

THE LEADING GAME INDUSTRY  
MAGAZINE VOL 19 NO 11  
NOVEMBER 2012 INSIDE:  
GAME DEVELOPER POWER 50



# THE BINDING OF ISAAC

POSTMORTEM  
**ARENA KINGS**  
MIDMORTEM

GAME DEVELOPER MAGAZINE





## Spark Unlimited Explores Lost Planet 3 with Unreal Engine 3

Capcom has enlisted Los Angeles developer Spark Unlimited to continue the adventures in the world of E.D.N. III. *Lost Planet 3* is a prequel to the original game, offering fans of the franchise a very different experience in the harsh, icy conditions of the unforgiving planet. The game combines on-foot third-person perspective action with first-person mech combat against an array of monstrous Akrids.

For the first time in the best-selling franchise, the game is being developed with Unreal Engine 3 (UE3), which suits Spark well thanks to a long history of working with UE3 technology.

"The previous two current-gen titles developed here used UE3, so the decision to use it for *Lost Planet 3* was natural," said Matt Sophos, *Lost Planet 3* game director, Spark Unlimited. "It allows us to leverage our past experience and optimized workflow, thus increasing our efficiency and quality. UE3's robust toolset also makes it the best choice for level design because it enables us to hit the ground running from day one – laying out gameplay spaces and prototyping encounters."

"The tools are extremely powerful for artists and designers who are creating content for a game," said Sophos. "UE3 has a fast iteration pipeline and the ease of use allows artists and designers to be directors so they can work on shaping the scene and the experience rather than simply creating assets that go into a game. Another major benefit to using Unreal is the amount of documentation and training videos made available by Epic. This can dramatically cut down our training time for new employees."

Although there are many tools to work with, Spark used Unreal Kismet and Un-

real Matinee extensively for *Lost Planet 3*. Sophos said these tools empower level designers, artist, animators and sound designers to quickly prototype, iterate and polish gameplay scenarios and cinematics. With multiple departments being comfortable with Kismet and Matinee, engineers and designers are no longer the bottleneck when it comes to implementing assets, which facilitates rapid development and leads to a higher level of polish across the entire game.

Sophos said the communication between Spark and Epic has been great in its ongoing relationship, plus his studio has been able to utilize the Unreal Developer Network (UDN) for any issues throughout development.

"UDN offers a great community knowledge base that we can tap into any time a question arises for how to tackle tough problems," said Sophos. "This has helped answer many questions that would normally have taken weeks of research, allowing us to iterate more quickly than if we had to recreate the process from scratch."

Spark has added proprietary technology on top of UE3, including dynamic storm states that punctuate the volatile nature of the hostile planet of E.D.N. III. The storm states allow for environmental storytelling, as well as giving the player new visuals that show the damage and effects of extreme weather conditions on the planet and its inhabitants.

"Most of our additions on top of UE3 are gameplay systems to support *Lost Planet 3*, such as a more robust third-person camera system, animation choreography, a collision system for larger creatures, a multi-threaded AI formation system, a quest system, and so on," he said.

One of the many things that stands out in this new adventure is the cinematic look and feel of the game. The world of E.D.N. III comes to life in a new way, thanks to the game's setting that takes place early on in the human habitation of the distant planet. Sophos said the art direction of *Lost Planet 3* has drawn

many inspirations from visionary directors such as Ridley Scott and John Carpenter. Using UE3's volumetric lighting capabilities of the engine, Spark was able to more effectively create the moody atmosphere and lighting schemes to help create a sci-fi world that shows as nicely as the reference it draws upon.

"Even though it takes place in the future, we definitely took a lot of inspiration from the Old West frontier," said Sophos. "We also wanted a lived-in, retro-vibe, so high-tech hardware took a backseat to improvised weapons and real-world firearms. The Utility Rig feels more like a trucker's big rig than a mech. Surprisingly, there was only so much inspiration we could take from the arctic. Snow environments can easily devolve into looking like Antarctica, so we were careful to accentuate the alien nature of the landscape to constantly remind players E.D.N. III is not Earth."

In addition to a very deep single-player experience, aims to deliver a campaign that's massive in scope yet fueled by an intimate story, down-to-earth characters, and the small personal touches that drives a player's desire to see what happens next, Spark is pushing the multiplayer experience. Sophos said the team has chosen to implement a robust suite of gameplay modes that is separate from the single-player narrative campaign in order to best serve the needs of both experiences. Gamers will be able to explore the extreme conditions of *Lost Planet 3* for Xbox 360, PS3 and PC in 2013.

Thanks to Spark Unlimited for speaking with freelance reporter John Gaudiosi for this feature.

### UPCOMING EPIC ATTENDED EVENTS

D.I.C.E Summit  
Las Vegas, NV  
February 2013

Cloud Gaming Europe  
London, UK  
February 21-22, 2013

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



# GAME DEVELOPER M

## post mortem

### 26 THE BINDING OF ISAAC

One year ago, SUPER MEAT BOY dev Edmund McMillen rocked the indie-game world with THE BINDING OF ISAAC, a highly controversial—and at times even disturbing—dungeon-crawling roguelike that captivated audiences with its grotesque, evocative imagery and design. In this month's postmortem, McMillen tells us how THE BINDING OF ISAAC started life as a game jam project conceived from a deeply personal place—and how it went from an underdog to an international indie hit.

*By Edmund McMillen*

## features

### 6 POWER 50

People power the game industry. In this year's installment of the Power 50, *Game Developer* and Gamasutra editors team up to acknowledge the individuals in and around the game industry whose efforts in art, audio, design, business, evangelism, and programming have inspired us to step up our collective games.

*By Patrick Miller, Christian Nutt, and Thomas Curtis*

### 14 SHOOT MANY ROBOTS: ARENA KINGS MIDMORTEM

How does a game developer "pivot"? Demiurge Studios's cofounder takes us through a special "midmortem" that explains how work on XBLA 2D shooter SHOOT MANY ROBOTS set up the studio to make a free-to-play multiplayer spinoff called SHOOT MANY ROBOTS: ARENA KINGS. Find out what has gone right and wrong (so far) with the process of taking work on one game and turning into a completely new game altogether.

*By Albert Reed*

### 23 HOW LOUD SHOULD IT BE?

Loudness metering standards can open up new avenues for game audio designers to be creative (and efficient)—but only if they use the right standards. Audio specialist Shaun Farley explains why the game audio community needs to look outside loudness metering standards designed for broadcast media or risk losing their oomph.

*By Shaun Farley*

## departments

- 2 **GAMEPLAN** *By Patrick Miller* [EDITORIAL]  
Help Us Help You
- 4 **HEADS UP DISPLAY** *By Lauren Manary* [NEWS]  
TECMO SUPER BOWL 2013 and including players with disabilities
- 34 **TOOLBOX** *By Tobias Heussner* [REVIEW]  
Articy:Draft Review
- 37 **INNER PRODUCT** *By Niklas Frykholm* [PROGRAMMING]  
10 Tips for Cleaning Bad Code
- 42 **PIXEL PUSHER** *By Steve Theodore* [ART]  
Seen Through Goggles
- 44 **DESIGN OF THE TIMES** *By Soren Johnson* [DESIGN]  
How to Become a Game Designer
- 47 **AURAL FIX** *By Damian Kastbauer* [SOUND]  
Dynamic Animation Sound Now
- 49 **THE BUSINESS** *By Sana Choudary* [BUSINESS]  
Tell Story, Sell Game
- 50 **INSERT CREDIT** *By Brandon Sheffield* [EDITORIAL]  
Video Games in Retrograde
- 52 **GDC News** *By Staff* [NEWS]  
GDC Next, App Developers Conference, and GDC China 2012 talks
- 53 **GOOD JOB** *By Alexandra Hall* [CAREER]  
Q&A with Derek Manning, new studios, and who went where
- 54 **EDUCATED PLAY** *By Alexandra Hall* [EDUCATION]  
OF LIGHT & SHADOW
- 56 **ARRESTED DEVELOPMENT** *By M. Wasteland & M. Underland* [HUMOR]  
The Real Hollywood Mojo



gd

GAME DEVELOPER MAGAZINE

CONTENTS.1112  
VOLUME 19 NUMBER 11



# HELP US HELP YOU

## ARE YOU A BAD ENOUGH DEV TO HELP GAME DEVELOPER?

Hello, world! Starting with this issue, I'll officially be taking over the Game Plan editorial (and the rest of the magazine) while editor emeritus Brandon Sheffield jumps headfirst into the deep end of game development. (You can still find him in the mag in his new column called Insert Credit, where he'll continue to opine on all things dev-related.)

### GDMAG: THE GAME

» Brandon usually used this editorial page for hard-hitting and insightful opinions and commentary on issues and trends in the game industry—which is a tradition I hope to continue—but for this issue I'm going to do something different. I'm going to describe a bit of the process of making a magazine in game development terms, and I'm hoping you'll be able to help us out with it so that we can help you make better games by giving you a better magazine.

At first glance, the process of making a magazine doesn't really look much like the process of making a game—but you'd be surprised how much they can have in common. Games have artists, producers, designers, and testers; we have a small staff that covers the bases (you can see who does what on the masthead to the right of this column). Games often have miserable crunch times for weeks or months before a major development milestone; magazines often have a miserable week every month while we scramble to get everything edited, laid out, and approved in time for our monthly ship date. We may have more in common than you might think!

### MAGAZINES AS A PLATFORM

» As a publishing platform for *Game Developer*, magazines are reliable and consistent enough, although we miss out on a few major features compared to web and mobile-focused publishing platforms. With magazines, we don't have to worry about maxing out your bandwidth or killing your battery life, and we've got our production process down pat. (If you are reading this on our digital

edition: Thanks! We're still trying to make it better, and we should have some new announcements on that front soon.)

On the other hand, magazines also give us a few major problems that game devs can sympathize with. First off, we have discoverability issues within the magazine—it's hard to surface each issue's excellent lineup. Outside of the GDMag loyalists who read every issue from cover to cover, we know you guys read the front page, the back page, and the postmortem, but it gets fuzzy after that. Now, we know that most GDMag readers have to wear a lot of developer hats—even if you're primarily a coder, you may need to know a bit about other dev disciplines too—but it's hard to surface all that stuff with only a table of contents and a few small cover lines. You can't really share articles with your friends online, either, so we can't really count on our magazine stuff going viral.

Related to the discoverability problem is an **analytics** problem—it's not easy to tell what you guys are reading! Print magazines don't really support JavaScript, so we can't track page views or social networking shares or anything like that. And considering our options for reader engagement are rather limited—paper doesn't have any ways to embed apps or plug in with other APIs—the readership statistics we *do* get aren't easily correlated to what we did in any given issue of the magazine in the first place.

Finally, we have something of a **content pipeline** problem—we have to outsource most of our content development. After all, pretty much everyone who is good enough at making games to merit publishing in *Game Developer* is, well, making games, not working

as a full-time magazine freelancer. We rely on a wonderful network of expert developers to help us fill our pages every month, but once you've written one or two articles about a game you've been working on for the last two years, we can't really ask you to write a third, so we have to wait for you to make a new game and write about that. As a result, we're constantly on the hunt for more people to tell us about the problems they've solved and the things they've built.

### HOW YOU CAN HELP

» It would be pretty cool if we had some kind of electronic, Internet-connected paper replacement that we could use to solve all our platform problems (—sent from my iPad). Fact is, however, we're still doing a print magazine because we love it, and from what we can tell, you love it too. So instead of trying to fix our platform problems with newfangled tech, we really just want you to help us make a magazine. We're not asking for much; there's one simple way you can help.

**Talk to us.** You can contact the edit staff through our Editorial Feedback Contact Form on our website (<http://gdmag.com/contactus>), you can "Like" the magazine's page on Facebook (search for Game Developer Magazine), and you can message us on Twitter (@GameDevMag). Tell us what you like and what you don't like. If something doesn't look right, tell us. If you have a question for us or one of our authors, ask us. If you are working on something you want your peers to know about, tell us about it.

Let's make a better magazine so we can make better games.

—Patrick Miller  
@pattheflip

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

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

Lauren Manary, Alexandra Hall, Albert Reed, Shaun Farley, Edmund McMillen, Tobias Heussner, Steve Theodore, Soren Johnson, Damian Kastbauer, Sana N. Choudhury, Brandon Sheffield, Magnus Uderland

#### ADVISORY BOARD

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

### ADVERTISING SALES

#### VICE PRESIDENT, SALES

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: [gina.gross@ubm.com](mailto:gina.gross@ubm.com)  
t: 415.947.6241

#### GLOBAL ACCOUNT MANAGER, EDUCATION

Rafael Vallin e: [rafael.vallin@ubm.com](mailto:rafael.vallin@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)



UBM

WWW.UBM.COM

the box office grew by **13%** in the **Arab World\*** but dropped by **4.9%** across the world.



get your ticket to the fastest growing film market.

**twofour54° Abu Dhabi** – the tax-free gateway to one of the world's fastest growing film and production opportunities.

Box office sales are growing at an incredible rate and **twofour54° Abu Dhabi** offers the film business a prime opportunity. With state-of-the-art production and post-production facilities, access to first-class creative professionals and a rebate of up to 30% on qualifying spend for all productions made in Abu Dhabi. These are just some of the reasons companies like the Digital Domain Media Group are setting up here. With over 100 film and entertainment companies already established at **twofour54°**, isn't it time you capitalised on the opportunity?

- 100% company ownership in a stable, tax-free environment, competitive & flexible rates
- Guidance & liaison with UAE content regulatory bodies, including film permits
- Easy licensing & business set-up services, unique campus environment & facilitated networking
- On-site media training academy for production professionals
- Diverse range of locations, experienced scouts & crew, freelance talent pool, OB units, trucks & fly packs
- State-of-the-art production complex & fully equipped TV studios
- Fully integrated HD digital & post-production facilities, 3D stereoscopic labs, award-winning Baselight colour grading system, VFX
- Broadcast & distribution facilities, satellite uplink, playout services, host broadcasting, file ingest

Find out how we could help grow your business today.

**twofour54.com/film**  
**+9712 401 2454**



**twofour54**  
Abu Dhabi

media & entertainment hub

\* Sources: United States Census Bureau. Age and Sex Composition 2010. Experian Marketing Services. 2010 American Movie-Goer Consumer Report 2010. Arab Media Outlook 2010. Media on the Move 2009. A. T. Kearney. Youth in the Arab World. Media Habits of MENA Youth 2010. Jad Melki.



# level playing field

IS YOUR GAME ACCESSIBLE TO PLAYERS WITH DISABILITIES?

Nonprofit organization Ablegamers recently published the most comprehensive text on game development for people with disabilities: "Includification," a 48-page guide that can assist developers with making their games available to the widest audience possible. *Game Developer* spoke with Steve Spohn, editor-in-chief of Ablegamers.com, about why game devs should be working to make their games more accessible.



**Lauren Manary: What kinds of accessibility issues do you look for in games?**

**Steve Spohn:** Take, for example, a player with color[-perception] deficiency who wants to play a new puzzle game. With no means of differentiating colors, he becomes frustrated and returns the game. If the game had included an accessibility option for the colorblind that allowed different shapes for the puzzle pieces, markers on the colors, or some way other than color to differentiate puzzle pieces, the player would have been able to enjoy his purchase.

**LM: What's the business case for making games accessible to people with disabilities?**

**SS:** Two business reasons. First, there are over 33 million gamers with disabilities in America alone. With these kinds of numbers, developers are potentially leaving a large amount of money on the table by not including some easily implemented features we outline

in the "Includification" document. Second, as the average age of the game enthusiast market continues to increase, players with disabilities will continue to be an ever-increasing percentage of the market. Studies from studios like PopCap have shown an overwhelming part of casual gamers consider themselves to have some sort of disability.

**LM: Do you see "Includification" making waves in the game dev world?**

**SS:** Waves? No. Actually, at PAX-E 2012, we gave STAR WARS: THE OLD REPUBLIC our game of the year award for most accessible mainstream game in 2011. The lead developer on the panel accepted the award, but refuted the idea that the game was made with accessibility in mind; he said that they simply created a game with as many features they thought the audience would want to see as possible. After the panel in an off-the-cuff conversation, we spoke about his comments, and I told him good game design includes

accessibility. He agreed good game design is all about keeping your options open, and adding features people want to see.

**LM: How have your guidelines been received by the development community so far?**

**SS:** "Includification" was actually launched due to demand from developers. We have been advocating for more than eight years on the importance of accessibility, but around two years ago, the questions from developers changed from "Why should we include accessibility?" to "How do we include accessibility?" At that point, we knew we needed to create a document that laid out as many problems the disabled game community faces as possible, as many solutions as possible, and the cost-effective/bottom-line argument for doing so.

Alex Rigopulos of Harmonix was the first studio head to inform us of the need of such a document. He wanted something he could give to

his lead developers and say, "This is important; we should implement what we can from these." And so we did. Since then, we have had contributions and encouragements from EA, BioWare, Rockstar, PopCap, and a huge welcome from independent studios such as Minicore and Uber Entertainment.

**LM: Given that alternative input devices (touchscreens and motion controls, for example) aren't going away, what can developers do to make games that use such controls more accessible?**

**SS:** Motion-controlled games are not going away, but they'll always be a niche market up until the point where you are talking holograms and the Holodeck. They are popular in their own right, but a Wii or Kinect just doesn't compare to the gaming experience of a console or PC. However, the accessibility of those devices depends on the game themselves.

