they have for play, indie vs. triple-A, and so on)—and those are things that models like the Big 5 cannot predict.

**HOW YOU CAN USE THIS**

» The domains of play are a map to conclusions about how satisfying our games are, what motivations our games are not satisfying, what kind's of players are enjoying our games, and what kind's of players could be enjoying our games if we were to make specific changes.

Imagine that for each motivation facet, every game has a "band" of that facet that it "offers." For example, SKYRIM will offer a high-to-medium Fantasy facet, which means that players with a high, average-to-high, or average score in Imagination will find SKYRIM satisfying. Players with a low Imagination score will find the setting too exotic for their tastes (specifically, the existence and use of magic, in my experience). With this approach, we can map out how much "coverage" our games have for all of the motivations in the Big 5.

During game development, I am often confronted with this statement: "Players want (x)," where (x) is the speaker's opinion on what everyone wants. These are usually inaccurate statements, and we often intuitively know this. But how do you respond?

Now, I can answer in this way: "True! Half of them do. The other half want (y)," where (y) is the motivation on the other side of whatever facet the speaker has referred to. That has changed my approach to many of my team's design efforts. If the Big 5 system gave me only that, it would be worth it to me. But from what I can tell, that's just the beginning. ☺

JASON VANDENBERGHE is a creative director at Ubisoft Montreal, which is a pretty good gig, actually. Comments, concerns, criticisms, and offers to help out can be aimed at [jason.vandenberghe@ubisoft.com](mailto:jason.vandenberghe@ubisoft.com)

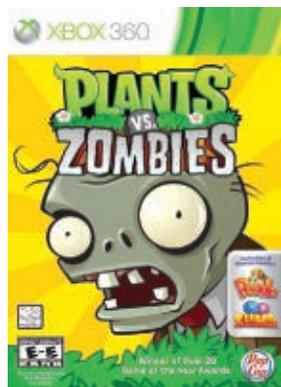
# GDC VAULT DEBUTS CLASSIC VIDEOS, GDC 2012 LECTURES

The GDC Vault service ([www.gdcvault.com](http://www.gdcvault.com)) has released both free and members-only video, audio, and slides from the Game Developers Conference in San Francisco, including free postmortem videos for games like *FALLOUT* and *HARVEST MOON*.

Following the conclusion of the record-breaking 22,500-person conference, these Classic Postmortem sessions, along with many other notable talks, are now available from GDC 2012's "Free Content" section ([www.gdcvault.com/free/gdc-12](http://www.gdcvault.com/free/gdc-12)) on GDC Vault.

Now in their second year, the Classic Postmortem lectures stood out as a particular highlight from this year's show, offering unique insight from some of the industry's seminal game creators.

The series includes notable talks from Frederick Raynal on the making of *ALONE IN THE DARK*, Ed Logg on crafting the classic arcade game *GAUNTLET*, Yasuhiro Wada on the quirky and successful *HARVEST MOON*, and Tim Cain on the original *FALLOUT*, which spawned one of the industry's most popular RPG franchises of all time.



Also available for free is an intimate chat with *MINECRAFT* creator Markus "Notch" Persson, hosted by *SPYPARTY* developer Chris Hecker. This session delves into Persson's creative process and provides a look into the mind of the indie game juggernaut.

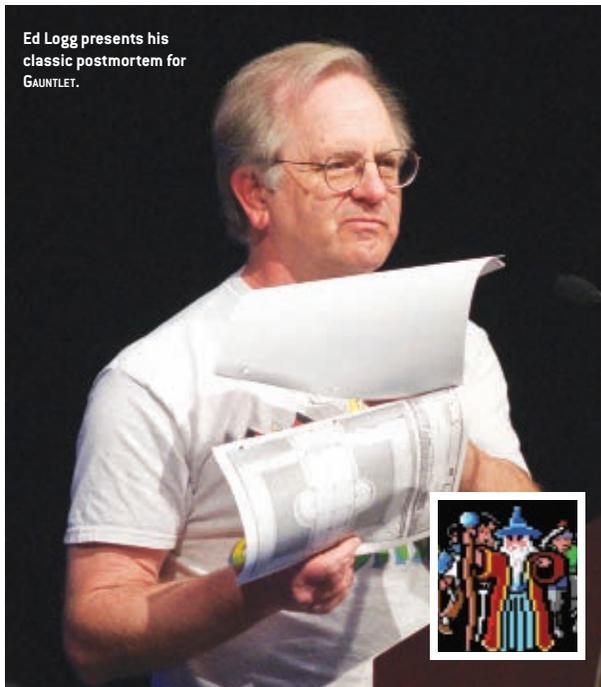
In addition, GDC Vault has debuted a panel featuring Persson alongside industry figures Jordan Mechner, Tim Sweeney, Adam Saltsman, John Romero, and Jane Pinckard, on the budding indie renaissance. Other feature-content on the GDC Vault includes a lecture from *PLANTS VS. ZOMBIES* creator George Fan on the best ways to teach players via game design.

Other Main Conference talks include a look at the art of *DIABLO III*, an audio session on Supergiant Games' indie hit *BASTION*, and production tips from Bungie's Brian Sharp. These lectures represent just a few of the free videos currently available on GDC Vault.

Outside these track highlights, GDC Vault also offers 25 free company-sponsored lectures from Intel, Nvidia, Google, and more. These free videos include high-quality panels on HTML5, Android, cloth simulation, and more, and are available after registering for a GDC Vault account. GDC organizers have also released hundreds of slide collections from the GDC 2012 presenters, offering a glimpse into many more of the show's most influential talks.