FRUIT NINJA is a game that can be controlled from a wheelchair. The iPad has hundreds of games for the autistic community. So, not all motion-control and touchscreen games are bad. They're just inaccessible to a certain segment of the disabled gaming community.

**LM: Are some genres more inaccessible than others?**

**SS:** Generally speaking, FPS and driving games are the most inaccessible due to the large amount of buttons that need to be pressed often and simultaneously, and the twitch reflexes these genres demand. MMOs and RTSes tend to be the most accessible because they're a little bit slower paced, and often include difficulty modifiers that allow players to go at their own speed. —Lauren Manary

# move over, madden

HOW A DEDICATED MODDING TEAM BROUGHT TECMO SUPER BOWL INTO THE MODERN ERA

Why line up for a midnight MADDEN release when you can stick to NES classic TECMO SUPER BOWL? The intrepid TecmoBowl.org team has been regularly updating TECMO SUPER BOWL with new features and updated teams and player rosters since 2007, so we thought we'd ask lead developer Matt Knobbe about the process of (and problems with) modding 8-bit games.

**Lauren Manary: Why Tecmo Bowl?**

**Matt Knobbe:** Actually, when people say TECMO BOWL, they usually mean TECMO SUPER BOWL; while Tecmo managed to burn the name of their company into the collective consciousness of gamers from the '90s, they made it really hard to distinguish between their various incarnations of the game. TECMO BOWL was a four-person arcade football game, and its success paved the way for the fairly popular NES game TECMO BOWL featuring licensed NFL players. Later, toward the end of the NES era, Tecmo combined all its programming knowledge from the two previous games—along with being the first company to secure the NFL team and NFLPA rights—to create the masterpiece known as TECMO SUPER BOWL.

I first started playing TECMO SUPER BOWL with my college roommate, and it soon progressed into a floorwide thing. Throughout the years, TECMO SUPER BOWL has been a social exercise of sorts, and eventually grew into a hobby, as I've always been into

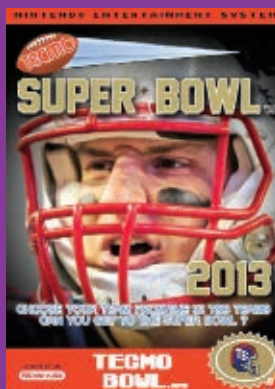
computers and programming. I started up a forum about modifying the game after finding some like-minded people on the Internet, and we've been doing this ever since.

**LM: Was modding TECMO SUPER BOWL different from working with other NES games?**

**MK:** It mostly came down to deadlines. We're essentially shooting for a release on a certain day and with the fluidity of a pro football roster which would rarely be a constraint of any other non-sport NES game. Otherwise, the majority of skills that might be leveraged on TECMO SUPER BOWL apply across the board to a majority of NES games. Since we're a sports-centric community, we have hacks based on RBI BASEBALL, BASEBALL STARS, and the catalog of games Tecmo put out like TECMO NBA.

**LM: How'd you keep the file size down?**

**MK:** When we're modding TECMO SUPER BOWL, we're essentially talking about deconstructing NES code written in the 6502



programming language, then emulating it. Thus while you're no longer really constrained by the physical limitations of the memory of the cart, you're still limited by the original design of the NES itself. The original TSB was 384KB, and many of the early advanced hacks out there were the exact same size. Over the years, as we've increased in knowledge and added the occasional superstar hacker to our team, we've been able to remap certain areas of the game with some versions topping out at a whopping 1MB! The 2013 version we released is at 512KB, which includes extra teams and some new menu options.

**LM: How'd you build your team?**

**MK:** Sometimes you get lucky and find someone who excels at low-level hacking and modern programming languages, but usually it's about connecting the people who do one with the people who do the other. One of the first programs ever written for this was by the guys at emuware.com—they essentially started the Tecmo modification movement. They originally made it possible for nearly anyone to

modify the original names and attributes of TSB without having to manually change pointers or learn hex. This caused the interest in modding TSB to surge, and soon people were asking, "Well, what else can we change?" Once you start programmatically eliminating modifications that are essentially trivial or well defined, you free up those who are capable of digging further into the game.

**LM: Were you concerned about legality issues?**

**MK:** Originally, we were quite afraid of Nintendo or Tecmo sending a cease-and-desist letter. Over the years, Tecmo sort of let us know it wasn't an issue, and Nintendo stopped caring about games for the original NES system. I suppose that the NFL could decide to send out a C&D, but I doubt it's worth their time. Now I'm not completely terrified to be candid about the issue—it would seem that being essentially nonprofit and perhaps a bit off-the-radar is a pretty good way to not get sued.

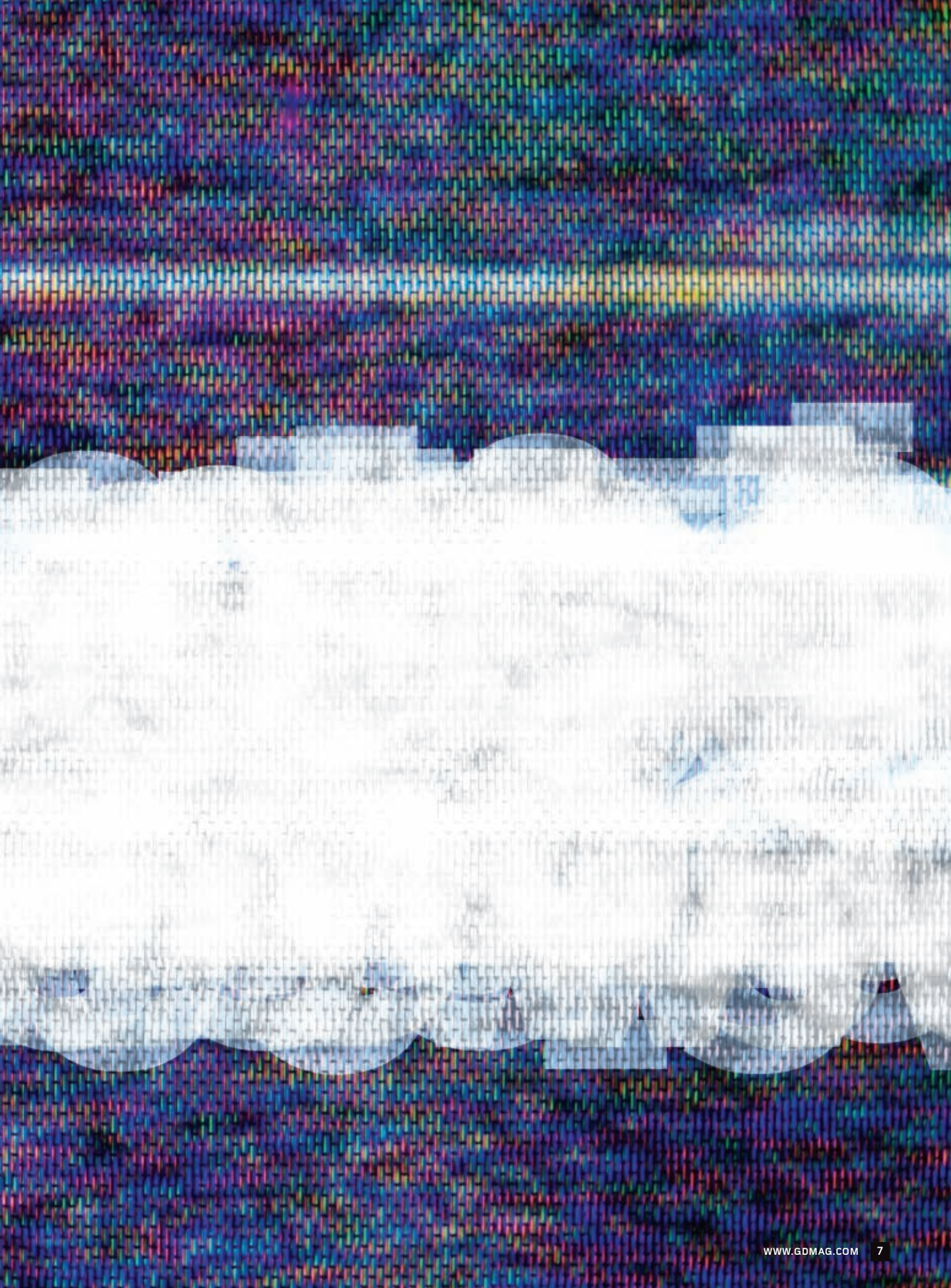
**LM: What kind of bugs did you find in the original release?**

**MK:** Over the years, we found that certain attributes weren't actually implemented by Tecmo—maybe they ran out of time, or forgot about them. The player attributes for passing accuracy on offense and quickness on defense were remapped as the values for other functions in the game. Probably the biggest bug in the game we found was when Player 2's attributes were displayed incorrectly when conditions change. —Lauren Manary









# power 50

## { ART }



**VINCENT Perea**  
Disney Mobile

❖ Design aside, the Disney-published *WHERE'S MY WATER?* really stands out because of its visual style. Illustrator/designer Vincent Perea uses a simple, personality-packed cartoon aesthetic with a fairly minimalist approach. Perea's fluid animations and adorable character designs manage to inject the experience with a ton of charm—which is particularly impressive considering the screen is filled with little more than dirt and concrete most of the time.



**SHELDON Carter**  
Digital Extremes

Hand-painted, cel-shaded visuals—check. Ruthless heavy-metal soundtrack—check. Sheldon Carter's creative direction in *THE DARKNESS II* is, well, dark, and that's exactly what a game about mob bosses with supernatural powers should be. Kudos to Carter and his team for building *THE DARKNESS II*'s unified and seamless aesthetic vision.



**ROB Nelson**  
Rockstar Games

❖ If you look at any recent Rockstar title, it's clear the studio has a real affinity for film. Its game cinematics often borrow editing and compositional techniques from movies and other visual media, and under creative director Robert Nelson's direction in *MAX PAYNE 3*, the studio experimented with some new visual techniques from outside the traditional GTA wheelhouse.

*MAX PAYNE 3* tells the story of a haggard ex-cop with a crippling addiction to both booze and painkillers, and Rockstar's cinematics go a long way toward reinforcing the character's tormented mental state. The camera often flickers out

of focus, key lines of dialogue linger onscreen to emphasize Max's guilt, and clever split-screen action scenes make poignant moments seem more frantic and distressed. These visual techniques are subtle on their own, but when combined they really elevate the game's depressing and action-fueled narrative.



**MARIEL Cartwright**  
Reverge Labs

❖ The fighting game genre may be alive and well these days, but the impeccably fluid, detailed 2D animation we loved about the genre in the 1990s is practically nowhere to be seen—except in Reverge Labs's *SKULLGIRLS*, anyway. Thanks to lead animator Mariel Cartwright, *SKULLGIRLS* looks every bit the 2D fighter of our dreams, with hand-drawn characters and backgrounds that ooze personality and craftsmanship out of every frame. Step aside, *STREET FIGHTER III: THIRD STRIKE*; *SKULLGIRLS* has set a new standard.



**DANIEL Dociu**  
ArenaNet

*GUILD WARS 2* looks like art director Daniel Dociu took a bunch of beautifully detailed concept art and somehow plugged it directly into a game. That Dociu could lead ArenaNet's art team to do that much is impressive; that he managed to do that in an MMO, where visuals must often be sacrificed for performance's sake, is nothing short of spectacular.



**BRIAN Provinciano**  
VBlank Entertainment

❖ "Retro" is probably the most-overused term in video games these days. Everything with visible pixels, TEMPEST-style vectors, or low-resolution textures can be called "retro," and we're kind of tired of it. However, there is something to be said for getting retro right, and *RETRO CITY RAMPAGE* creator Brian

Provinciano knows this. It takes more than a crash course in pixel art to make a proper retro game; Provinciano's deep understanding of the NES hardware's capabilities and limitations were instrumental in making *RETRO CITY RAMPAGE* ring true to the old-school.



**DAN Miller**  
The Blast Furnace

❖ Now that we're done ranting about the overuse (and abuse) of "retro" aesthetics, we thought we'd call out a rather exceptional case. New Activision mobile dev studio The Blast Furnace made an intriguing *PITFALL!* iOS game that we would describe as "new retro," if that makes sense. Under Dan Miller's artistic direction, *PITFALL!* for iOS brings us into a world of cartoony-looking untextured polygons that look kind of like the original *PITFALL!* and *SUPER MARIO 64* had a video game baby. It's kind of how we imagine the original *PITFALL!* would look if we were trapped inside of it.



**THOMAS Shahan**  
Northway Games

Next time you have a hard time finding an artist for your game, try looking around on Wikipedia. That's how *INCREDIPEDE* dev Colin Northway found photographer/woodblock-print artist Thomas Shahan, anyway; Northway was reading the entry for "jumping spiders," saw one of Shahan's illustrations, and tracked him down from there. While we may not be tremendous fans of spiders (jumping or otherwise), we have to admit that we can't think of a better match than insect-specialist Shahan for a game that is "about life and feet."



**SIMON Flesser**  
Simogo

❖ In early 2012, Simogo's *BEAT SNEAK BANDIT* rocked our worlds (and snagged Best Mobile Game in the Independent Games Festival

2012 awards) with its cartoony, syncopated charm. The credit for said charm belongs to Simogo cofounder Simon Flesser, who was the art-and-sound whiz behind BSB's whimsy. In an interview with Gamasutra, Flesser described BSB as "Cool, but in a silly cartoon show kind of way." We want more of that.



**RANDY Smith**  
Tiger Style Games

❖ With *WAKING MARS*, Tiger Style Games founder Randy Smith led his dev team to make something atmospheric, curious, and gorgeous—and as we're looking at all the different ways mobile game devs try to make their work visually stand apart from the pack, we think that Smith and his team are on the right track.

In an interview with Gamasutra, Smith described *WAKING MARS*'s aesthetic as "a combination of fine details and abstract implication...not realistic, exactly, but believable." We weren't sure what to expect when you described your game as "action gardening," Smith, but you and your team ended up nailing it.

## { AUDIO }



**SHAW-HAN Liem**



**JONATHAN Mak**  
Queasy Games

❖ PlayStation 3 indie hit *SOUND SHAPES* marries music and platform-hopping together in a manner so elegant and intuitive you might not at first realize how many iterations it took to get right. Jonathan Mak and Shaw-Han Liem (also known as I Am Robot And Proud), from Toronto-based Queasy Games, are the two responsible for making *SOUND SHAPES* work.

Games built around music largely live or die by how well their designers can integrate music into

the core design. In *SOUND SHAPES*, songs are the levels, and with the level editor, we too can make and play our music. With *SOUND SHAPES*, Mak and Liem remind us that music games can be more than a series of notes that we plug into a bulky, plastic, guitar-shaped controller.



**DAVID Kanaga**  
N/A

✦ If *SOUND SHAPES* gently massaged our brains into a state of musical play, composer David Kanaga's work on *DYAD* simply melted said brains outright. Thanks to *DYAD*, we can check "David Kanaga and [*DYAD* creator] Shawn McGrath" off our list of fantasy indie game dream collaborations.



**RICH Vreeland**  
Polytron Corp.

✦ After years of anticipation, FEZ finally wowed indie game scenesters with its throwback look and feel. We would be remiss if we didn't include Rich "disasterpeace" Vreeland's FEZ soundtrack work in this year's Power 50 audio nominations. Thanks to Vreeland, FEZ feels atmospheric, pensive, maybe even a little bit melancholy.

## { BUSINESS }



**DAVID Hayes**  
Lionsgate

✦ Both the *Gamasutra* and *Game Developer* staff were in complete agreement about one of this year's Power 50 candidates for the business category, but we didn't know his name. We all just wanted to make sure we recognized "Whoever it was that got Adam Saltsman [*CANABALT*], Paul Veer [*SUPER CRATE BOX*], and Danny Baranowsky [*SUPER MEAT BOY*, *CANABALT*] to make a licensed game for iOS based on *The Hunger Games*."

Turns out that person is film studio Lionsgate's VP of digital marketing, David Hayes. In the world of film marketing, Hayes has made a name for himself by building innovative marketing campaigns; by getting an indie dream team to make *THE HUNGER GAMES: GIRL ON FIRE*, by getting that indie dream team to make more than just a *CANABALT* reskin, Hayes has left his mark on the game industry as well.



**GREG Rice**  
Double Fine Productions

✦ By now, everyone knows the story of the \$3.3 million *DOUBLE FINE ADVENTURE* Kickstarter, so we'll keep this one short. These days, game-related Kickstarter campaigns are flooding our in-boxes and news feeds, and we blame all of that on *DOUBLE FINE ADVENTURE* (tentative title) producer Greg Rice, who was that Kickstarter's mastermind. Next time you disrupt the traditional developer-publisher business model, give us a minute or two to prepare our spam filters first, okay?



**MARCIN Szymanski**  
Robot Entertainment

From the outside, designing free-to-play games looks kind of like juggling flaming swords. Your game needs to be profitable, but not exploitative; engaging, but not demanding. We don't know if *HERO ACADEMY* lead designer Marcin Szymanski can actually juggle flaming swords, but considering how well *HERO ACADEMY* attracted casual and core audiences and got them to open up their wallets without sending off pay-to-win vibes, we think he could probably do it if he really, really wanted to.