This new free content debuts alongside more than 300 additional lecture videos from GDC 2012 for GDC Vault subscribers. GDC 2012 All Access pass holders have full

Ed Logg presents his classic postmortem for *GAUNTLET*.



access to GDC Vault, and current subscribers with access issues can contact GDC Vault admins.

To purchase a standalone GDC Vault subscription, send an email to [help@gdcvault.com](mailto:help@gdcvault.com). Group subscriptions are also available; game-related schools and development studios who sign up for GDC Vault Studio Subscriptions can receive access for their entire office or company. More information on this option is available via an online demonstration, and interested parties can contact Gillian Crowley at [gcrowley@techweb.com](mailto:gcrowley@techweb.com).

Now that the GDC 2012 content is online, be sure

to keep an eye on GDC Vault for even more content in the months ahead. Show organizers will also archive videos, audio, and slides from upcoming events like GDC Europe, GDC Online, and GDC China, so there's plenty more content to come. To stay abreast of all the latest updates to GDC Vault, be sure to check out the news feed on the official GDC website ([www.gdconf.com](http://www.gdconf.com)), or subscribe to updates via Twitter (@Official\_GDC), Facebook ([www.facebook.com/gamedevelopersconference](http://www.facebook.com/gamedevelopersconference)), or RSS ([feeds.feedburner.com/gdcnews](http://feeds.feedburner.com/gdcnews)). #GDC



# TECH TAO FOR TINY TEAMS

HOW TECHNICAL ARTISTS CAN FIT INTO LOW-BUDGET DEV TEAMS

To me, the rise of technical art as an independent discipline has been one of the most interesting stories of the last decade in the video game business. The last console generation, born in titanic competition over graphics and production values, inflated many triple-A teams (and their budgets) to ungainly proportions. As the numbers grew bigger, developers learned to value those mysterious artists who specialized in making other artists more productive. The hobbyist riggers and hard-to-pigeonhole scripters of a decade ago have evolved into high-tech specialists in everything from shader programming, to SQL databases, to shuffling data over the web. At any given GDC you can find tech artists (more prolific than their pure-art brethren) arguing about the subtleties of Python syntax, the right way to manage a shader pipeline, or the hassles of getting data from a mocap studio on the other side of the Pacific. Tech art has gone big time.



PHOTO COURTESY OF GAME DEVELOPERS CONFERENCE

production work?" the answer is obviously "Hell, yeah."

## GIT 'ER DONE

» Tech art is a multifaceted (read: "schizophrenic") discipline. It melds technical concerns with art, and mixes spit-and-bailing-wire inventiveness with sober, serious process engineering. Tech artists are characterized by self-reliance, a bent for problem solving, and a grudging refusal to be stymied by technology—all things you need to ship a game without the money, time, and manpower of a megastudio.

Tech art is about getting things done with the tools at hand, which is important in an environment with limited time and scarce resources. Events that would be minor bureaucratic speed bumps for a 300-person team—say, a buggy 3ds Max hotfix that corrupts important files, a random publisher decree that all the green monsters have to become brown, or a last-minute push to crank out microtransaction-friendly tchotchkes—can be deadly for a smaller project on a budget.

In lean times, the left half of the tech-art brain—the one that comes up with the crazy, quick-and-dirty fixes at the pit of a crisis—is priceless. When a small team discovers that every texture in the game has to be converted from DXT1 to DXT5 in time for next week's demo, they don't have a pool of interns or contractors who can open and resave five thousand Photoshop files. The entire art staff is going to waste precious polish

Nowadays, the big-studio ecosystem in which tech art evolved has taken a few significant knocks. The rise of casual, social, and mobile gaming has put some brakes on team bloat. Sure, there are still 300-person megastudios toiling away on megaprojects, but a lot of the growth (and new investment) in the industry is coming from smaller groups who can take more creative risks and gamble on new platforms or

audiences. The crowd at any GDC lecture or IGDA meetup today is less grizzled, more diverse and often less lavishly funded than in years past. Which raises the question: What place is there for a tech artist on a tiny team? Is the tech-art field only a luxury for the heavy-iron houses, like rolling your own engine?

The answer to the question depends a lot on exactly how you ask it. If you're trying to figure

out whether a five-person art team needs a dedicated Motion Builder integration whiz, the answer is probably no. It takes a large organization to make productive use of people with deep but very narrow skills. On the other hand, if you phrase the question as, "We can afford only five artists. Shouldn't our art team have somebody who can put together some lightweight tools and do automation in addition to

time staring at file dialogs—unless, of course, one of those artists is the kind of person who can string together a command line file converter, a .bat file, and a copy of AutoHotKey into a temporary file conversion pipeline.

## CLEAN AND SOBER

» Of course, this kind of Rube Goldberg technology doesn't sit too well with the other side of the tech-art brain—the part that believes in building and maintaining solid, well-engineered tools. Scripting starts off as an intoxicating rush of new power, but as Spider-Man has taught us, “with great power comes great responsibility.” Far too many technical artists get their education in computer science by hobbling an entire studio with an innocent mistake born of eagerness and ambition. As they have become integral to the functioning of big teams, they've received a collective crash course in responsible software engineering.

This sense of responsibility is why so many GDC talks for technical artists focus on things like software patterns, error handling, and test-driven development. Improvisation is great for emergencies, but it's not how you run things day in and day out. In a big team, there's an infinite amount of work for the right-brained TA who dreads downtime and disorder. Every system can be made more bulletproof. Every U.I. can be streamlined. Even the biggest teams seem unable to keep their wikis and tutorials up-to-date.

Small teams tend to look on the industrial-strength side of tech art as an unaffordable luxury. With quick turnarounds and creative chaos, it can be difficult to budget time for under-the-hood improvements. Nonetheless, small teams also need to practice safe technical art. Even tiny teams need working tools to ship a game; indeed, they are even more threatened by downtime than the big kids. When you're shipping in six months instead of four years, busted tools hurt a lot more.

## PRACTICING SAFE TECH (ART)

» In practical terms, this means that small-team tech art needs to be tightly focused and minimalist, with few bells and whistles. Small teams can rarely afford a slick custom GUI, elaborate procedural rigging systems, or supersophisticated engine integration. On the other hand, small teams should not skimp on the basics: A fast, reliable export pipeline, no-nonsense tools distribution, and efficient asset management are even more important for tiny teams than they are for the behemoths.

The single most important skill for a small-team tech artist—more important even than Maya savvy or Max-Fu—is the ability to distinguish between the tools that ship games and the kind that serve mostly to wow your friends on Tech-Artists.org. Though your technical conscience may cry out for a cool, object-oriented and data-driven menuing system, you should settle for ugly old-school mel scripts if they can do the job. You may loathe P4Python and dream of replacing it with a cleaner API, but you'll have more important things to do than satisfy your need for order. Huge triple-A franchises can afford that kind of thing, but a \$750,000 project cannot.

This is not to say that small team TAs can ignore infrastructure—quite the contrary. Chaos is the natural order for small projects, and the only way to stay sane when things get crazy is to work from solid foundations. For example: An efficient, low-maintenance way of getting tools into your users' hands pays big dividends when crunch time looms. If you have to push a new fix out quickly, or get a bunch of contractors set up for a sudden content push, you can't rely on email attachments or sneaker-net. Any time you save by skipping on the distribution system will be paid many times over in debugging problems that come from botched installations, version mismatches, and oversights by busy artists.

Small teams also need to provide a high level of reliability, which in practice means some kind of automated testing and validation

for new scripts. Monster teams can afford to have a real human being show up at 6:30 every morning to run through the tools and make sure everything is working. The poor dev's alternative is some form of automated testing. Developers in other chaotic, fast turn-around businesses (for example, web development) make heavy use of automatic software testing to guarantee uptime, but this is still a novelty in the tech art world. It can be hard to convince your teammates that writing tests instead of tools is a good use of precious time. That said, a system that can vet every check-in to make sure it won't bring down the studio due to a missing file or a syntax error is an essential survival tool for the small team TA. (If the phrase “test driven development” doesn't

for the curious, DIY type. And for the tech artists who don't want to lose touch with their roots in actual artistry, small teams also offer (or more likely, demand) content creation as well as tools and troubleshooting.

Of course, the jack-of-all-trades lifestyle doesn't appeal to everyone. Some tech artists prefer the master-of-one route, which suits them far better for the kind of big projects that need them to find a specialization and refine it. The choice between the extremes is eminently personal, but it's important for every TA to have some idea where they are most comfortable on the spectrum from generalist to specialist.

So, does tech art have a place on smaller-scale projects? The answer is an unqualified yes!



PHOTO COURTESY OF GAME DEVELOPERS CONFERENCE

ring any bells for you, check out a TDD website like <http://onlamp.com/pub/a/python/2004/12/02/tdd.pyunit.html>, *The Foundations of Agile Python Development* by Jeff Younker, or *Head First Software Development* by Dan Pilone and Russ Miles. You'll be glad you did.)

The most critical element of small-team tech art is flexibility. This is true for tech, but even more true of people. Casual, social, and mobile games attract a lot of veteran developers precisely because small teams need generalists. If you hate being pigeonholed, small teams offer the opportunity to spread your wings. Tiny projects demand the kind of artists who can do effects on Monday, shaders on Tuesday, and update 300 buggy collision volumes on Wednesday—perfect

As young companies and new genres push the boundaries of the industry, they can benefit from both the technical agility and the hard-won production wisdom of technical artists. Many TAs, meanwhile, will find a great arena for their skills and all-rounder habits on the wide-open frontiers. (It isn't often we get to end these columns without a hedged bet or qualification somewhere. Enjoy it while it lasts!) 🎮

---

**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*.



# MOVING TO MOBILE

MATT SUGHRUE LEAVES SEVEN45 STUDIOS FOR METAVERSAL STUDIOS

These days, it seems like everyone has their eyes on in mobile game development. Matt Sughrue tells us about his transition from Seven45 Studios (POWER GIG: RISE OF THE SIXSTRING) to Metaversal Studios's (UPDRAFT JACK) relatively tiny team.

**PATRICK MILLER:** *What made you decide to leave console development for iOS?*

**MATT SUGHRUE:** The project my team was working on (a band game) hit the streets at the exact time when the music game genre was crashing and burning. That was enough for the owners of the company to want to move on to other kinds of consumer products. The team was dismantled and I found myself on the job market for the first time in a very long



time. After that experience, I wasn't keen on getting back into large-scale development again. I had some breathing room thanks to a decent severance package, so I decided to take my time and find a gig that had the potential to rekindle my passion for making games.

I'd been an avid iPhone gamer since the App Store first opened, and the idea of making smaller games for that platform was very appealing. I got in touch with a recruiter colleague of mine and found the opportunity at Metaversal Studios not long after that. Metaversal at that time was a ten-person team made up mostly of current and former college students from Northeastern University. There wasn't a lot of formal "game company" structure like source control or schedules—the team just decided what game they would make and made it. They had released a number of games this way, but none of them had been a commercial success. I came on as VP of product development and brought some of the best practices tools, like consistent documentation, formal bug reporting, and milestone deliverables, that I had been using over the years to the table. The end result is that the raw talent and creativity of the

team has been channeled into a development process that is manageable and delivers great games on time.