**BOBBY King**  
FarSight Studios

While game-related Kickstarters were a dime a dozen in 2012, FarSight Studios caught our eye with

*THE PINBALL ARCADE*, a cross-platform pinball game that featured licensed digital remakes of real-world classic pinball tables. FarSight Studios's VP of development Bobby King, along with President Jay Obernolte, were the ones responsible for getting the licenses to more than 20 tables from manufacturers Bally, Williams, Stern, and Gottlieb—which included running successful Kickstarter campaigns to get the necessary funding for licensing fees and production for the *Star Trek: The Next Generation* and *Twilight Zone* tables. We wouldn't wish that kind of red-tape wrangling on our worst enemy.



**ARAM Jabbari**  
Manager, PR/Sales, Atlus

✦ Sure, PR and marketing are hard at any publisher. But when you're a niche company like Atlus, which focuses almost exclusively on the decidedly untrendy genre of Japanese RPGs, you live or die by the thinnest margins. That's what makes Aram Jabbari's job so hard. Jabbari interfaces with the company's finicky community on games like the *PERSONA* series, making sure their voices are heard, and along with vice president Tim Pivnicny, makes sure the games are presented to their exacting standards.

Though the trend for Japanese niche games is 'supposedly downward' Atlus's most successful years have been its most recent—and its "Atlus Faithful" fans, carefully cultivated by Jabbari's always-on-message but sincere marketing efforts, are driving that success.



**DAVID Perry**  
Gaikai/Sony

✦ Two cloud-based streaming game companies entered 2012 with vastly different business plans; OnLive aimed its cloud game service directly at consumers, while Gaikai developed its tech and courted bigger hardware companies like Samsung and Sony. Gaikai, led by cofounder and veteran game developer David Perry, sold to Sony

for \$380 million, while OnLive laid off its entire staff and got bought and restructured by a venture capital group. Perry's work speaks for itself; it takes more than good tech to make it in this business.



**IBAI Amezttoy**  
Active Gaming Media

Active Gaming Media has its fingers in a lot of pies; CEO Ibai Amezttoy founded the company as a Japan-based game localization agency, but since then AGM has done work in public relations, QA, and other aspects of the industry. This year, Amezttoy and AGM get the Power 50 nod for Playism, a digital distribution platform that they're using to bring Japanese indies to overseas markets (the English-language Playism portal launched with a localized version of indie Japanese hit *LA MULANA*) and vice versa (localizing and publishing *DEAR ESTHER* and *SPACECHEM* in Japan). Way to spread the indie love (and make a buck doing it), Amezttoy.



**TORSTEN Reil**  
NaturalMotion Games

✦ If you first heard of NaturalMotion Games during Apple's iPhone 5 event, when CEO Torsten Reil demoted *CLUMSY NINJA*, then you haven't been paying attention. NaturalMotion Games seems to get exactly how to build a free-to-play mobile game that looks good, plays well in short bursts, and—if CSR *RACING*'S \$12 million take during its first month on the iOS App Store is any indication—get players to pay with well-designed monetization strategies. Watch out, world: Reil is one to watch for 2013.

# power 50



**RICCARDO  
Zacconi**  
King.com

❖ It wasn't that long ago that conventional wisdom dictated breaking into the top five Facebook developers was impossible, thanks to the unassailable value of cross-marketing to an existing audience. Tell that to King.com CEO Riccardo Zacconi, who, on the back of the success of *BUBBLE WITCH SAGA*—a title with 3.7 million daily active users in October 2012—now runs the number-two game developer on the social networking site.

Yes, King.com's premier game is, at its core, a refresh of Taito's 1990s arcade hit *BUST-A-MOVE*, but according to Playdom's Steve Meretzky, it's the expert application of social mechanics that made it a success, and that's King.com's secret (and explains why Taito didn't get there first). The company releases over a dozen games a year to its casual portal, and only the top performers are selected for Facebook. A carefully designed "social envelope," in King.com parlance, is wrapped around a game, and then it's ready for social network deployment.

The company has also moved into mobile versions of its titles, with clever integration between Facebook and iOS—meaning that players' progress carries over between versions. This led to a very successful launch of *BUBBLE WITCH SAGA* on that platform without a meaningful marketing effort. This is leading, in turn, to an expansion of King.com as a company—a big business win for an organization that saw an opening for traditional casual games on Facebook, realized it could provide them, carefully executed its plan, and took the platform by storm.

## { DESIGN }



**NELS  
Anderson**  
Klei Entertainment

❖ Klei's *MARK OF THE NINJA* feels like a distinct breath of fresh air

for the stealth game genre. Lead designer Nels Anderson opted not to follow in the footsteps of *METAL GEAR* or *SPLINTER CELL* and instead bring stealth to a 2D game by introducing a number of clever systems that eliminate the genre's rough edges.

In many stealth games, for instance, it can be hard to tell when enemies can detect you, but in *MARK OF THE NINJA*, all of these systems are clearly telegraphed via a number of visual cues. Enemy sight lines are represented as beams of light, loud noises project shockwaves into the environment, and characters lose saturation as they step into the shadows, so players know when they're at risk and how their actions will affect their enemies.

By making all of the mechanics so easy to read, the game becomes less about trial and error and more about using your wits to work make the most of the tools at your disposal. It makes the stealth genre far more enjoyable, and other games should make sure to take note.



**JENOVA  
Chen**  
thatgamecompany

❖ Thatgamecompany is well known for designing games to evoke specific emotions. With *JOURNEY*, thatgamecompany cofounder and creative director Jenova Chen managed to avoid the frustration, disappointment, and general misanthropy we normally feel while playing games online with strangers, and replace them with camaraderie, joy, and gratitude. Props.



**SEAN  
Vanaman**



**JAKE  
Rodkin**  
Telltale Games

Many games try—and fail—to elicit an emotional response from their players, but *THE WALKING DEAD* is one of the few series that gets things right. Co-lead designers Sean Vanaman and Jake Rodkin

have guided *THE WALKING DEAD* to emphasize smart writing over complex mechanics, and in doing so, they've created one of the most affecting interactive stories we've seen in quite some time.

Like the comic book it's based on, *THE WALKING DEAD* series doesn't spend all its time focusing on the horrors of the zombie apocalypse, and instead takes plenty of time to develop its characters and create a world that players can invest themselves in. While there's plenty of zombie fighting to go around, it's the game's quieter, more thoughtful moments that make its more horrifying scenes all the more poignant.

Perhaps most interestingly, the game forces players to make some extremely tough decisions that'll affect the characters they've grown to care about. Every decision comes with major consequences, and the game doesn't hesitate to twist the knife when things seem to be at their worst. Very few games manage to present choices that have an emotional impact on the player, and the fact that *THE WALKING DEAD* manages to do so over and over again is a tremendous accomplishment.



**TETSUYA  
Takahashi**



**KOH  
Kojima**  
Monolith Soft

❖ While *FINAL FANTASY* and other classic JRPG franchises have struggled to find their voice during this console generation, Monolith Soft lead designers Tetsuya Takahashi and Koh Kojima's work on *XENoblade* proves the genre has plenty of life left—not by revisiting the genre's glory days, but by making a number of important changes that help bring the genre into the modern era. *XENoblade* gives players a sandbox-style world to explore at their leisure, and even uses a number of classic MMO systems to help expedite combat, streamline quest systems, and offer more freedom to the player. In other

words, *XENoblade* celebrates (and updates) the best of the JRPG genre.



**SAMI  
Saarinen**  
RedLynx

❖ Why build a game when you can get your players to do it for you? RedLynx lead technical artist Sami Saarinen is responsible for building the *TRIALS* series' original 3D editor in 2008, advocating for its inclusion in *TRIALS HD* in 2009, and guiding it into its *TRIALS EVOLUTION* incarnation. But this isn't just a level editor, mind you; Saarinen's *TRIALS EVOLUTION* editor actually gives players access to the game's visual programming language, so they can make and share fiendishly difficult tracks—or even entirely new games altogether.



**HARVEY  
Smith**



**RAF  
Colantonio**  
Arkane Studios

❖ *DISHONORED* wowed the industry at E3 and Gamescom this year with its compelling steampunk setting and remarkable focus on player agency. In an industry with lots of games focused on violence and killing, co-lead designers Harvey Smith and Raf Colantonio's direction in *DISHONORED*'s design adds some genuine emotional weight behind the decision to kill, which is something we'd like to see more of.



**CORY  
Davis**

Yager Development  
(former)

Spark Unlimited (present)

❖ Yager Development/2K Games's *SPEC OPS: THE LINE* made waves this year for adding a liberal dose of *Apocalypse Now*-inspired narrative twist to an otherwise fairly standard cover-based shooter. Creative director Cory Davis's initial vision documents from 2008

describe SPEC OPS: THE LINE as an “intense third-person military shooter with a dark and mature narrative...[that] confronts you with the horrors of war, as you face choices between bad and worse in challenging moral dilemmas.”

Well, with SPEC OPS: THE LINE, Davis did just that. We were impressed by Davis’s creative vision in 2008, and we’re impressed by how faithful the end product was to that vision four years later. Love it or hate it, you can’t deny that SPEC OPS: THE LINE’S union of narrative and game design—and Davis’s willingness to use said design to mess with players’ expectations—was a bold statement in a genre that needed one.



IAN  
**Marsh**

DAVID  
**Marsh**  
NimbleBit

❖ “Casual airline sim” doesn’t exactly sound like the most enthralling premise for a mobile game—until you hear it’s from TINY TOWER dev NimbleBit, anyway. Kudos to NimbleBit cofounders Ian and David Marsh, whose design work on iOS free-to-play hit POCKET PLANES had us constantly picking up our phones for a quick hit of airline tycoondom. Making a profitable free-to-play game that doesn’t feel like a naked cash grab is rare enough; that POCKET PLANES is genuinely a blast is noteworthy indeed.

n/a

DEAN  
**Dodrill**  
Humble Hearts

❖ Everything animator Dean Dodrill knows about programming, he learned while working on DUST: AN ELYSIAN TALE for three and a half years—which was about 39 months longer than he had anticipated it would take to finish the game more or less by himself. Along the way, he won the Dream, Build, Play competition, made it into the Summer of Arcade promotion based on his title’s

strength, and even finished the game ahead of schedule to meet that deadline.

Aside from voice acting, music, and a little bit of writing, almost all of DUST: AN ELYSIAN TALE came from Dodrill and Dodrill alone, and we can’t help but recognize the developers who devote themselves to such projects of passion.



BENJAMIN  
**Rivers**  
N/A

❖ If we had a category for Neat Stuff, we’d probably put Benjamin Rivers on top of the list for HOME, a critically acclaimed PC adventure game that plays something like a choose-your-own-adventure horror short story. HOME is unapologetic about how it wants to be played; turn the lights off, put your headphones on, and set aside an hour or two, because you can’t save your game. We like that.

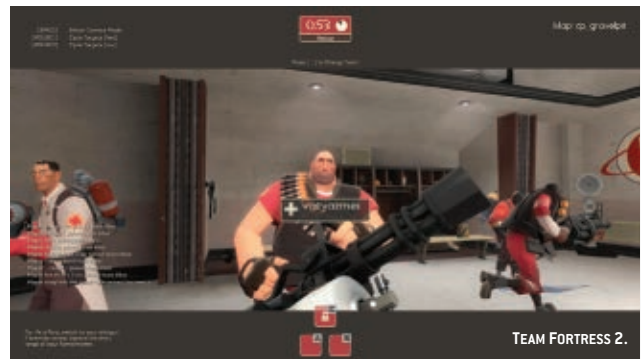
## { EVANGELISM }



DR. ELIZABETH  
**Broun**  
Smithsonian American  
Art Museum

❖ This year, the Smithsonian American Art Museum featured an exhibition called The Art of Video Games from March 16 to September 30, with still images and video footage from 80 games across 20 systems, developer interviews, historic consoles, and five playable games to represent their respective artistic eras: PAC-MAN, SUPER MARIO BROTHERS, THE SECRET OF MONKEY ISLAND, MYST, and FLOWER.

While the exhibit was curated by Chris Melissinos (with input from an advisory board and public polls), we wanted to acknowledge the Smithsonian American Art Museum’s director, Dr. Elizabeth Broun, for giving the exhibit the go-ahead. We know video games are an artistic medium, and we know you know this too, but it does



## Valve Software

❖ This year, we’re including a special Power 50 candidate: Valve Software. We could have easily padded out this list with Valve nominees across the categories, but we didn’t think that would be fair (and due to Valve’s notoriously decentralized internal organization, we weren’t sure we’d be able to find individual devs willing to take credit for specific achievements!). Instead, we decided to give Valve Software itself a nod for evangelism. Every industry needs to have a company that reminds us we can (and should) do better; we can treat our customers better, we can treat our employees and colleagues better, we can make better products, and we can even make a bit of money doing all of those.

For us, that company is Valve. To consumers, Valve is nothing but player-friendly, known for excellent support—and seasonal Steam sales that somehow make us excited to empty our wallets. To developers, Valve seems like the place to go if you want to focus on building cool things with talented people. And to everyone else in the industry, Valve is taking measured steps to push the envelope, whether it’s by turning a five-year-old core title (TEAM FORTRESS 2) into a runaway free-to-play success, starting an internal hardware development lab to prototype some virtual-reality goggles, or hiring an economist just because it sounded kind of useful to have one on staff. In an industry that seems more mercenary now than ever before, it’s nice to know Valve is still there doing the Valve thing.

us no small amount of good to see that fact recognized by a major American art institution.



JORDAN  
**Mechner**  
N/A

❖ PRINCE OF PERSIA creator Jordan Mechner has made industry news headlines a few times in 2012 (he’s got a Karateka remake in the works, and the iOS remake for THE LAST EXPRESS just came out) but we wanted to acknowledge him for something a bit more mundane—going through his old stuff in his attic and posting the contents on his blog. Before you rush off for some premature spring cleaning, let us explain:

Mechner and his crack team of digital preservationists managed to find a box with floppy disks containing the original 1988 Apple II source code for PRINCE OF PERSIA, salvage it, and post it on Github for everyone to see.



The medium of video games is still young, and the tech-obsessed part of our industry makes it easy to forget the old in our ceaseless pursuit of the new. But take a second to imagine how many of

# power 50

you reading this article have fond memories of PRINCE OF PERSIA, and you'll understand why we wanted to give Mechner a Power 50 spot for evangelism. Developers: We want you to preserve your stuff, no matter how old and busted you think it is, because we don't want anyone to forget about your hard work.



SHAY  
**Pierce**  
Deep Plaid Games

✦ When Zynga bought DRAW SOMETHING developer OMGPOP and offered all of its employees jobs, designer/programmer Shay Pierce made news headlines simply for saying, "No, thank you." Pierce's reason was simple: He had developed his own game in his spare time called CONNECTRODE, and he couldn't get Zynga's legal counsel to agree to an addendum in their employment contract that would ensure CONNECTRODE remained Pierce's property. Given the choice between potentially giving up his baby or giving up a job, he chose to quietly walk away from the deal and instead revive Deep Plaid Games, his own one-man development studio. CONNECTRODE may not be a big seller, Pierce, but darn it, we're glad you fought to keep it.



RANDY  
**Pitchford**  
Gearbox Software

✦ While the *Game Developer* and Gamasutra staff were hashing out the Power 50, one of us scribbled the following note next to Gearbox Software cofounder and CEO Randy Pitchford's name: "Evangelism—talking all the time." Really, that kind of sums it up.

Pitchford and Gearbox are positioned at the center of the American game industry; they've worked across over a dozen platforms, with several publishers, on everything from HALF-LIFE and HALO to TONY HAWK'S PRO SKATER and SAMBA DE AMIGO. They've even shown that they can grow and nurture their own IPs (see BORDERLANDS) in addition to

working with others. We think that Pitchford's experience, combined with his willingness to speak frankly about our industry, is an invaluable asset for the industry as a whole. Randy, thanks for talking all the time.



ANNA  
**Anthropy**  
N/A

✦ What makes Anna Anthropy special is that not just that she's living a brave creative life by sharing her unique perspective with the world through games; she's encouraging everyone else to take the necessary steps to do it, too.

Recognizing the homogeneity of the game development scene, Anthropy champions the emergence of new perspectives. It's her talks on inclusiveness, her own games—such as DYS4IA, which quickly and cleverly takes the player on a journey through the difficulties Anthropy has encountered obtaining gender transition treatment and being recognized as a woman—or her 2012 book, *Rise of the Videogame Zinesters*, which encourages both "freaks" and "normals" to make the games that the mainstream isn't making, that make her an invaluable voice in the expanding community of game developers.



EPONA  
**Schweer**  
Indie Bits

✦ In 2009, burnt out on crunch, Epona Schweer turned down a producer's job at L.A. NOIRE developer Team Bondi to teach aspiring game developers in Sydney. By the time the course ended in 2010, she realized there was nowhere for her charges to work, thanks to the near-total collapse of the Australian development scene. Her solution? Beef up the local indie game scene by throwing meet-ups and holding talks, collecting the power of individuals who had worked in isolation, and helping to form a thriving local scene. The lesson here is that building community



PHOTO: VINCENT DIAMANTE

## Brandon Sheffield

Game Developer / Necrosoft Games

✦ It might seem a little self-serving to include our own editor-in-chief in our yearly Power 50—but since Brandon Sheffield has left *GDMag* to focus on starting up his own game development studio (except for the occasional column and Gamasutra editorial, anyway), we figured he deserved an evangelism nod as well.