**PM:** *Did you just wake up one morning and say to yourself, "Man, I really want to go make some mobile games!"*

**MS:** Not really. I've been making games for the past twenty years and the progression of my career had always been to take on more challenges at bigger companies. In 2010 I was running a console studio with seventy developers that I had built from the ground up. Mobile games seemed like a backward step at that point.

**PM:** *What's it like working with a small team?*

**MS:** It's a big adjustment to go from many departments who are building out chunks of a single game over the course of two years that will be sold through retailers, to individuals who are working and collaborating closely over three or four months to make a game that's sold digitally. What I love most about this market is that it's still possible for a small team to make original games that sell well. That's very difficult on the console side, where the cost of developing and marketing your own games is much higher and the developer is up against publishers with big brands and big marketing budgets. While I do miss the challenges of pulling a big console team and title together, I love the collaborative aspects of working with a small team to make a game and the challenge of getting that game up the charts. We've been fortunate to have some success with our titles, and that's given us the ability to continue to make the games we want to make. That's rare in this industry, and I'm grateful for that opportunity every day.

**PM:** *Do you find that the hyper-competitive App Store is significantly affecting the way you make games?*

**MS:** Competition makes us work harder, get more creative, and make our games better. We are driven by our own desire to make great games and by the people who play them. Our players tell us without restraint what they think of our work in the App Store, through tweets, and on our Facebook page, and we pay attention and do our best to improve the experience with every new release. 

## who went where

SONIC THE HEDGEHOG co-creator Hirokazu Yasuhara has left Namco Bandai for an undisclosed role with Nintendo Software Technology, a North American Nintendo subsidiary in Redmond, WA. Yasuhara got his start at Sega in 1988, serving as director, game planner, and designer for many of the original SONIC THE HEDGEHOG titles. He has also worked on Naughty Dog's JAK & DAXTER and UNCHARTED franchises, and more recently, Namco Bandai's PAC-MAN PARTY.

Thatgamecompany saw several recent departures after releasing JOURNEY. Executive producer Robin Hunicke has left to join Katamari Damacy creator Keita Takahashi at Tiny Speck; WAY designer Chris Bell, "feel engineer" John Nesky, and WAY programmer Walt Destler have left to remake WAY as a new studio called The Wilderness; and co-founder and president Kellee Santiago has left simply to "seek new challenges."

WORMS franchise creator Andy Davidson has returned to Team 17 fourteen years after he originally left. Davidson came back to Team 17 to work on the upcoming WORMS REVOLUTION, due out for Windows PCs and consoles later this year.

Infinity Ward's CALL OF DUTY creative strategist Robert Bowling has stepped down from his position and left Activision entirely. As of this writing, Bowling is working on a yet-undisclosed project.

## new studios

Gabriel Knight creator Jane Jensen and composer Robert Holmes have founded a new studio dedicated to adventure games called Pinkerton Road Studio. Pinkerton Road Studio aims to focus on storylines that are "deep and compelling," and has launched a Kickstarter project to fund its first year of development. Kickstarter backers will be able to influence game development decisions and participate in beta tests.

Gael Giraudeau and Nicolas Godement-Berline have started a new Paris-based mobile studio named Majaka, which has just launched its debut iOS title SKI CHAMPION. Giraudeau previously worked for Arkane Studios, and Godement-Berline worked on several PlayStation 3 titles for outsourcing house Virtuos. Majaka plans to release three to five games by the end of 2012.



# THE MAGIC OF FREE-TO-PLAY

## HOW I STOPPED WORRYING AND LEARNED TO LOVE THE FREE-TO-PLAY BUSINESS MODEL

The first successful free-to-play (F2P) games—defined as “games whose primary revenue source are in-game purchases”—hit the market over a decade ago. Now they’re everywhere. They account for eight of the top ten grossing games on iOS as I write this. Rumor has it that all the major consoles will support F2P games in the next generation. Even our industry’s most prominent, respected developers (PopCap and Valve, for example) have begun to embrace the model.

Let me tell you what F2P represents to me: an opportunity to bring entertainment to billions of people without relying on advertising revenue or government subsidies. An opportunity to embrace players who want to play our games but can’t (or won’t) pay, instead of forcing them to become pirates. An opportunity to stop making disposable entertainment experiences and instead create games that live forever, supported by devoted fans who happily spend money to keep their favorite hobby alive.

For the first time in the history of mass media, we can entertain huge audiences without first bombarding them with advertisements for sugar water and corn flakes and without making them pirates. It’s a beautiful thing. I’m not personally opposed to advertising in games, but I find it puzzling that so many developers accept advertising, which is essentially a form of psychological manipulation, while decrying in-app payments.

### ANY GOOD TOOL CAN BE USED FOR EVIL

» Yes, you can build F2P games that resemble slot machines and are designed to prey on people with addictive personalities. This is also

true of card games (i.e., Blackjack), but you don’t hear people protesting against all card games (i.e., *Dominion* or Solitaire). So please, stop confusing the bad things you could do via F2P with everything that can be done via F2P!

Here’s a challenge for every curmudgeon that hates F2P games: Start thinking about them as a form of progressive taxation, and allow your mind to expand from there. That’s right: F2P is a system that subsidizes the poor with payments from the relatively wealthy.