Over the last eight years (100 issues, actually!), Sheffield has worked hard to make sure *Game Developer* could offer devs a way to share their successes and failures with their colleagues so that others can learn. Internally, he has served as a sort of underdog's advocate for both the *GD Mag* and Gamasutra staff. When all our attention was on the U.S. and Japanese game industries, Sheffield was paying attention to the nascent Korean game industry; when we're following triple-A, Sheffield was following the indies; when we're following the major indies, Sheffield was spending his evenings sorting through obscure Xbox Live Indie Games.

Sheffield's advocacy extends to the people in the industry as well; he is unafraid to use the editorial pages on *GDMag* and Gamasutra to point out the industry's deficiencies, blind spots, and controversial issues when he feels that something needs to be done. Thanks, Brandon—let us know when your games come out!

requires work and ingenuity, but people fundamentally want to connect—if you can enable them.



"PETER  
**Molydeux**  
@PeterMolydeux

✦ Yes, you read that right: "Peter Molydeux," the novelty Twitter account that describes itself as "just a twisted parody based on the legendary British Game Designer," is one of *Game Developer's* Power 50.

@PeterMolydeux has been around for a while now, tweeting whimsical, emotionally evocative

ideas for games in under 140 characters. Examples range from "If I made a zombie game it would feature just one dangerous zombie, your child. You must sneak out avoiding society, trying to find help," to "Platformer where if you fall in a pit you're trapped forever unless you can emotionally manipulate nearby enemies to pull you back up." Of course, these tweets are inspired by the real Peter Molydeux's bombastic descriptions of the work he does (or wants to do).

At first, we simply laughed at our industry in-joke. But then a strange, wonderful thing happened: People across the world, devs

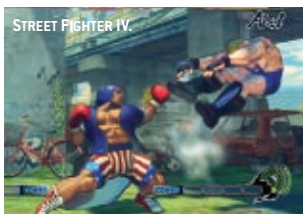
and players alike, realized that, well, some of these ideas sound pretty good, and we'd rather spend a weekend trying to make and play those games instead of the ones we spent Monday through Friday making and playing. Yes, the parody Twitter account accidentally inspired a worldwide grassroots game jam called "What would Molydeux?"—and Molyneux himself even showed up to the London chapter!

Wherever you are, @PeterMolydeux: Thanks for the laughs—and the inspiration.



**YOSHINORI Ono**  
Capcom

✦ Last year, STREET FIGHTER steward Yoshinori Ono made the Power 50 for his design work bringing the franchise back. This year, he's making the list again—but for evangelism, not design.



In a remarkably candid interview with *Eurogamer*, Ono explained that he had endured a medical emergency brought on by work-induced stress—a medical emergency involving an ambulance and a blood acidity level “on par with someone who had just finished a marathon.” During that interview, he called Capcom out for overworking himself and other employees, scheduling an unreasonably intense promotional tour, and forbidding its employees from organizing a union.

Game developers know they're in a tough business, with many grueling schedules and unforgiving crunch periods. But while it's one thing to acknowledge the business as a whole is tough, it's another thing entirely to speak on the record about how bad your employer is for your health. Shoutouts to Ono for saying what needed to be said.

## { PROGRAMMING }



**BOYD Multerer**  
Microsoft

✦ Microsoft's XNA framework (and associated dev tool XNA Game Studio) has been something of an unsung hero for indie devs over the last console generation, and since its future is in question (XNA applications won't be included in Windows 8's Metro UI or app store), we thought it only fair to give XNA—and Xbox director of development Boyd Multerer—proper acknowledgement.

XNA has made it easier for small-time indies and hobby game devs to make games and put them on Xbox 360s, Windows phones, and PCs around the world. We're fans of tech that democratizes game development, and XNA was unprecedented in terms of how available and accessible it made the Xbox 360 and Windows Phone 7 platforms. We're hoping that XNA sticks around in some form—there are a few projects out there working to adapt XNA to other platforms, which could eventually enable XNA devs to build games for Metro, Android, iOS, Mac OS, and PlayStation Mobile—but even if the worst happens and XNA falls by the wayside, we want to salute Multerer for his excellent work.



**NIKLAS Smedberg**  
Epic Games

✦ By now, it's no secret that the Unreal Engine can make mobile games look amazing—and some of that credit goes to Epic Games's senior engine programmer Niklas Smedberg. Between Smedberg's under-the-hood look at mobile GPUs at GDC 2012, his work on the post-process graphics effects on the INFINITY BLADE series, and his current work on Unreal Engine 4, it's pretty clear that if you want your mobile game to look like it came straight from a console, he's the go-to guy.

n/a

**JAROD Pranno**  
Phosphor Games

✦ Unreal Engine is a great piece of tech, but we can't forget to show some love to the devs out there who make it sing—and Jarod Pranno, studio art director on Phosphor Games's HORN (iOS) did just that. With Horn, Pranno demonstrated that Epic Games/INFINITY BLADE dev Chair Entertainment aren't the only ones who can make a great-looking mobile game, and we're eagerly paying attention to see what Pranno and Phosphor will be doing next.



**DEREK Yu**  
Mossmouth

✦ Some games so tightly bind their programming and design together that it's hard to truly determine who deserves the credit. One such is Mossmouth's brilliant SPELUNKY, which released this year for Xbox Live Arcade. Its randomly generated levels are the cornerstone of the game's addictiveness—and a marvel of designer Derek Yu's algorithmic design. They're always navigable, always fun, and ever changing. You'll never complain that they weren't created—or at least not directly—by human hands.

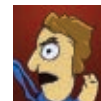


**PATRICK Wyatt**  
N/A

✦ Patrick Wyatt is practically the definition of “industry veteran”; between his stint at Blizzard leading the original Battle.net, cofounding GUILD WARS dev ArenaNet, and more recently working as En Masse Entertainment's COO (TERA), it's hard to find an MMO that doesn't have his fingerprints on its network code.

When looking at a new MMO, it's easy to overlook the underlying nuts and bolts that keep customers happy. Wyatt's work on the platform underlying TERA's account management, billing, and other

functions he described to *Game Developer* as “all the other unsexy parts of games” has shored up many player-experience design flaws others simply consider a fact of MMO life—such as beefing up account security, filtering spam from chat, building in better analytics to improve player retention rate, and so on. More recently, Wyatt has been making efforts to share his knowledge on game server code by writing articles on his blog at codeofhonor.com and giving in-depth talks at the Game Developers Conference.



**SIMEON Nasilowski**  
Two Lives Left

✦ There is something to be said for programmers who work on making programming more accessible to a wider range of people. Two Lives Left's Simeon Nasilowski did just that with CODEA, a newbie-friendly iPad app that lets you quickly build game prototypes with Lua (see the June/July 2012 issue of *Game Developer* for the review). With CODEA, anyone with an iPad and \$10 can start dipping their toes in the game-dev pool, and we think that's pretty cool.



**JOACHIM Ante**  
Unity Technologies

✦ As we're wrapping up 2012, we've seen one very clear game-dev trend: Everybody loves Unity. Whether you're an experienced dev in a major studio tasked with throwing together a quick-and-dirty prototype, a small-time indie studio looking for an off-the-shelf 3D engine to build a game for multiple platforms, or just a hobbyist dev throwing together a fun project for a game jam, you'll probably be using Unity. Unity Technologies CTO, cofounder, core development team lead Joachim Ante has been central to that success; under Ante's leadership, Unity has blossomed into a tool that is powerful, polished, and relatively easy to use. 📌

shoot many robots

## midmortem

BY ALBERT REED, CEO, DEMIURGE STUDIOS

**WE'VE BEEN MAKING GAMES HERE AT DEMIURGE STUDIOS FOR 10 YEARS,** during which we've gotten used to a certain way of making them. We select our production practices, technologies, and even people with the idea that we need to package a perfect product that will ship to customers and never be touched by us again. Despite that background, we decided to take a stab at building a free-to-play game-as-a-service. Normally, *Game Developer's* development postmortems are written after a game is finished and on store shelves, but since ARENA KINGS, like many free-to-play games, is still being built, we felt it'd be fair to call this a "midmortem," and recap what has gone right (and wrong) with the process so far.

ARENA KINGS is a player-versus-player spinoff of SHOOT MANY ROBOTS (SMR1), our recently launched XBLA/PSN/PC run-and-gun. Shortly after SMR1 shipped, we spun up a small team to slap together a prototype of a 2D deathmatch game mode targeted at the \$10 downloadable market. The results were very encouraging. People at the office couldn't put down the controllers, and there was gleeful shouting throughout each playtest. The prototype exhibited a nice mix of competitive skill needed for PvP, and bombastic randomness that is characteristic of the SHOOT MANY ROBOTS franchise. So, we packaged the demo, built a more comprehensive plan and a PowerPoint deck, and took it to various publishers to try to raise capital to build the game. >>>>>>



# 11

# 15



# ARENA KINGS

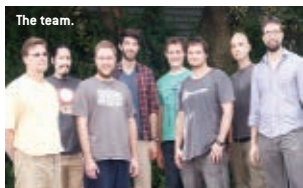
midmortem



## 1 what · went · right STANDING ON THE SHOULDERS OF SHOOT MANY ROBOTS

We should begin by acknowledging that we got a head start thanks to SMR1. Once we turned on friendly fire in SMR1, we were pretty far along from a production standpoint to creating player-versus-player combat in the SHOOT MANY ROBOTS universe. I'd wager that many established developers also have technology and IP from a previously shipped game that could find new life as a digital title.

One of the surprising successes of SMR1 was the attach rate of our in-game currency and weapon sales. We didn't have any analytics other than the leaderboards in that title, but we put together some info from snapshots provided by Ubisoft, our publisher. Because of that gap, the data is a little opaque,



but here's what we were able to deduce in the first few days after release:

**OUR IN-GAME CURRENCY  
"NUTS" GENERATED \$0.70 ARPU  
(AVERAGE REVENUE PER USER).**

**DURABLES IN THE FORM OF  
GUNS AND GEAR GENERATED  
\$1.00 ARPU.**

With in-app purchases, we were able to effectively lift the retail price of SMR1 by 17 percent. We applied a very light touch to the in-game purchasing functionality and, frankly, grafted it onto the game as an experiment pretty close to its ship date. If you ask anyone in the free-to-play space to evaluate the design and functionality for our in-app purchases, they'll tell you there's a lot of room for improvement. Nonetheless, something about our mechanics was working great, so we felt strongly that we could reproduce and build on that success in our next title. The final edge we picked up from the launch of SMR1 was the ability to leverage the social-networking integration included

in the PC version. As we'll describe later, our social platform (called "duckduck") gave us a highly valuable connection to our previous customers and got us on the path to building game services.

**2 "LAUNCH" MENTALITY**  
Perhaps the most exciting thing about developing for the PC rather than console was the opportunity to make a game based on what our customers were telling us, rather than trying to speculate about what features or content they would find appealing. Good developers all strive for quality, but most of us are left to decide what that means based on our own subjective views, previous experience, and the limited user-testing we can do prior to shipping. We had to change this mentality to "*We are not the sole arbiters of quality*" instead.

Developers can debate all day about how minimal a viable product can be, so we cut short that question by saying that it's not up to us alone to decide. Only by listening to our customers can we confirm that our decisions and priorities are the right ones. That realization pushed us to actually put the game

in the hands of customers before we felt it was viable (for example, it didn't yet have the e-commerce solution). We were terrified at the thought of putting a game with our studio and IP brands attached to it in the hands of total strangers when we knew it wasn't yet good enough. This led us to another mentality shift: "*No fear.*" (As an aside: We seriously considered naming all of our new development practices after mid-1990s T-shirt slogans, but "co-ed naked game development" appealed to absolutely no one.)

We launched the game in a tightly closed beta. People played it, complained (a lot), and the dev team saw that the studio walls didn't come crashing down around them. Our transformation was in many ways complete. Liberated of a fear that any game that left our office needed to be Metacritic-ready, we became eager to test new ideas and engage our audience in a conversation about what they valued—what they viewed as "quality."

To keep our first release's scope under control, we applied this litmus test: "Is including this feature worth delaying our chance to learn from customer feedback?"

There were some things we just couldn't skip, such as making a new logo and splash screen. Other items, like swapping out the programmer art in our new scoreboard, were skipped in the push to start testing the game.

Without changing the way we as a studio thought about making games, we wouldn't have been able to put a solid game in our PC customers' hands with only four weeks, an Xbox 360 prototype, and a team of six developers.

### 3 THE PILE

Getting the game launched and then providing regular updates required that we be more nimble than the milestone-driven wings of our business. Teams at Demiurge follow pretty strict Scrum methodology, but we opted to try something different for ARENA KINGS because we wanted to be able to realign near-term objectives more often than every sprint. With a small team, we also felt comfortable giving the team members flexibility to decide what was most important.

To get to launch, we had our team focused on getting the product in the hands of customers. Because we already had a working

prototype, folks were able to see our first milestone without needing a long-term schedule. Out of this situation we put in place a light task-management scheme affectionately referred to as "The Pile." For our first few releases, we picked specific scope goals that were printed out and hung on the wall. These either took the form of a user story or a hypothesis we wanted to test. Anyone could throw a task they felt necessary on The Pile and tag it for a particular release. Likewise, anyone could grab a task to do off The Pile. Tasks also were given a priority and everyone was instructed to take things off the top of The Pile. We held daily stand-ups, which gave the project lead the ability to catch folks veering off-course before they had sunk more than a day's worth of work into something. Everything was tracked in Jira, but it could have easily been cards, sticky notes, or a bug database. We could devote a whole article about the merits and drawbacks of The Pile compared to Scrum, but we'll save that for another time.

Our first few releases were scope-locked. When the user stories were implemented and the hypotheses were ready to be tested,

we released an update to current players. We eyeballed the amount of work in a given set of goals and tried to keep it to around a week's worth of development. That functioned reasonably well, but we wanted to be providing updates to players every week—and (unsurprisingly) when we fixed the scope, the schedule tended to slide around. We felt that providing weekly updates would be key for driving our retention rates (we'll get to that later), so we switched around to saying that we were going to try to release Monday morning's build regardless of whether we met our goals. The idea of releasing a lightly tested game on schedule rather than when it was done took another "no fear" leap.

Shortly before we began work on ARENA KINGS, we hired a web developer from outside the game industry. He brought with him a very different way of looking at releases than our company did. His work on our social platform was pushed live nearly every day after being locally unit-tested and smoke-tested by QA. These small-batch releases tended to not be prone to major failures, and the few times something broke, we detected the problem quickly and pushed up a fix. Seeing this focus on Mean Time to Recovery (MTTR) gave

us the courage to try to do the same with our game. Unit-testing alone wasn't going to sufficiently test the software, so we had our QA team spend a couple of hours with the game each day.

On Monday morning, we branched the build for release. We looked at The Pile and moved everything off to the next release if it wasn't ship-stopping, or assigned it to be taken care of right away at the morning stand-up. Most of the released versions of the game had less than one hour of testing on them with a four-person test team. Ten updates in, we've never had to release an emergency patch.

One might think that this fast pace would lead to late nights and lots of crunch. The opposite proved to be true—the entirety of ARENA KINGS development to date has been crunch-free.

### 4 THIRD-PARTY LIBRARIES

We got huge velocity improvements from our team by using as much off-the-shelf or pre-existing technology as we could. Relative to most indies, our costs are high, so we tend to select tools that save us the greatest number of man



# ARENA KINGS

midmortem

hours—and usually, those are the most expensive ones.

ARENA KINGS is being distributed to our testers via Steam, and we're making full use of the Steamworks platform for achievements, leaderboards, friends, and so on. The engineers like working with the Steam APIs, and Steam's online services have extraordinary uptime. We used their local emulation of the distribution services to deploy builds to QA so that QA had the exact same experience as our customers. Being able to smoothly distribute and update ARENA KINGS was critical to our success; historically the process of pushing a build to Steam involved a lot of manual effort, but Valve kindly allowed us to test-drive a new system that allows developers to push live builds themselves. This update made our MTRR tiny, and allowed us to exactly time the release of updates with our email campaigns. Having the patches delivered automatically, with minimal size and in the background, removes one of the biggest pieces of friction that exists in the game-as-a-service world on the PC. We highly recommend it. Long-term, we want to offer the game via other means to maximize our cut of our customers' expenditure, but to get off the ground Steam was the clear choice.

One of the odd quirks of developing free-to-play games is that our players need to have some

sort of account to associate with their save file. They need to be able to retrieve this account if they forget their password, and we want to be able to reach out to them directly with information about happenings in the game. User identification is also critical to making analytics effective. Most web-based games need a solution to this exact problem, but these systems tend to rely on private keys being stored server-side to communicate with the authentication and microtransaction systems. We had the somewhat unusual requirement that our game client ran locally on individual computers. We also felt strongly that we wanted to have direct access to our customers via email, so using Steam's systems wasn't going to work.