Think it can’t be done? Check out *TRIPLE TOWN* and *REALM OF THE MAD GOD*. Both heavily favor skilled play over “purchased” advantages; unskilled, wealthy players absolutely cannot purchase their way above skilled players on the leaderboard. Neither contains systems that encourage insane levels of spending, though large monthly expenditures are possible. Those large monthly expenditures are nothing beyond the level of what an enthusiast might spend on a favorite real-world hobby such as remote-control cars, golf, or gardening.

### ROTMG AS PROGRESSIVE TAXATION

» *REALM OF THE MAD GOD* generates revenue primarily via the sale of “character slots,” which allow you to play more than one character at a time, and “vaults,” which allow your characters to squirrel away more loot. Neither of these things are required to play the game, and both can essentially be acquired for free by creating additional free accounts, though that’s obviously not as convenient. A large additional source of revenue comes from the sale of “keys,” which are instant portals to dungeons that must otherwise be sought out in the game. Again, buying keys isn’t a precondition to playing the game or even gaining access to dungeons;

they are simply a convenience.

What’s particularly interesting about the dungeon keys in *REALM OF THE MAD GOD* is that they are, in many ways, the purest incarnation of the idea of F2P as a progressive tax or social good. Players want to plunder dungeons because they contain good loot. But buying a key just gets you a chance to earn that

Whether it’s DRM in PC games or unnecessary “online only” features in console games intended to deter their resale, developers are constantly struggling with business challenges imposed by consumer desire for a cheaper (or free) product. There’s also the common player desire for online games to live forever, even when



loot; you still need skill to actually earn it. And because the most lucrative dungeons are also the most deadly, wealthy players who buy keys have an explicit incentive to invite along other players, lest they die alone and lootless in their own private dungeon.

### ROSE-TINTED GLASSES

» It always amuses me when people pine for the “good old days” of game development, when designers weren’t concerned with base financial considerations. The arcade games that many of us grew up playing were explicitly and painstakingly designed to munch quarters every few minutes! But many of us still fell in love with *PAC-MAN*, *DONKEY KONG*, and *STREET FIGHTER*, and were inspired by those games to become the developers we are today.

Even modern games on consoles and PCs have seen their business model change.

those games require servers and other expensive infrastructure. So why not embrace those desires?

### SIGNING OFF

» I’m not suggesting that F2P is for everyone. There are many amazing games that would be difficult and perhaps impossible to make as F2P games. So yes, if you love those games, keep making them. Just understand why the rest of us have chosen a different path. We’ve chosen the opportunity to entertain millions of people, for free, often without any forced advertising or government support, for years and years to come. It’s an amazing thing, when you stop to really think about it. ☺

*DAVID EDERY is the CEO of Spry Fox and has worked on games such as REALM OF THE MAD GOD, STEAMBOYS, and TRIPLE TOWN. Prior to founding Spry Fox, David was the worldwide games portfolio manager for Xbox Live Arcade.*



# VROOM VROOM

## A STUDY OF SOUND IN RACING GAMES



Sometimes, the best way to learn about a game's sound design is just to listen. While reading about technology and techniques exposes us to their existence, I believe that as much can be learned about the technical and aesthetic aspects of game audio by starting from the player's perspective. Recently, I joined forces with audio enthusiast and motorhead David Nichols (Track Time Audio) to dissect the sounds of racing games. We picked apart twelve different games to expose the current generation's state-of-the-art in vehicle simulation sound just by listening deeply on the player side of the magic screen. Here are a few highlights from our study.

### START YOUR ENGINES

» Sound has a unique ability to communicate directly to the player in a way that is impossible to represent visually—all the power and fury of an engine revving, for example, can be transferred directly from the speakers to the player's brain in a pure expression and feeling of speed. Racing games lose a feeling of immersion without sound. The sound of the engine is positioned front and center, and it gives the

The earliest racing game engine sounds used the simulated vehicle RPM parameter to drive the pitch of a square wave. Increasing the RPM increased the pitch of the engine note, which communicated to the player their current speed.

Racing game sound has made tremendous leaps, even within the current generation. For example, the quality of a vehicle's engine sound design is currently measured by how naturally the RPM progression is handled during acceleration

well. When the player's tire loosens its grip, the player will hear a screeching mayhem of kinetic friction and know that their car is no longer driving at optimum speed. This sound element has been around since the beginning of racing games.

The current generation of racing games takes the sound of tires and brings an additional level of communication to help players better understand their connection to the road. From the driver's perspective, the sounds a tire makes as it approaches its grip limit helps to identify how soon they will exceed the force of static friction and lose traction. The approach of this limit is reflected in a howling or hum present in the sound of each tire. The current crop of racing games represents this sound using multiple loops of sound that are transitioned between and pitched using a parameter from the tire simulation for each individual tire.

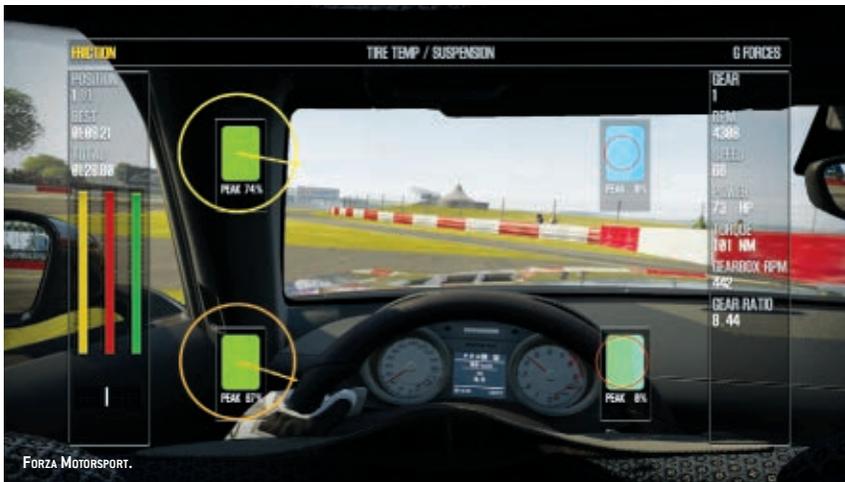
The telemetry view in Turn 10's *FORZA MOTORSPORT* series allows you to see the dynamic compression of a tire and helps to frame the change in sound.

### ROLL OUT

» A good vehicle sound design needs to be heavily integrated into the vehicle and tire simulations—and we hope that these informal sound studies can help you tease out exactly how the sound designers did it. The remainder of the racing game sound study covers the use of sound for surface materials, camera perspectives, physical damage, NPC vehicles, and user-interface design—and, of course, the subtle and extreme differences in technique between each game and audio team. You can find the rest of our study at Track Time Audio ([www.tracktimeaudio.com](http://www.tracktimeaudio.com)) and Lost Chocolate Lab ([www.lostchocolatelab.com](http://www.lostchocolatelab.com)), and Game Audio Podcast #18 ([www.gameaudiopodcast.com](http://www.gameaudiopodcast.com)) as well.

*"Let the good times roll"—The Cars*

**DAMIAN KASTBAUER** is a technical sound design rat fink who can be found hot-rodding game audio at [LostChocolateLab.com](http://LostChocolateLab.com) and on Twitter @LostLab.



player a constant indication of speed in a way that the speedometer can't.

The engine is also there to remind you that "the car is the star," as Mike Caviezel recently said in the racing game-focused Game Audio Podcast 18. The three main sound components of the engine are the exhaust system, air induction, and the internal mechanics of the engine, which can be implemented using a loop-based system, granular synthesis, physical modeling, or a hybrid of the three. Once the basic sounds for each component are in place, each sound is then tied to the same vehicle simulation used for every other aspect of the game using parameters to drive the change and modification of content.

and deceleration; an engine noise can quickly sound unrealistic when it is taken from its natural recorded state and modified parametrically using DSP without the proper technical design. It's easy to pick out the early racing titles from the current console life cycle that use a loop-based engine sound because the engine sounds are dramatically pitched, which makes them sound synthetic.

### GETTING TIRED

» "Where the rubber meets the road" is a common idiom used to describe the point where things start to get serious. It's the same thing for a racing game: The sound of the tires serves to make the player feel "at one with the road"—if it's done



WWW.MATTIATRAVERSO.COM/GAMES

# ONE AND ONE STORY

PLENTY OF NEW GAME DEVELOPERS CHOOSE TO START OUT WITH A SIMPLE 2D PLATFORMER, BUT FEW WOULD THINK TO MAKE A 2D PLATFORMER BASED ON A ROMANTIC GAME MECHANIC LIKE IGF STUDENT SHOWCASE FINALIST ONE AND ONE STORY. WE TALKED WITH ONE AND ONE STORY DESIGNER AND PROGRAMMER MATTIA TRAVERSO ABOUT THE DEVELOPMENT PROCESS AND THE TRANSITION TO IOS.



**Patrick Miller:** Tell me about the team behind *ONE AND ONE STORY*. Who worked on it, and how did you all meet?

**Mattia Traverso:** I made the core of the game, (with very rough graphics) in about a month, and then I started looking for artists and musicians. David was easy to find because he was very well known in the Flash game scene, but before finding Gabriele I wasted two months with various artists around the web who couldn't get the mood of the game in their drawings and mock-ups.

I realized a few days ago—maybe seven months after the development of the game—that I have never talked directly with them! We used Skype to text, but we've never heard each other's voices. Strange, huh?

**PM:** How did you come up with the design for *ONE AND ONE STORY*? Did you draw upon any life experiences or memories?

**MT:** I think the best way to describe the game is that I had a relationship, but it was full of crates and spikes, so we had to break up!

Joking apart, the design phase worked in a similar way to Jenova Chen's design process, but I'm not saying I'm that good, of course! His design philosophy is different from the common one: Normally, you take a mechanic and make sure it is fun. But Jenova does not start with a mechanic; he starts from an emotion, from a feeling that he wants to transmit to the player.

And that's how we worked on *ONE AND ONE STORY*, exploiting the game mechanics to work toward the narration and therefore the emotions of the player.

**PM:** Did you decide to do *ONE AND ONE STORY* in Flash from the beginning, or did you start with something else?

**MT:** Sadly, I was forced to use

Flash because I'm just starting my developer career and that's the only tech I know. Since then, I have come to know a lot of programmers. But back then, nobody would have listened to a boy with such a simple idea. We are now remaking the game for iOS devices thanks to a very good C++ programmer, Tommaso Checchi.

**PM:** Recently, you found an unauthorized copy of your game on the iOS App Store. How did you respond to that? Were you able to find anyone who could help you deal with the legal matters and get the copy pulled off the store?

**MT:** When I discovered the copy, I shouted some not-so-polite words on Twitter. Surprisingly, that helped a lot: I was retweeted by around 100 indie developers, and my problem got spread around the web very quickly. I got a lot of contacts willing to help me, to name a few: Adam

Atomic (CANABALT creator) sent me the DMCA module to take the app down, IGDA and the Vlambeer guys (SUPER CRATE BOX) contacted Apple. I got a fantastic wave of support from the community. A few lawyers also contacted me even though I never intended to sue the "thieves." After two weeks or so, the app was removed from the store, though I haven't heard anything else from Apple!