Bizarrely, we couldn't find a single off-the-shelf system that met our requirements. In the end, we updated duckduck, our back-end social services platform, to have this functionality. Users can create accounts or automatically wire their accounts to ARENA KINGS using Facebook via duckduck. The same tool also allowed us to pull in the names and profile pictures of our players so that we could display them to their Facebook friends and do matchmaking via real-life friends. (Yes, we're thinking of offering the service as middleware to developers.)

To make microtransactions work, you need both a payment

back-end and an online service to track what goodies the player has bought. Thanks to the success of Facebook and mobile gaming, there are tons (seriously...tons) of providers that will handle one or both of these services. We had two unique requirements. First, we wanted to take advantage of the fact that our initial users would all be Steam-based and offer Steam Wallet as a payment option. Second, we had the issue of being a C++, PC-based game, so we needed a solution that allowed untrusted clients to run the game logic, and preferably one that offered a C++ library. In the end, we opted to go with the Player.IO tool suite. The team at Player.IO offered absolutely outstanding customer service, and their engineers quickly answered all our questions so our team could focus on making the game.

## 5 TABLEAU

Since the team needed to test their hypotheses, we added analytics to the game. Most of the issues stemmed from figuring out how often users are returning and how many games they are playing. We were able to use duckduck to gather data about each of the matches being played.

What we *didn't* have was a way to meaningfully break down the huge amount of data. Tableau to the rescue! Tableau is an analytics

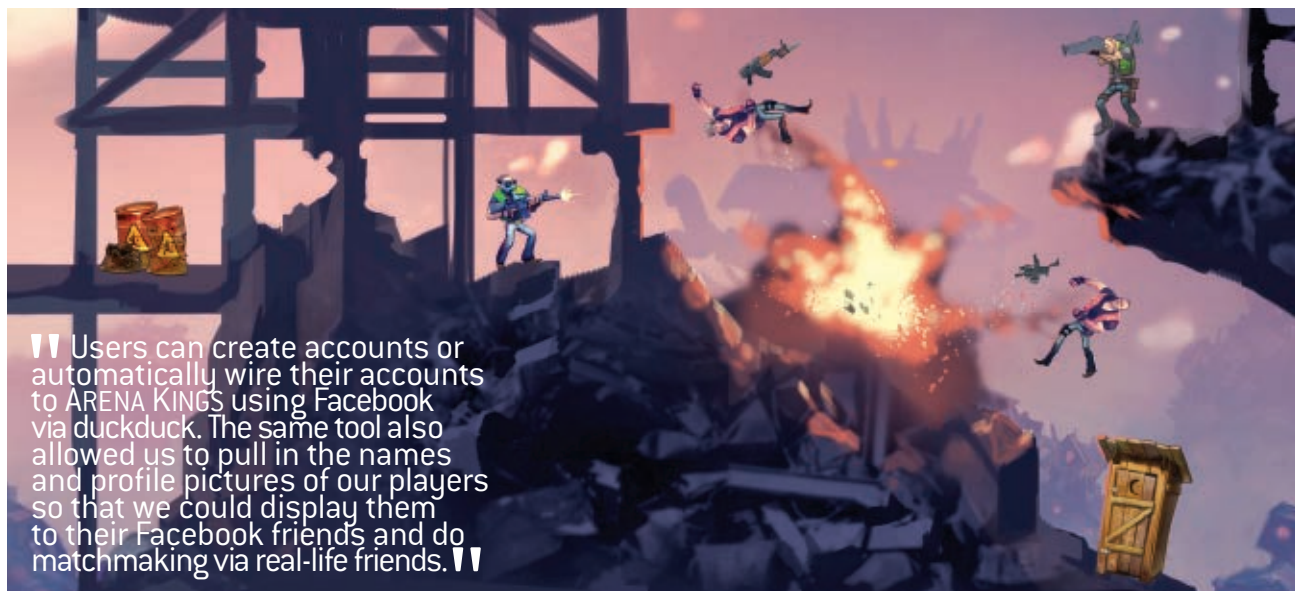
tool that makes it easy to filter and manipulate data and then build charts and graphs. Tableau solves our data visualization problem elegantly, but it took some time to learn how it works. We've previously used other analytics packages, and usually they provide both the data-gather and data-visualization, which can get frustrating when we can't get that package to display the data the way we want to see it, or perform a particular analysis.

By making it easy to answer random questions ("When are our North American customers playing?" and "How many games on average are those who joined between Updates 2 and 3 still playing?") without involving an engineer to do a database query or adding some new specialized instrumentation, we are able to learn way more about our players' behavior and optimize engineering resources. It's super-pricey, so for now we have it set up on a shared computer.

## what · went · wrong

### 1 RETENTION

Early on in ARENA KINGS'S development cycle, when the user base was just a few hundred people, we needed a way to gather qualitative feedback about the game. Someone came up with the idea of building a Google



▸▸ Users can create accounts or automatically wire their accounts to ARENA KINGS using Facebook via duckduck. The same tool also allowed us to pull in the names and profile pictures of our players so that we could display them to their Facebook friends and do matchmaking via real-life friends.▸▸

Docs-based survey and having that automatically pop up in a browser when players quit. We shoved the ID of players into this spreadsheet automatically, along with a timestamp so we could later tie feedback into user behavior. In the end, these are the fields that have proven most useful:

**HOW MUCH FUN DID YOU HAVE THIS SESSION? (SCALE OF 1-10)**  
**MOST FAVORITE PARTS**

**LEAST FAVORITE PARTS**

**BUGS—DID YOU NOTICE ANY?**

**ARE YOU EXCITED TO PLAY AGAIN? WHY OR WHY NOT?**

**ANYTHING ELSE YOU'D LIKE TO TELL US?**

Without any fancy technology, this simple survey form told us some of our biggest problems over the first few updates:

**USERS NOT BEING ABLE TO FIND ANYONE TO PLAY WITH**

**LAGGY GAMES WHEN THEY DID FIND SOMEONE TO PLAY WITH**

**PALPABLE HATRED FOR ALL THINGS FACEBOOK (WHICH WE REQUIRED UP UNTIL VERY RECENTLY—MORE ON THIS LATER)**

Solving the lack of opponents was a gnarly problem and highlighted a previously overlooked project risk:

Getting PvP games off the ground is challenging because you need a critical mass of players of varied skill *active* at any given moment. In hindsight, tackling a game with no single-player option as our first free-to-play game was a mistake.

In the end we overcame our player-count problems by sending out beta keys to all to the duckduck users from SMR1. Our battle with lag is an ongoing technical challenge.

Once we had the game live, the team got together and laid out goals for weekly active users (WAU), percent paying users (PPU), and the average number of dollars spent per week by each of those paying weekly users (we'll abbreviate that with ARPPU). We opted to measure things by week since that was the cadence of our development. Each week we got together and asked ourselves how we were tracking relative to our goals, and pushed to implement features we thought would have positive impact on those numbers.

Our first update that included the ability to buy in-game currency gave us:

**PPU: 1.5%**  
**ARPPU (PER WEEK): \$1.25**  
*(Tableau was critical here—we needed to filter out all Demurge employees!)*

This was pretty bad by industry standards, but given that we were still "beta" and our players knew we weren't finished yet, we were pretty pleased that the basics seemed to be working. We monkeyed a bit with the price and added a bunch more

User join	Age of users by releases			
	0	1	2	3
2	100%	38.9%	16.7%	16.7%
3	100%	26.7%	12.2%	6.7%
4	100%	18.5%	14.4%	0%
5	100%	14.4%	0%	0%
6	100%	0%	0%	0%

**FIGURE 1:** This table shows what percentage of users that started playing during a given patch stuck around ARENA KINGS for subsequent patches.

goodies to buy and a couple of updates later the numbers moved a bit:

**PPU: 1.2%**  
**ARPPU (PER WEEK): \$2.12**

These numbers aren't great either, but up to this point we had made very little effort to tune our economy, offer a good variety of items for purchase, or reduce friction. We were confident those changes would improve PPU and ARPPU, but we had significant problems maintaining our user base, so we turned our attention to WAU.

Because we're in a tightly closed beta, our WAU measurements weren't very meaningful—we could juice those numbers by simply sending out more beta keys. The meaningful statistic behind WAU is our retention. To make our retention numbers even more useful, we broke our players into cohorts by the update in which they first played the game. Using Tableau, we spit out this table (see **Figure 1**) without any new work from the engineering department.

Along the vertical axis is the update during which the player first joined. Along the horizontal axis is the number of updates for which

the users stuck around. (Note that in this chart, the 0 percent entries are there because we don't yet have data for the seventh update integrated.) We can also view this same data by week (see **Figure 2**).

This cohort analysis allows our development team to drill down to the exact impact of the new features and react accordingly. With this analysis, we were able to start working to address the issue of retention around update six. Our approach was to implement features that directly went to the game's stickiness by making progress feel more satisfying, adding achievements, and increasing the amount of content in our RPG systems. As you can see, those efforts haven't yielded much improvement. Our next effort will be to make more fundamental changes to the game. Retention is a great measure for how sticky your game is, but we think it's also a fair measure for how much people actually enjoy playing. We've got work to do!



**EMAIL SERVICES**

Email communication proved to be an immensely useful way of engaging our existing customers and

Week of first place	0	1	2	3	4	5	6	7	8
May 13, 2012	100%	66.7%	33.3%						
May 20, 2012	100%	28.6%	28.6%	21.4%	7.1%	7.1%	7.1%		
May 27, 2012	100%	20.8%	12.5%	4.2%	4.2%	4.2%	8.3%	4.2%	8.3%
June 3, 2012	100%	12.2%	7.8%	4.4%	1.1%	3.3%	3.3%	1.1%	
June 10, 2012	100%	12.3%	7.5%	6.8%	6.2%	3.4%	1.4%	3.1%	0.3%
June 17, 2012	100%	8%	7%	6%	2.7%	2.9%	1.2%		
June 24, 2012	100%	9.9%	6.4%	5.5%	2.5%	1.9%	0.5%		
July 1, 2012	100%	4.7%	2.4%	2.8%	1.7%	0.4%			
July 8, 2012	100%	7.1%	3.6%	4%	0.4%				

**FIGURE 2:** Weekly active user retention rates for Arena Kings. Note that the far right column for each row has incomplete data.

# ARENA KINGS

midmortem

reaching out to new ones. There are many options out there for sending bulk emails. In our case, we needed to bake beta keys into the emails and wanted to be able to easily import our spreadsheet-based contact database into the mailer. We ended up going with Constant Contact, based largely on recommendations from peers with prior experience. During one of our key updates, we were pushing out beta keys to about 10,000 customers.

Timing was crucial for this period, because our goal with this push was to get our active user count high enough so there was always a game to join. Without warning, Constant Contact opted to put us on a probation of sorts whereby they sent emails in small batches to make sure that they weren't getting too many spam reports. They waited until we hit "send" rather than prompting us with a warning when we built the list. This probation jeopardized the effectiveness of our campaign. To their credit, they provided outstanding customer service by phone—tutoring us on the ins and outs of email marketing and walking us through what was going to happen, often at night and over the weekend. Despite that, I wouldn't recommend them as an option going forward.

## 3 FEARING THE PREMATURE "ANNOUNCEMENT"

Up until recently, all of our beta players came to us without any meaningful PR. Because we need concurrent testers to keep the game alive and because our retention was low, we called our email database of potential testers "Rocket Fuel"—a precious commodity to be spent only when it would do us good. We spent countless hours trying to figure out how to gather beta testers for the game without spoiling the opportunity to announce the title and get the resulting PR bump.

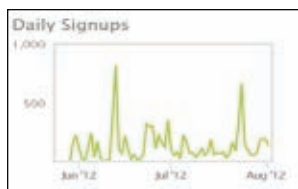


FIGURE 3: Sign-up graph for Arena Kings.

Ultimately, we didn't need to worry. We did just about everything we could think of (short of directing outreach to media outlets, posting to forums of similar games, and even using our Facebook/Twitter feeds), and still our PR team was able to get pretty good coverage by "announcing" the game long after it was quite public.

Gathering testers was critical to our ability to evaluate progress, and holding back on gathering those testers impeded the game's evolution, but there were some successes in our user-gather efforts. We wanted an easy way for people to sign up to be in the beta, and didn't want to take the time to muck around with our existing websites. LaunchRock did exactly what we needed. A web form to gather emails isn't a complicated feature, but their system took us just a few hours to set up and never went down. We did get a lot of bogus emails, but overall their tools were great.

We pulled users to the LaunchRock site by posting to online forums for similar games and by posting to the SHOOT MANY ROBOTS and Demiurge Studios Facebook/Twitter feeds. Check out **Figure 3** to see what our sign-up graph looks like currently.

We expected to get the biggest spikes in new sign-ups and active players when we announced a new update. However, the large spike on the left actually correlates to the sending of the first wave of beta keys. As soon as word spread that we were live, we got a huge influx of new potential players. This trend continued—at the moments we sent beta keys, we got more beta sign-ups. We capitalized on the word of mouth we were getting by beginning to send out four keys to each player. In addition to further generating buzz for the beta, it helped players find games by coordinating with their friends.

To date, we have about 10,000 sign-ups through LaunchRock. Our bounce rate to those emails is usually between 1 percent-3 percent, and our open rate (measured by people displaying images in the email) is in the range of 30 percent-40 percent. As we saw the sign-ups come in, it was clear there was a lot of abuse taking place—many sign-ups were

coming from the same IP address. We gradually just ignored that issue since it wasn't causing any harm. Because Steam doesn't allow us to track the use of each key, there's little way for us to determine what happens to users until they fire up the game for the first time.

## 4 REQUIRING FACEBOOK

We received a ton of negative feedback to our Facebook-based account management. Early in the development, we flipped off all posting to activity feeds to test whether that was the source of the ire, but it had little impact on the complaints. Our analytics were also showing that users were dropping out at the registration step. Uptake on the totally optional registration with SMR1 was quite high—around 30 percent during the first couple of weeks, so the backlash was unexpected. We think the key difference was that by presenting users with a choice, they found the idea of Facebook integration less intrusive. While we worked on a Facebook-less registration process, we improved our funnel by telling users what the Facebook application was used for and by including a "coming soon" preview of our account-creation system. We think that this nicely explained the purpose of Facebook.

## 5 MATCHMAKING AND LAG

The biggest complaint received from customers on the feedback form centers around lag in the games. Clearly for a PvP experience, having a fair, responsive multiplayer environment is critical to our success. Our engine does not currently have support for dedicated servers nor did we want to incur the complexity and overhead of setting up a bank of those sorts of servers while we were still trying to learn about the game's design. Instead, we focused on improving the peer-to-peer matchmaking and communication to the player about what is happening during that process.

We also tried to come up with a metric for determining connection quality. On our console titles, we had great luck using simple ping times but on the PC and using Steam's peer-

to-peer libraries, we had no means of testing the quality of the connection between peers without opening up a real connection to each potential host and then testing the connection quality. That has the potential to be very slow for users if there are many servers running and we never managed to get it working reliably.

In the end, we used some of the geographic information provided by Steam as a very rough proxy for how good the connection between two players was likely to be. It didn't work very well, though. We also tried to work the issue by providing players with better messaging. We added a little dialog explaining to the users during the search process that we hadn't been able to find any high quality games and offering them the option to connect to poor-quality games or just wait longer for a match. At least that left players understanding why their experience was sub-par during the match rather than simply thinking the game was broken.

## ! NOT DOING THIS STUFF SOONER

One of the most exciting developments at Demiurge has been our application of these lessons to our traditional game development business. One of our internal projects is now doing "weekly releases," where developers push a stable, testable build to a set of dev kits sitting in our lounge. Other developers at the studio stop and play the game and then submit feedback on their experience through a web-based form. At our weekly staff meeting, we even use a show of hands to see what their WAU numbers look like. Those changes have helped focus the team and significantly improved the software's stability. Soon, we'll expand that system to include one-time on-site testers. It's also wonderful to get to see big, meaningful changes happening to our in-development games every single week. 🎮

**ALBERT REED** is the CEO and one of three co-founders of Demiurge Studios. He graduated from Carnegie Mellon University, where he and fellow founders Chris Linder and Tom Lin shared a common love for pizza and recreational modding. He can be reached at [al@demurgestudios.com](mailto:al@demurgestudios.com).

# 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.



VFS STUDENT WORK BY NIKOLAS LAZAR

VFS

Find out more.  
[vfs.com/enemies](http://vfs.com/enemies)



## GAME DEVELOPER MAGAZINE

the best of postmortems, product reviews, and standout columns

GET THE **PRINT+DIGITAL** ACCESS BUNDLE FOR ONLY

# \$49.95

 /YEAR

- + DIGITAL ACCESS TO BACK ISSUES
- + EXCLUSIVE INTERACTIVE EXTRAS

**INCLUDES:**

PRINT SUBSCRIPTION

DIGITAL + GAME DEVELOPER APP

+

**BONUS!**

BEST OF POSTMORTEMS PRINT ISSUE

**SUBSCRIBE TODAY!**  
[GDMAG.COM/SUBSCRIBE](http://GDMAG.COM/SUBSCRIBE)



## The best ideas evolve.

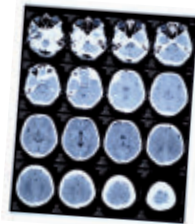
Great ideas don't just happen.  
They evolve. Your own development  
teams think and work fast.

**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](http://perforce.com/try20)**

Or request an evaluation license  
for any number of users.