**PM:** Now you're working on your own iOS port. Is it hard to make *ONE AND ONE STORY* work well on a phone and tablet?

**MT:** Yes! Since it's a platformer, the controls are a major issue. I personally don't like games with a lot of buttons on the screen, because that's just a way to adapt a game that was not originally designed for iOS. I'm trying to rethink the control scheme to work without a single button on-screen. The system is up: Now we just need to playtest it!

Another problem is that the game is too short to be in a paid form: We need to add content, but without ruining the atmosphere and the mood, which is tough.

Besides that, I should thank Apple: The art now needs to be redone twice as large for the new iPad! 🍏

**RELEASE DATE:** 10/10/2011

**LENGTH OF DEVELOPMENT:** 3 months. 90% of the game was finished in a month, but we spent two months polishing the final 10%.

**BUDGET:** One half-eaten lollipop. That's it.

**LINES OF CODE:** 4,000+

**FUN FACT:** An early build of the game had an ugly brown background, and one playtester said "Those five minutes were the worst of my life, I was just wandering around a brown nothing," which prompted Gabriele to make the great art we used in the final build. Guess we should thank that tester today.

# 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](http://fullsail.edu/outcomes-and-statistics).



800.226.7625

[fullsail.edu](http://fullsail.edu)

# BUILD THE FUTURE

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.

All courses are taught with a curriculum designed and delivered by prominent game industry professionals. Students are immersed in a comprehensive educational game experience within a creative community in courses like:

- Analog Game Theory
- The Business of Games
- Game Art
- Game Audio
- Level Design
- Project Management
- Game Programming
- Game Production

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

THE  
**LOS ANGELES**  
FILM SCHOOL  
ANIMATION • AUDIO • FILM • GAMES



**Call or Click Today!**  
800-406-7485 • DesignLAFilm.com

FOR MORE INFORMATION ON OUR PROGRAMS AND FEES VISIT [WWW.LAFILM.SCHOOL](http://WWW.LAFILM.SCHOOL) ©2012 The Los Angeles Film School. All rights reserved. The name "The Los Angeles Film School" and "The Los Angeles Film School" are service marks or registered service marks of The Los Angeles Film School. Accredited by ACCSC.

## ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
E3 EXPO .....	15	MAGIC PIXEL GAMES.....	3
EPIC GAMES .....	6	PERFORCE SOFTWARE.....	C2
FULL SAIL REAL WORLD EDUCATION .....	54	RAD GAME TOOLS .....	C4
HAVOK .....	C3	SIGGRAPH .....	31
LOS ANGELES FILM SCHOOL.....	55	VANCOUVER FILM SCHOOL .....	23

*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.



# HOT RUMORS!

## SECRET INSIDE INFO HERE

Hey. You've probably heard of me. Yeah—I'm pretty much the guy with all the super top-secret rumors and mind-blowing insider stories about the game business. That's right, I've got my finger on the hidden pulse of what's really happening underneath those superficial press releases, official statements, and stooge journalists. I'm really well connected, and I know everyone that matters. My vast network of highly placed sources at the upper echelons of every single game company in the world ensures that I'm updated with top-secret information 24 hours a day. I get all the best game industry rumors even before the guy at GameStop does.

Don't believe me? I'll give you some scuttlebutt right now—on the down low, of course. Promise what I tell you stays between you and me. Okay?

### THE NEW CONSOLES

» No doubt you've heard about the next generation of consoles under development at Microsoft, Sony, and the secret Google/Samsung/Mattel partnership (you did know about that, didn't you?). And you've probably already heard that the next Xbox is codenamed "the 2012 Hyundai Azera," which multiple sources have already confirmed will get better gas mileage and feature more passenger room than any other four-door vehicle in its class. But that's all the amateur level stuff—I knew about all that literally five years ago. Have you heard about Sony, though? Do you know about the new technologies that their crack team's been cooking up in their secret undersea laboratory?

Well, I'll tell you, if you keep it to yourself: I have it on really good

authority that the PlayStation 4 will include a chip with a radical new design that will blow everything else—the PS3's Cell included—completely out of the water. The details are sketchy at this point, but the tantalizing hints I was able to coax from my sources suggest that the new silicon die will include as many as 27 "Empathic Processing Units," which will vibrate in harmony with the feelings of the central processor. That will make it by far the most powerful video game hardware conceived by humankind! Don't tell anyone I said that!

### BIG UPCOMING GAMES

» That reminds me. I was hanging out in a bar in Redmond, Washington when some members of the Wii U development team came in and sat right next to me. Naturally, my ears were on super-alert for interesting new tidbits. The Nintendo guys were muttering something about "margaritas could be stronger" and "pass me those tortilla chips"—both of which sound like code names for new Nintendo projects. New METROID, anyone?

Speaking of games in development, did you know that all kinds of stuff gets worked on and then cancelled? Totally true story: A few years back there was an actual, playable HALO game for the Nintendo DS! I'm not kidding! It was called "HALODS." Guess who nixed that one... wrong! It was nixed by Sony, actually, during a session of the Console High Council—a regularly held symposium at a highly secret location where the reigning lords of Nintendo, Sony, and



Microsoft gather and discuss matters of mutual importance (I've heard it's near a volcano, but come on—that just sounds crazy and made up!). The Sony PlayStation Grand Priest, resplendent in his traditional dark purple and black robes, argued that HALODS might "upset the balance" of the industry if it were to come to pass, and instead he suggested that each of them spend a lot of money developing a console-based MMO that they would then cancel several years into the future. They all agreed.