### UNDERSTANDING BROADCAST METERING SPECS

★ Let me qualify this before you stop reading: I don't think loudness metering is inappropriate for games. What I think is inappropriate is the specification most people keep talking about: **EBU-R128**. I'll come back to why I think this is wrong for games, though, because I want to talk first about the specification you should be focusing on instead.

**ITU-R BS.1770** is a metering specification embedded with three important qualities. First, it employs a scale that emulates human perception of the frequency spectrum (thanks to the K equalization curve). Second, that measurement takes place over a period of time, because the duration of a sonic event affects the way we perceive it. Third, it specifies a need for "true peak" metering, a way to calculate intersample peaking and to correct for failures of Peak Program Meter (PPM)/Quasi-PPM integration time. These are the elements that are most important to us as listeners. Any meter that is ITU-R BS.1770-compliant will meet these requirements.

The awesome thing about BS.1770 is its emulation of human perception. This method comes closer to measuring audio in the way we hear than any previous meter does. Let's say you had two sound files that measured -30 LKFS (that's "Loudness, K-weighted, relative to Full Scale"; for those who are more familiar with the nomenclature LUFS, the two are analogous). One file has no frequency content above 300Hz, and the other is broadband and noise-like. That equal loudness measurement means that when played through our speakers, the noise is perceived as being equal in volume. Those same two sounds adjusted to the same meter reading in another scale, dBFS or LEQ-A for example, will likely not be perceived as the same volume. This measurement scale provides subjective predictability, regardless of spectral content, which is a good thing.

### WHY BROADCAST MODELS AREN'T GOOD FOR GAMES

★ Now let's talk about EBU-R128 (and lump in ATSC-RP A/85, for

good measure). These broadcast specifications are implementations of ITU-R BS.1770. They do not specify metering criteria, other than the fact that BS.1770 is required. They specify measurement criteria. What they state is that a specific loudness measurement, over "infinite" duration, must be met by broadcast materials. ["Infinite" simply means that the measurement must be averaged over the length of the program or advertisement.]

These broadcast specs also dictate that the measurements also need to remain relatively static throughout the show; your measurement at the end needs to match measurements earlier in the program. To achieve this immovable magic number, the dynamic range of the program is necessarily limited to a narrower state—nowhere near as narrow as we find in pop music, but not as wide as film either. Because the content of the piece is perfectly predictable (remember, we're discussing broadcast right now), it's possible to work some dynamic range back into the piece.

In a game, your measurement is going to be affected by the duration of gameplay states (travel, dialogue, stealth, battle, and so on). How do you predict the duration of those states when they are, in most cases, entirely dependent upon the player? If you have too many sustained quiet moments, your measurement will drop, and the opposite is true as well. You could also easily have a section that goes beyond reasonable volume limits, but still have a valid infinite measurement. Working under an "infinite" measurement window in this scenario would likely force you to emulate a broadcast mix and add compression to hit that magical number—or possibly move closer to the kind of compression found in pop music. There may be appropriate cases for this, which I'll touch on later, but this shouldn't be the ultimate goal.

Personally, I like to play games that have some dynamic range to them. I don't need a film mix, but I want to feel that the sonic experience is appropriate for the situation onscreen. If you're trying to model a broadcast spec—a specific number over infinite duration—you

might end up with a game where explosions on an action sequence are the same perceived volume as footsteps during a stealth section of gameplay. Unless there's a specific reason to do that in terms of narrative or game experience, you're losing opportunities to create impact.

People have recognized this, and I think that's why there's been a lot of head scratching over the implementation of loudness metering. While it's possible something modeled after the broadcast specifications may work—and could even be ideal—for browser-based social games or mobile phone platforms, no one has figured out how to effectively apply it to console and PC games.

Game audio folks: These broadcast specifications that you've been hung up on are the problem. EBU-R128 and ATSC-RP A/85 are useful for games only if you're trying to understand the logic behind them. You need to first understand why they work in the broadcast conditions they're designed for, then go back to the source, ITU-R BS.1770, and develop your own set of rules. I even have a few suggestions to get you started.

### NORMALIZING LOUDNESS IN YOUR ASSETS

★ Remember why BS.1770 is so awesome? That predictability of perceived loudness could be the goose that lays a golden egg of workflow efficiency. Let's say that your game's sonic assets fall into the following categories: ambient sounds, small weapons, character/vehicle movement, music, and user interface. Start by defining a target loudness measurement for assets in each category. If you can predict that all of your ambient assets are the same perceived volume, wouldn't it make it easier to mix and transition from one ambient area to another? What if, months after your initial implementation of weapon sounds, you had to create three new weapons? Or perhaps you've been working with placeholder music and have just received the final versions. If you knew that all of your existing weapons measured at -27 LKFS, you could easily create

new sounds for implementation into the existing mix engine.

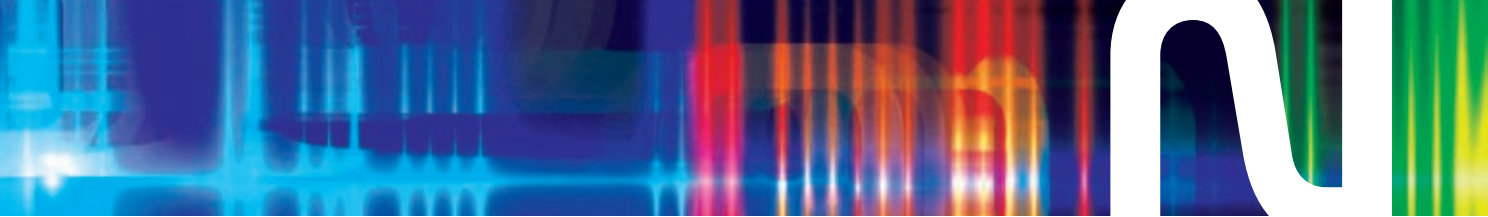
You could conceivably create a fairly detailed rough mix using only a limited set of assets in each category, because your in-house loudness standards established for each category of sound asset make it easier to work in new sound assets later without worrying about your levels. This would leave more time for you to focus on engine optimization, using system resources, or polishing your game's final mix.

If it's so critical to create predictable behaviors in the sound engine for that real-time mix, why not give that engine predictable material? NuGen Audio even has a piece of software called the Loudness Management Batch Processor that will take care of it for you, though I haven't tried it myself. Simply drag your files to the In folder, and the software analyzes and processes the sounds to match your loudness and peak-limiting settings. Done.

### "SHORT-TERM" MEASUREMENTS AND LOUDNESS RANGES

★ PROTOTYPE 2 audio director Rob Bridgett recently shared some interesting numbers from his work. Here are some "long-term" loudness measurements from specific states of the final mixed game: cutscenes -23, stealth and dialogue-heavy gameplay -23, action -19, "insane" action -13. These states were mixed artistically for maximum impact and experience, which is important for the artistic nature of games. Picking one infinite loudness measurement to suit this game would not only be a difficult task, it would lessen the audio team's creative contributions. One solution is to work in smaller measurement windows and to utilize a minimum and maximum measurement range—a pair of loudness thresholds.

When I say smaller measurement window, I'm referring to somewhere in the neighborhood of five seconds—maybe as short as two. In this manner, the loudness measurement will more accurately reflect the in-game events as they happen. Because I'm arguing for the use of dynamic range in games, a



target range is more appropriate for short-term measurements. Establishing this would require the identification of anchor states. When should the game be loudest? When should it approach the bottom volume threshold?

If we use PROTOTYPE 2 as an example, we're going to be looking for archetypal moments of "insane" action and stealth, respectively. Start with those two moments and get a good idea of where the audio should sit. Those become the anchor points and inform the rest of your mix. From then on, the goal is to keep the loudness levels between the two thresholds. Mix artistically, and only worry if you exceed or drop below the respective loudness levels.

With the exceptions of examples of social and mobile games, I think there is only one appropriate scenario for a specific target measurement in games. Start with a dynamic mix that provides flexibility, and then take a page from DICE's playbook for BATTLEFIELD 2—more on that in the next section.

## PLAYER-CONTROLLED DYNAMIC SETTINGS

★ BATTLEFIELD 2 did something interesting with the dynamic range—namely, it put that range in the player's hands. Not everyone has a home theater system, and there are definitely times when it's a bad idea to piss off the neighbors with sudden volume swells. We still want people to have a good experience in these situations as well. DICE created a base mix with a wide dynamic range, but also allowed players to choose an audio setting called "War Tapes." This setting was a more compressed mix configuration and wound up being very popular with the game's core players.

You could do this pretty easily in many games—after all, it should be a simple matter of a final layer of compression on the master mix bus within the game engine. Granted, few things are ever that simple, but it's not a complicated theory. If you've mixed with the minimum/maximum threshold measurement model in mind, then you already know what the perceived dynamic range of your game is. Narrowing



that range in a predictable manner, at the parent level, becomes a lot easier this way. This is a case where targeting a single infinite loudness measurement makes sense.

There is also precedent for this in consumer media devices. When multichannel bitstreams are authored, they frequently have a piece of metadata to indicate an appropriate compression scheme: "Film Standard," "Film Light," and so on. Most DVD and Blu-ray players have a feature built into them that might be referred to as "Night Mode" or something similar. When it is activated, the equipment checks that piece of metadata within the audio bitstream and adjusts its compression settings

appropriately. To my knowledge, DICE is the first developer to emulate this functionality in the game community. If I were to meter "War Tapes" mode with an infinite measurement window, I'd bet that it behaves very similarly to a broadcast television mix.

## METERING IS NOT MIXING

★ There is a way to employ loudness metering in the game industry. Hell, there are probably several. I've laid out a few simple ideas that I think would benefit the game-audio community, but feel free to ignore them. I really only want you to take one thing away from this article: Find what

works for you. It's important to remember that metering, even loudness metering, is merely a tool for mixing. It provides feedback and helps the sound professional predict how a mix will behave on other systems. It should not dictate your artistic choices, but inform them. The broadcast industry has found an application of ITU-R BS.1770 that works for it, and the game industry can too. 🎧

---

**SHAUN FARLEY** is a sound editor and re-recording mixer, and currently holds down the fort at Teleproductions International, working on television and film productions. He is also a contributing editor on [DesigningSound.org](http://DesigningSound.org). Contact him at [shaun@dynamicinterference.com](mailto:shaun@dynamicinterference.com).





# ISAAC



When I started working on THE BINDING OF ISAAC, I was still haunted by the end of SUPER MEAT BOY's development, and the hoops we had to jump through to get there [See *Game Developer* April 2011]. I wouldn't say SUPER MEAT BOY was "selling out," but it was the closest I was going to come to it when it came to playing by the rules to make sure that we could sell the game that consumed two years of our lives (and all of our money). After SMB, I no longer had those worries—I could afford to take a bigger risk and fail, if I felt like failing. I wanted to make something risky and exciting now that the financial aspects of that risk were gone. And I wanted to really push my limits to get back to where I had come from—a place where there were no boundaries, where I could create anything without worrying about making a profit.

THE BINDING OF ISAAC started in a weeklong game jam. Tommy Refenes (SUPER MEAT BOY co-developer) was taking a vacation, so I decided to do the game jam with Florian Himsl, who programmed a few of my previous Flash games (TRIACHNID, COIL, and CUNT). Florian is the kind of guy who is up for anything; he wasn't worried about his reputation, and was basically down with whatever I wanted to do in terms of content. This was good, because I had two clear goals when I started designing ISAAC: I wanted to make a roguelike game using the LEGEND OF ZELDA dungeon structure, and I wanted to make a game about my relationship with religion.

Both goals were challenging but very fun to design, and after seven days we had something that was turning into a game. It seemed too good to pass up, so we continued working on it in Flash (using

ActionScript 2). At this point in the process, I wasn't thinking about how we were going to sell this game (or if we were going to be able to sell the game at all!); it was just a challenge we both wanted to finish.

We finished THE BINDING OF ISAAC after about three months of part-time development. We released it on Steam, and it was selling okay; for the first few weeks, the game was averaging about 100-200 copies a day, eventually stabilizing at about 150 a day after a few months. By this point, the game had already exceeded my expectations, but five months after release something very odd happened. Our daily average started to climb. 200 copies per day turned into 500 copies, then 1,000 copies, and by the seven-month mark ISAAC was averaging sales of more than 1,500 copies a day and climbing. I couldn't explain it—we hadn't put the game on sale or

anything, so I was clueless as to why sales were continuing to grow.

Then I checked out YouTube, and I noticed that fans of the game were uploading Let's Play videos constantly—over 100 videos every day, each getting tons of traffic. ISAAC had found its fanbase, and that base was growing larger and larger. Not bad for a game that was meant to fail!

## what · went · right

### ROGUELIKE DESIGN

The roguelike formula is an amazing design plan that isn't used much, mostly because its traditional designs rely on alienatingly complicated user interfaces. Once you crack the roguelike formula, however, it becomes an increasingly beautiful, deep, and everlasting design that



allows you to generate a seemingly dynamic experience for players, so that each time they play your game they're getting a totally new adventure. I wanted to combine the roguelike formula with some kind of real-time experience, like *SPELUNKY*, but I also wanted to experiment more with the traditional role-playing game aspect of roguelike games *CRAWL* and *DIABLO*. Fortunately, using the basic *LEGEND OF ZELDA* dungeon structure as the game's skeleton made it easy to rework almost all the elements of a traditional roguelike formula (procedurally generated dungeons, permadeath, and so on) into a real-time dungeon crawler format. Almost every aspect of the game seemed to fall perfectly into place with little effort.

Let's start by looking at the *LEGEND OF ZELDA* dungeon and resource structure—it's simple,

and really solid. Keys, bombs, coins, and hearts are dropped in various rooms in the dungeon, and the player needs to collect and use these resources to progress through each level. In *ISAAC*, these elements were randomly distributed and not required to progress, but I included them to add structure to the experience. I also pulled a lot from *ZELDA*'s "leveling structure," where each dungeon would yield an item as well as a container heart to level up the character and give the player a sense of growth; in *ISAAC*, each level contains at least one item, and the player can get one stat-raising item by beating the boss. These items are random, but still designed in a way that made it so your character would have some kind of physical growth as you progress through the game.

I approached the roguelike design from many different directions with *ISAAC*, but at its core,

what made *ISAAC* different than most roguelike games (well, aside from its visuals) was how I dealt with the difficulty curve. Instead of using traditional difficulty settings, I simply made the game adjust to players as they played, adding increasingly difficult content to the game as they progressed. This made *ISAAC* feel longer, richer, and gave it the appearance of a story that writes itself. Using this design also allowed me to reward the player for playing and playing well, with more items that would help aid in their adventures and keep the gameplay fresh and exciting. Once the player finally overcomes Mom, they usually assume the game is over, but instead get a new final chapter, six new bosses, a new final boss, and new items that shuffle into the mix. When the player beats the final chapter, they unlock new playable

characters and items, and when they beat the chapter with each new character, they'll unlock even more content that makes the game even deeper still.

With *ISAAC*, my goal was to create "magic." I wanted players to feel like the game was endless and alive, that the game had a mind of its own and was writing itself as they played. I remember the original *ZELDA* having this feeling of magic and mystery. You weren't sure what things did



LEARN.NETWORK.INSPIRE.

GAME DEVELOPERS CONFERENCE

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

2013

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





until you experimented with them, and you had to brainstorm with your friends and put all your findings together in order to progress. I felt like since I was referencing ZELDA so much in ISAAC's core design, I should also complement it with the feeling of mystery I felt it had back in the day.

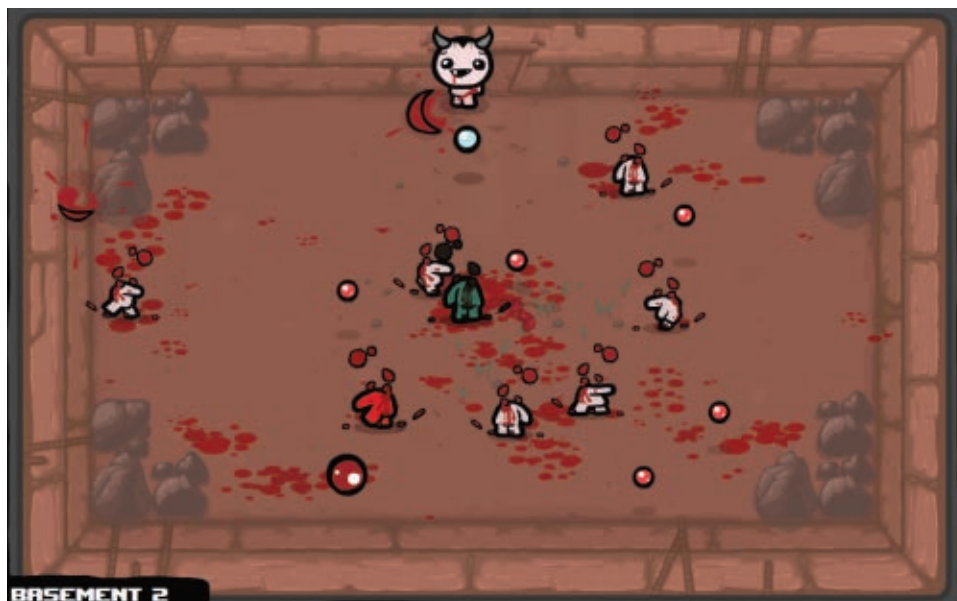


## UNCENSORED UNIQUE THEME

I strongly believe that game enthusiasts want what they haven't seen yet, and that adult gamers should be treated like adults. Some people might argue that ISAAC isn't "mature" when it comes to its content, but those people would be ignorant fools! When I designed ISAAC's story and overall theme, I went in wanting to talk to the player about religion in a manner I was comfortable with—that is, with dark humor and satire.

A lot of the content in ISAAC is extremely dark and adult. It touches on aspects of child abuse, gender identity, infanticide, neglect, suicide, abortion, and how religion might negatively affect a child, which are topics most games would avoid. I wanted to talk about them, and I wanted to talk about them in the way I was comfortable with, so that's what I did with ISAAC. I'm not saying everyone who played ISAAC did so because they cared about these themes, or that they even understood why they were in the game, but I strongly believe that this adult conversation I dove into with ISAAC is what made the game stand out to people and kept them thinking.

I grew up in a religious family. My mom's side is Catholic, and my dad's side is born-again Christian. The Catholic side had this very ritualistic belief system: My grandma could essentially cast spells of safe passage if we went on trips, for example, and we would light candles and pray for loved ones to find their way out of purgatory, and drink and eat the body and blood of our savior to be abolished of mortal sin. As a child growing up with this, I honestly thought it was very neat—very creative and inspiring. It's not hard to look at my work and see that most of the themes of violence actually come from my Catholic upbringing, and in a



lot of ways I loved that aspect of my religion. Sadly, the other side of my family was a bit more harsh in their views on the Bible; I was many times told I was going to hell for playing *Dungeons & Dragons* and *Magic: The Gathering* (in fact, they took my *MtG* cards away from me), and generally condemned me for my sins.

I wanted ISAAC to embody this duality I experienced with religion. I wanted it to show the positive and negative effects it had on me as a child—the self-hate and isolation it instilled in me, but also the dark

creativity it inspired. The Bible is a very good, creatively written book, and one of my favorite aspects of it is how so many people can find different meanings in one passage. I wanted ISAAC to have this in its story as well, which is why the game's final ending(s) have many possible interpretations.



## THE WRATH OF THE LAMB EXPANSION

Doing a DLC expansion was never in the plan for ISAAC; I assumed the

game wasn't going to do well, so it wasn't something we really ever even talked about. I had a few pages of "dream ideas" that I wanted to add to the game, but I had to stop working on them and put them on the back burner, since I wasn't sure how much the extra content would matter if the game didn't do well. Six months later, we ended up taking these dream ideas and expanding them into an extra-large DLC expansion.

The WRATH OF THE LAMB expansion added over 80 percent more content to an already-

# ISAAC



bloated experience—and people ate it up. 25 percent of the people who purchased ISAAC also paid for the expansion, and that ratio is going up by the day. We honestly didn't expect to make it, but once I started seeing such a positive, creative fan response, I felt obligated to continue Isaac's adventure.

Honestly, however, the number-one reason why I did the expansion was because my wife Danielle had already 100-percent-completed the game. It was the first game I had designed that she became obsessed with (she's actually playing it right now, behind me, while I'm writing this), and it made me extremely happy to see her fall in love with something I had made. I just had to continue it (also, she wouldn't shut up about wanting more).

## **C**IRCUMVENTING CENSORSHIP WITH STEAM

Steam is amazing, and with the ISAAC release experience I've found another crucial reason why: You can use it to sell uncensored and unrated games. This was vital with ISAAC, because I wasn't going to bother with getting an ESRB rating for a game I wasn't sure was going to sell more than 100 copies. Valve knew the game was weird and could possibly get some backlash, but they allowed it on Steam because they felt it had potential, and I love them for that.

Another huge plus to working on Steam was the ability to constantly update ISAAC with fixes, updates, and new content. They would upload a new build within the day we submitted it to them, and if we had released it on any other platform this would have been impossible (and probably cost us about \$40,000).

## **S**ISAAC'S FANS

The main reason why you've heard about ISAAC is its fans. Releasing SUPER MEAT BOY and being in *Indie Game: The Movie* has shown me a wide range of fan types, but ISAAC fans are just in a league of their own. At the time of this writing, there are well over 30,000 videos of ISAAC on

YouTube, countless pieces of fan art, animations, and plush toys all over the Internet, and over 30 fictional fan blogs where people can ask characters in ISAAC questions and get in-character responses. It's totally surreal. Something in ISAAC just spoke to a large group of creative people, and they held him up and ran with him.

Recently, I've been trying to find out how ISAAC attracted such a creative and dedicated fanbase. What is it about the game that spoke to this large group of artistic men and women? I can't ever know for sure, but I strongly believe that something in ISAAC'S theme and story connects to a large number of "creative outcasts." I made it from the standpoint of a creative outcast; the game is about a creative child who is looked at as "made wrong" by the one person who cares about him, and his only real escape is his imagination. This is a story I could relate to, and it's one I think a lot of creative people latched on to mostly because it's not really a story you see in video games at all.

I am forever in debt to these people. Not only did they get the game to the masses, they also inspired me so much. You guys make me want to continue designing this game forever.

what · went · wrong

## **—** SHAKY LAUNCHES

THE BINDING OF ISAAC was updated every day for two weeks during launch, and each time we thought we had solved all the issues. (Each time we were wrong.) Luckily, we were able to remove all game-breaking bugs in the first two days, but there were still many smaller bugs left that gnawed at us for a long time. It sucked to launch with so many issues—we had save bugs, game-breaking bugs that wouldn't let you complete the game, bugs that would not reward unlocks and achievements, and even some really odd ones that would scramble item clips and cycle through art from the game constantly. It wasn't pretty, and it was even more painful to watch



so many upset players posting in the forums about the many issues with the game. (The biggest question, of course, was “Why didn’t you test the game?”)

The reason we released ISAAC when we did was because it was done (if untested), and I didn’t want to waste any more of my time on something I expected would crash and burn. I was just so worried it would suck that I wanted to get it out and over with.



## TESTING (AND THE LACK THEREOF)

At launch, THE BINDING OF ISAAC had 100 items and five playable characters. 70 percent of the items in ISAAC stack, and all the item abilities will affect Isaac in some way, so there were so many variables to keep track of that all the testing in the world couldn’t have prepared us for launch. Everything about the game was based on complex variables that multiply with each level you pass. In order to fully test all the variables we had in place, it would have taken hundreds of testers several days of extensive play time to fully debug this little monster—there were bugs that actually took 100,000+ people four weeks to find due to how buried and rare some of them were. Also, launching on PC meant launching on 10,000 different PC configurations, so we had bugs that would be caused by antivirus software, clean-up tools, and even some types of keyboard configurations.

The sad fact was that it was the day-one buyers that ended up fully testing ISAAC for us, and I felt really shitty about that. A few weeks after launch I put together a free mini-expansion to make up for our shaky launch—but that, too, was flaked with bugs.



## PERFORMANCE AND FEATURE ISSUES WITH ACTIONSCRIPT 2

The biggest downfall of THE BINDING OF ISAAC is its performance. ISAAC was designed in Flash using ActionScript 2; that’s what Florian could program in, so those were the limitations we had to work

around. Sadly, Flash AS2 is quite outdated, and even with all the amazing work Florian put in, we simply couldn’t get the game to run well on lower-end PCs. Flash even had major issues with PCs that used dual-core processors, so even PCs with amazing specs would slow down at times.

If I had known that anyone would have cared about ISAAC, I wouldn’t have made it in Flash at all. Framerate issues aside, Flash’s lack of controller support and integrated Steam features really hurt ISAAC. It pained me to release a game that was lacking features almost all games have. You’d think by now Flash would have added some kind of controller support, but no. Tommy actually wrote an achievement program specifically for ISAAC so it could award Steam achievements, which was hugely helpful, but I couldn’t ever really feel satisfied with the product due to our AS2 limitations.



## TOO BIG IN SCOPE FOR FLASH

Aside from the performance issues and AS2’s limitations, late in ISAAC’S development we soon realized that Flash simply wasn’t at all made to support a game of ISAAC’S size. Once the .FLA file rose above 300MB, we couldn’t even consistently generate an .SWF file from it without crashing.

This issue almost prevented WRATH OF THE LAMB from coming out at all; we were at a point in development where simply saving the .FLA would corrupt it about 25 percent of the time. Florian would have to restart his PC and save the .FLA in a new folder every time we had to export an .SWF just to test it, and 50 percent of the time it wouldn’t work for no apparent reason. It was quite a horrible experience, and if we could have seen into the future with a crystal ball, we would have simply not used Flash. (Maybe this will be a feature in Flash CS7....)



## BLASPHEMY AND CONTROVERSY

Not surprisingly, controversy made a few appearances in ISAAC’S release year, but not in the way

you might think. During ISAAC’S German retail launch, the German ratings board gave ISAAC a 16+ due to “blasphemy.” That itself didn’t cause controversy—instead, it was the idea that said blasphemy could affect the age rating on a video game. Blasphemy isn’t something you can define for everyone (what’s blasphemous for one religion isn’t necessarily so for another), so how could one define something as containing blasphemy? It was a very interesting argument, and I’d be lying if I said that having the first game rated 16+ due to blasphemy didn’t feel awesome, but sadly it was this controversy that I believe eventually led to Nintendo’s decision not to port ISAAC to the 3DS.

I remember my wife being worried about ISAAC’S release, worried that it might offend the wrong people and someone could end up being hurt. I can’t say I didn’t have some hesitation about this aspect of talking about religion in a satirical and possibly blasphemous way, but I couldn’t help but avoid the simple logic that, well, most of those kind of people don’t play games. And after over a year, I really believe that’s true. (Thank God!)

## ISAAC REBORN

As of writing this postmortem, THE BINDING OF ISAAC has sold over one million units on PC and Mac in its first year on Steam, one-quarter of the people who own the main game paid for the WRATH OF THE LAMB expansion, and the interest seems to continue building.

A few months ago I was contacted by Tyrone of Nicklas (CAVE STORY, VVVVVV) and asked about how I felt about remaking THE BINDING OF ISAAC for consoles. I love consoles as much as the next guy, but dealing with the business end of console development wasn’t something I wanted to dive back into at this point. I told them yes, but I had a few strict guidelines to make sure an ISAAC remake was perfect. I wanted the game to feature the second planned expansion that I couldn’t do in the

Flash version, I wanted it to feature local co-op, I wanted the graphics to be totally remade in 16-bit but still look and feel like the Flash version, and finally, I didn’t want to deal with anything when it came to business. Nicklas has agreed to these terms, and development has started on THE BINDING OF ISAAC: REBIRTH.

It’s still too early to tell for sure what consoles the game will end up on, but both Microsoft and Sony feel like it would be a perfect fit for their digital platforms, and we have a feeling the new look might soften up a few people at Nintendo for a possible Wii U/3DSWare release. I’m wary about how the game might control on iPad, but if they can make it work, I’m all for it.

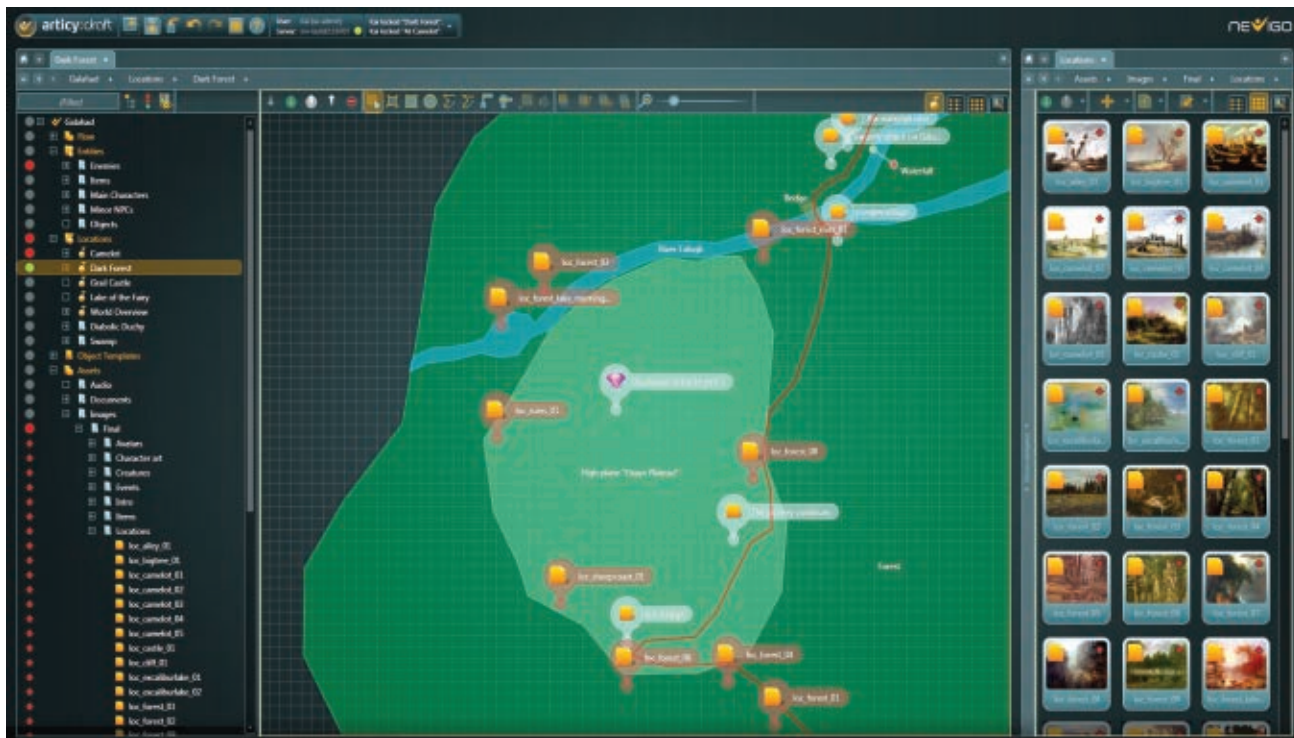
THE BINDING OF ISAAC was a huge personal achievement. I was able to talk about something personal and meaningful in a way I felt comfortable with, and I was able to get my feet in the water with the roguelike formula and random generation.

When I started on ISAAC, my goal was to make a niche cult classic, something with a tiny but die-hard fanbase. What I didn’t expect was how large a “tiny” niche audience would actually be.

But what moved me the most is the amount of creativity ISAAC inspired in others. Every day I read fiction blogs, watch YouTube animations, and look at others’ illustrations while thinking about how honored I am to have made something that could have helped motivate so many to create. The three months Florian and I put into creating THE BINDING OF ISAAC didn’t just pay off with a financial windfall—it also gave us an eye-opening experience that proves to me without a doubt that people truly want and respect games that are uncensored and risky, and that ask more of the player than most games these days do. 

**EDMUND MCMILLEN** is an independent game designer known for his work on SUPER MEAT BOY, GISH, and THE BINDING OF ISAAC. He is based out of Santa Cruz, CA. You can find his personal blog at [edmundm.com](http://edmundm.com).





Sketching locations in articy:draft.

## Neviso GmbH

# ARTICY:DRAFT

BY TOBIAS HEUSSNER

Writers, game designers, and narrative designers have been on the quest for the perfect creation tool for years. Excel has been the best choice in most cases (other than in-house solutions, anyway), but now Germany-based Neviso claims to have developed the first professional story and game design tool. This tool allows you to compose nonlinear storylines and dialogues, create collections of entities such as characters and areas, display flowcharts, and define custom templates.

### BETTER THAN EXCEL?

» Almost all game professionals have tools they're unsatisfied with: Programmers have their IDEs and debuggers; artists have Maya, 3ds Max, and Photoshop; and level designers have tools like UnrealEd. Writers have Excel—which is hardly a great writing tool because the amount of information needed for a compelling story slows it down so much that it becomes almost unusable. Excel also doesn't have good tools to lay out story flows or

to display nonlinear content. The only reasons I know of that writers use Excel is so information can easily be put into columns, and so they can create material that can easily be used for voice recording and communicating with the development team.

But unless you're working for a developer with a better in-house tool, Excel alternatives are rare—which means that a tool like Neviso's articy:draft could be very compelling. But is it right for your team's workflow?

### ARTICY:DRAFT 101

» Think of articy:draft as a repository for data where you can store your ideas, plots, characters, and whatever else you can imagine, and then organize it according to your needs and to the project you're developing.

Articy:draft has seven main workspaces, and you can view these sections side-by-side by using multiple windows to show different parts of the project. All you need to do is move to one of the

edges of your workspace and drag it to the side to open another work area. You can also display different work areas on different monitors.

The first section is the flow chart, where you can lay out all your storylines, dialogues, and anything else that is best designed as a graph. Flow elements can contain other flow elements, and they can be combined with the template feature, but they don't have to be connected with each other in case you need to build parallel storylines or feature designs.

The second section is the entity collection, which lists all of your game's entities: characters, items, or even abstract things like quest information blocks or interactive objects. You can add

more specific information about these entities in the template editor (more on that later).

The third section covers the locations for your game and story. You can start with rough sketches, populate those sketches with objects from the entity section, and gradually add more detail to your locations over time. Place an entity on your map, and you can immediately examine and edit that entity's properties by double-clicking on it. Also, the entity will contain a link to the areas it is placed in. These two features make it much easier to avoid duplicating information and keep everything in sync.