### THE WINDS OF CHANGE

» But the word on the street is that the era of stability brought about by the all-powerful High Council is drawing to a close. The winds of change will soon blow across the industry again, upending established order. For just one example, I had the chance to catch up with a friend of mine who just recently caught up with an acquaintance of Yu Suzuki. "Oh? And what is Suzuki-san up to these days?" I asked (the "-san" indicating that I'm all hip and with it vis-à-vis Japanese culture!). "Oh, not a whole lot," he said. "I heard he's currently in negotiations

with Valve to fund and publish a new three-game SHENMUE series as an exclusive for the upcoming Steambox console." Bam. Megaton. You heard it here first, folks.

There are other signs that the landscape is changing. A very well-known and well-connected game industry analyst who spoke to me on the condition of anonymity—let's just call him Michael, uh, Bachtter—mentioned an interesting fact that underscores the shift. Rovio, the maker of ANGRY BIRDS, is in talks to buy Sega. How exciting! Imagine what they'd do with those franchises! And that's nothing to say of the rumor that a whole mess of Zynga executives were spotted looking really happy and relaxed in the Konami headquarters office recently. Oh, and let's not forget my good buddy was at a secret invite-only party at GDC earlier this year, and personally witnessed Notch make Yoichi Wada an all-cash tender offer to buy Square Enix—right there on the spot!

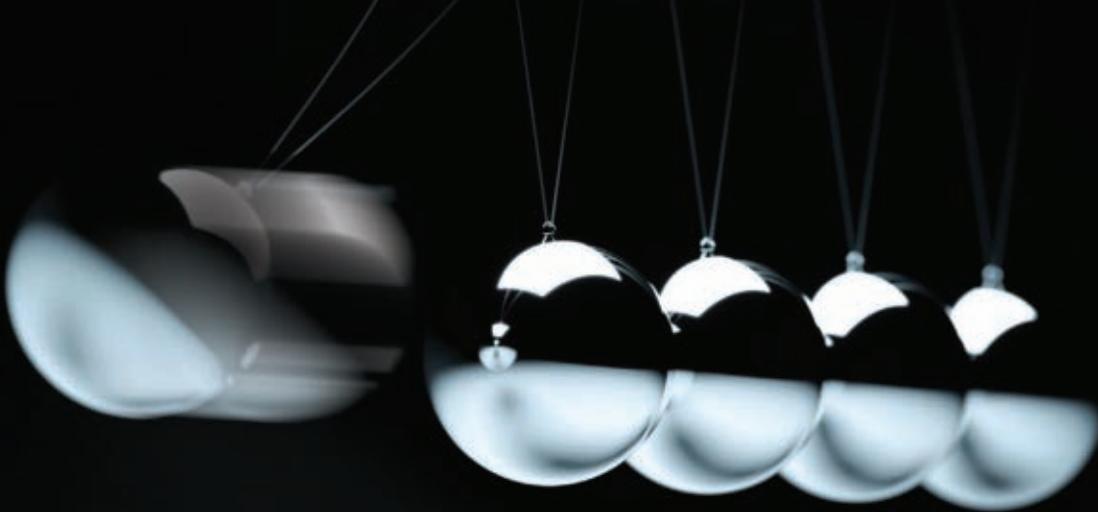
### ULTIMATE RUMOR MASTER

» So what's the theme that emerges from all this talk? Well, I think it's obvious: For inside scoops, nobody's got me beat! Keep reading this column and paying attention to me for more amazing secret information the big boys don't want you to know! 🗨️

**MATTHEW WASTELAND** writes about games and game development at his blog, *Magical Wasteland* ([www.magicalwasteland.com](http://www.magicalwasteland.com)). email him at [mwasteland@gdmag.com](mailto:mwasteland@gdmag.com).



# See where we're taking Physics next...



**havok™ Physics**

*Technology for the Future*

Havok has always been at the forefront of innovation.

We have been building an arsenal of tools and technology for where the industry is heading next, pushing every element of Havok to new heights.

**Contact Havok sales today to learn more about  
the future of Havok technology.**

[www.havok.com/sales](http://www.havok.com/sales)



*Thank you to all our customers and fans who voted for Havok Physics for the 2011 Game Developer Front Line Awards!*

**Havok™ Technologies Include:**

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

THE EASIEST WAY TO MAKE

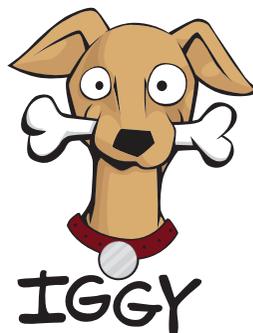
user interface



# SIZZLE

IS WITH

I LIKE BACON



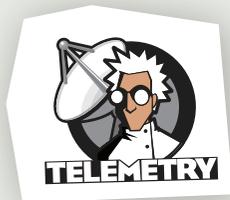
# Iggy

Our powerful new system for cooking up UI using content created with Adobe Flash®

SUPPORTS ACTIONSCRIPT 3 and

INCLUDES A HIGHLY OPTIMIZED GPU RENDERER

IT'S ALSO INTEGRATED WITH



SO KICK IT UP A NOTCH WITH

# Iggy

IT'S NOT JUST DELICIOUS...

# IT'S rad!



[www.radgametools.com](http://www.radgametools.com)  
(425) 893-4300