The fourth section is a simple notepad, which has some basic dialogue and cutscene layout features usually found in screenwriting programs, which a lot of writers find useful for brainstorming. It is also a good place for your meeting notes and minutes, since it makes sense to store them in the same place as the rest of your design material. Another great feature of the note section is that you can directly import Final Draft documents into a note, mark the parts you'd like to use, and drag those parts into the flow chart section to automatically create a flow chart based on this information. (Articy:draft will automatically link character tags to the correct entity if there is one; otherwise it'll show you a warning.)

The fifth section is the template feature, which allows you to create the templates you need for your project. So far articy:draft includes only a few very basic ready-for-use templates, but since most writers and designers prefer to use their own templates, it's probably not a major problem.


The sixth section, Journeys, is articy:draft's newest addition; it became part of the tool with patch 1.3. Journeys are recorded versions of walkthroughs through your flow charts. These walkthroughs can be created directly from within the flow chart section, and display similarly to a PowerPoint presentation. Note that articy:draft does not automatically evaluate any game conditions linked to your

choice points this feature is simply to present your game flow and the results of different in-game choices while designing your narrative or explaining it to other team members.

The last section is the asset collection, which stores all your reference material so you can use it within the other sections. Assets could be anything—artwork, sketches, screenshots, videos, documents, sound files, and so on.

### ARTICY, MEET THE TEAM

» As you may have deduced, articy:draft is a fairly complex program, and it can be overwhelming at first. I recommend having a look at the tutorial videos on the Nevigo web site ([www.nevigo.com/products/articydraft/media.html](http://www.nevigo.com/products/articydraft/media.html)) to see its features yourself, because the program doesn't make it immediately obvious how you're supposed to use it. You can also download a sample project, which can clue you in to what a possible workflow could look like, but it's not a full tutorial, so you may still need to experiment a bit before you figure out the right way to fit it into your workflow. Once you get used to it, though, you'll be defining NPCs in one place and maps in another, and easily using that information in story graphs. Get used to the way articy:draft uses references as its core organizing principle and you'll see how useful it can be and how much faster it is to work in compared to Excel.



**NEVIGO GMBH Articy:draft**  
[www.nevigo.com](http://www.nevigo.com)

**PRICE**

- > Single-user application: \$450

**SYSTEM REQUIREMENTS**

- > Windows XP/Vista/7 (32- or 64-bit)

**PROS**

- 1 Entity system makes it easy to keep track of story/game elements
- 2 Highly customizable
- 3 Import function for Final Draft

**CONS**

- 1 Slight learning curve
- 2 No export function for templates
- 3 Sometimes a bit slow when creating complex templates

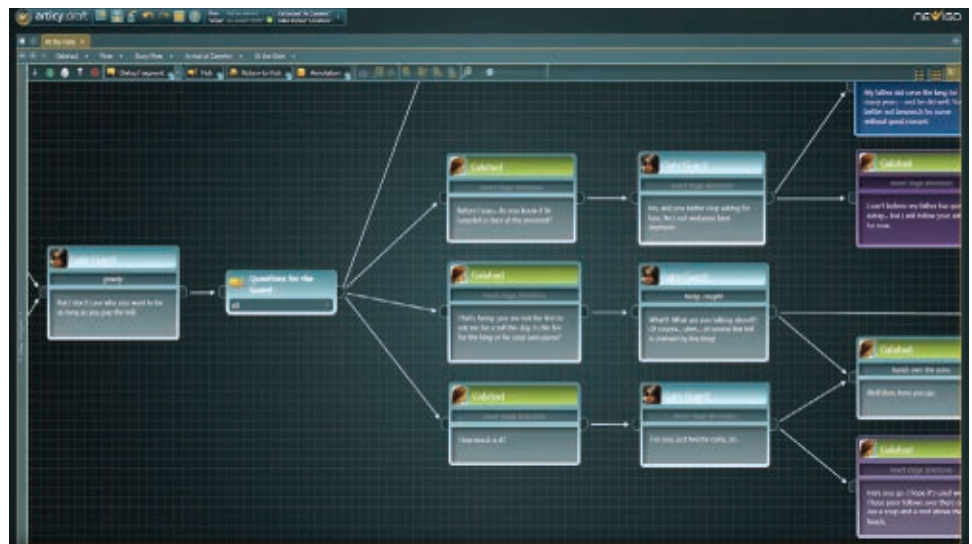
Of course, articy:draft is at its most useful if you have as many licenses as you have developers involved in your game, which can get expensive. Fortunately, Nevigo offers a cheaper viewer license that enables staff members who only need read access to view the project without spending the money for a full client. Also, articy:draft readily exports to Microsoft Word and Excel, though if you start working with Word docs or Excel sheets you'll need to make sure that they're up-to-date when other people start working on them. Note that the

export-to-Word function is a fairly new addition to articy:draft. It works okay, but it still has some minor issues with the layout of the different fields, so the Word file may not look exactly like your template in articy:draft.

### FLOW CHARTS, ENTITIES, AREAS

» As I said before, everything in articy:draft is built upon the concept of references—every object exists only once in your articy:draft project and then is referenced to every place where it is needed. For example, if you put a character in a relationship with another character, those two are linked so that you can jump back and forth between them. Characters are similarly linked when you add them to an area, which makes it easier to avoid duplicating information (thus reducing story glitches).

The flow chart feature is useful for story flow and dialogue trees, but it can also come in handy for more than that; game designers can use this feature to create skill trees or menu flows, and level designers can use it to lay out level events and connect them to maps. You could even use the flow charts to design complex state machines, which could help communicate features with the programming department. (The flow chart features don't seem



Writing a branching conversation.



to have any notable advantages over other such tools, mind you the main reason you'd do it in articy:draft is simply to have all the content development stuff in the same place.)

With the area feature, you can sketch out an area with vector-based tools (and include more detailed background pictures). However, it's mostly useful because you can link your other elements into the map and easily reference their information—place an NPC guard next to a guard house, and articy:draft will place a link to your guard. If you want to play a certain event at a certain spot in your map, you can drag and drop a flow chart element to this spot and link it with the map. This is extremely helpful when you review map concepts or when you need to talk about the narrative space and the choices a player will have in a certain map. The fact

that everything is referenced is helpful too, because during the review process you don't need to open other documents or programs; you can jump to the needed information whenever you want and this information can never be outdated, because if any other developer updates any of the references, these changes will also be visible in your map.

### TEMPLATES: DIY ONLY

» Whether you find articy:draft useful depends largely on how well you design your templates. Articy:draft has some great tools to create new templates, but it doesn't include many sample templates, so you have to experiment a bit to find the best way to lay out and structure them. It took me some time to find a good workflow and figure out what to put where, but it got pretty easy once I figured out what worked for me. Be careful—

once you update a template, it will also update all entities that use this specific template, which means you have to be careful not to lose any information. (On the plus side, you'll always be up-to-date.) Articy:draft will warn you about the consequences of changing templates, which helps.

As of this writing, articy:draft doesn't allow you export your templates and use them in another project, which is one of the main problems I see, because normally I don't want to create the same templates over and over again. You could create an empty project and copy it to create a new one, but it should be easier than that. Nevigo has promised to work on a solution, and hopefully soon it will be possible to move templates from one project to another. Other than that, the template feature works fairly well, though it can get a bit slow if the templates get highly complex.

### HAPPILY EVER AFTER

» All in all, articy:draft is a great tool for interactive story and game design. It unites all the different parts that go into good story and game design in one tool, and though it has a steep initial learning curve, it gets pretty easy to use. For the first hour that you're using it, you might not be convinced to switch from Excel, but after that hour you'll find yourself developing your story faster and communicating that story to your teammates faster than before. 🎮

---

**TOBIAS HEUSSNER** is a game content/narrative designer, currently working as a senior level designer on online games. His areas of expertise are game content design and game system design, and he has been involved in professional game development in design and management for over 14 years. You can find more information on his personal web site at [www.theussner.com](http://www.theussner.com).

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

**gamasutra.com**

the art and business of making games





# 10 TIPS FOR CLEANING BAD CODE

## FOLLOW OUR 10-STEP PROGRAM FOR REHABILITATING CRAPPY CODE



Guess what! You've just inherited a stinking, steaming pile of messy old code. Congratulations, it's all yours.

Bad code can come from all kinds of places: middleware, the Internet, or even your own company.

You know that nice guy in the corner who nobody had time to check up on? Guess what he was doing all that time. Yep, he was churning out bad code. Do you remember that module someone wrote years ago, just before she left the company? Twenty different people have since hacked, patched, and "fixed" a few bugs in that module without really understanding what they were

doing. And then there was that open-source thing you downloaded that you knew was horrible, but it solved a very specific and quite hairy problem that would have taken you ages to do by yourself.

Bad code doesn't have to be a problem, as long as it's not misbehaving (and nobody pokes their nose in it). Unfortunately, that state of ignorant bliss rarely lasts. Someone will find a bug, or request a feature, or release a new platform that breaks your bad code, and now you have to dig into that horrible mess and try to clean it up. I know—it's no fun for anyone, so here are 10 tips that will help you save your code (and your sanity).



## STEP 0

### IS THIS WORTH FIXING?

The first thing you need to ask yourself is whether the code is worth cleaning. I'm of the opinion that when it comes to code cleaning, you should either karate do "yes," or karate do "no," Daniel-san. If you decide to fix it, assume full responsibility for the code and rework it until you end up with something that you are actually happy to maintain and proud to have in your code base.

The other option is to decide that even though the code looks horrible, it isn't cost effective to take time out of your busy schedule to fix it. So instead you just do the smallest change possible that solves your current problem.

In other words, you either regard the code as *yours* or *theirs*.

There are merits to both alternatives. Good programmers get an itch when they see bad code. They bring out their torches and pitchforks and chant: "Unclean! Unclean!" That is a good instinct, but cleaning code is also a lot of work. It is easy to underestimate the time it takes. It can be nearly as time consuming as writing the whole thing from scratch, and it doesn't bring any short-

Stereotypes aside, there are merits to both arguments. Believe it or not, but sometimes the managers are right. I've seen many cases where an ambitious rewrite goes over time and over budget and ends up delivering a product that while better from a programmer's perspective actually ends up being less useful to the end users.

Both programmers and managers should listen to the arguments from the other side. Managers should realize that unchecked code rot leads to a culture of sloppiness, abandonment, and despair, which is the death of a development team. Programmers should realize that their time and capacity is finite, and they can't make every piece of code perfect.

Still, the choice is never easy.



## STEP 1

### GET A TEST CASE

Seriously cleaning a piece of code means messing around with it a lot. You will break things.

If you have a decent test case with good coverage, you will immediately know what has broken and you can usually quite quickly figure out what stupid mistake you just made, which will save you a ridiculous amount of time and

manual one. Fire up a game level and run the character through a specific set of actions related to the code you are cleaning, and check that the behavior doesn't change.

Since such tests are more time consuming, it might not make sense to run them after every change you make, which would be ideal. But since you're going to put every single change you make into source control (more on this later), all is not lost. Run the test every once in a while (maybe every five changes or so). When it discovers a problem, you can do a binary search of those last few commits to find out which one caused the problem.

If you at any time discover an issue that wasn't detected by your test, make sure that you add that to the test, so that you capture it in the future.



## STEP 2

### USE SOURCE CONTROL

Do people still have to be told to use source control? I sure hope not.

Source control is absolutely crucial for cleaning work, because you will be making lots and lots of small changes to the code. When something breaks, you want to be able to look back in the revision history and find out where it broke.

download mercurial (or git), create a new repository, and put the code that you checked out of your company's stupid system there. Do your changes in that repository, committing as you go. When you are done you can merge everything back into the stupid system.

Cloning the repository into a sensible source-control system only takes a few minutes, and it's absolutely worth it. If you don't know mercurial, spend an hour to learn it. You will be happy you did. Or if you prefer, spend 30 hours to learn git instead. (I kid! Not really. Git fanatics, you can send me your hate mail now.)



## STEP 3

### USE THE COMPILER

For languages such as Java and C#, there are lots of tools that can help with typical cleaning tasks—renaming methods and variables, for example.

In C++, you have to do most of the heavy lifting yourself, but you can get some help from the compiler. For example, if you rename a method, you will get compile errors everywhere it was used.

You can use this to your advantage by deliberately introducing a compile error. For example, suppose there is this method:

```
set_pos(float y, float x);
```

Say that you want to change the order of the x and y parameters so that it is consistent with other methods. Instead of doing a global search (which may find lots of other methods named set\_pos), just rename the method to:

```
set_pos_swapped(float x, float y);
```

Now all the old calls will give compile errors. As you fix those compile errors, you swap the parameters, and when you are done, you can rename the method back to set\_pos.



## STEP 4

### MAKE ONE (SMALL) CHANGE AT A TIME

There are two ways of improving bad code: revolution and reform. The revolution method is to nuke



**Managers just want quick fixes and don't value the work required to maintain and improve code. Programmers just want to spend their time on highbrow rewriting projects that make the code beautiful, but don't add any value to the company.**

term benefits—two weeks spent cleaning code won't add any new features to your game, but might give you some new bugs.

On the other hand, the long-term effects of never cleaning your code can be devastating. Entropy is the code-killer.

Often this gets expressed as a conflict between *managers* and *programmers*. *Managers* just want quick fixes and don't value the work required to maintain and improve code. *Programmers* just want to spend their time on highbrow rewriting projects that make the code beautiful, but don't add any value to the company.

anxiety. Get a test case. It's the first thing you should do.

Unit tests are best, but not all code is amenable to unit testing. Sometimes the amount of hoops you would have to jump through to make something testable is just insanely impractical. (Test fanatics, you can send me your hate mail now.)

The next best things are automated integration tests—scripted tests that produce a predictable output. Check both the test and the output into source control, so that you can see if it changes and how.

Any test is better than nothing. If you can't do an automated test, do a

Also, if you are anything like me, you will sometimes start down a refactoring path (like removing a stupid class) and realize after a while that it wasn't such a good idea, or that it was a good idea but everything would be a lot simpler if you did something else first. In that case, you'll want to be able to quickly revert everything you just did and begin anew.

Your company should have a source-control system in place that allows you to do these changes in a separate branch and commit as much as you like without disturbing anybody else. If it doesn't, you should still use source control—





Before you commit to fixing the code, ask yourself these questions:

**Q** *How many changes do you expect to make to the code?*

**A** Is it just this one small bug that you need to fix, or is this code that you expect to return to many times to tweak, tune, and add new features? If it's just this one bug, then perhaps it is best to let sleeping dogs lie. However, if this is a module that you will need to mess around with a lot, then spending some time to clean it up now will save a lot of headaches later.

**Q** *Will you need/want to import upstream changes?*

**A** Is this an open-source project that is under active development? If so, and you want to pull the changes made upstream, you can't make any big changes to the code or you will be in merge hell every time you pull. So just be a nice team player, accept its idiosyncrasies, and send patches with your bug fixes to the maintainer.

**Q** *How much work is it?*

**A** How many lines of code can you realistically clean in a day? An order of magnitude estimate says more than 100 and less than 10,000, so let's say 1,000. So if the module has 30,000 lines, you might be looking at a month of work. Can you spend that? Is it worth it?

**Q** *Is it a part of your core functionality?*

**A** If the module does something peripheral, such as rendering fonts or loading images, you might not care that it is messy. You might swap out the whole thing for something else in the future—who knows? But you should own the code that relates to your core competence.

**Q** *How bad is it?*

**A** If the code is just slightly bad, then perhaps you can live with it. If it is mind-numbingly, frustratingly, incomprehensibly bad, then perhaps something needs to be done.

**Q** *Do you have to clean all of it?*

**A** Perhaps you can clean just the high-level interface and leave the low-level code as it is. Dividing the cleaning task into smaller, simpler steps makes it more manageable. It also provides useful checkpoints where you (and your manager) can ascertain that the cleaning progresses as planned, and abort if there are problems.

everything from orbit and rewrite it from scratch. The reform method is to refactor the code with one small change at a time without ever breaking it.

This article is about the reform method. I'm not saying that revolutions are never necessary. Sometimes things are so terrible that you have no other option. But people who get frustrated with the slow pace of reform and advocate revolution often fail to realize the full complexity of the problem, and thus don't give the existing system enough credit.

When you do a full rewrite, you often miss some of the nuances, special features, and bug fixes in the original code, and you end up with a system that is *less useful* than the original one. Then, over the next few months, you have to add those features back one by one. In the end, your new system might be just as messy as the one it was intended to replace—and the Code Phoenix's cycle of destruction and renewal can continue.

The *reform* approach is a more ecologically sustainable method of code development. When you reform, you know at each step what changes you make. You can still decide to throw out a little-used feature in order to simplify the code, but then at least you know that's what you are doing. You can look at each single change you make and see that it is good. This doesn't mean that you can't do drastic changes to the code, just that you have to do them one step at a time.

relevant read:

Stack Exchange co-founder Joel Spolsky has written a classic article about this called *Things You Should Never Do*. Look it up here: [www.joelonsoftware.com/articles/fog0000000069.html](http://www.joelonsoftware.com/articles/fog0000000069.html)

The best way of reforming code is to make one minimal change at a time, test it, and commit it. When the change is small, it is easier to understand its consequences and make sure that it doesn't affect your existing functionality. If something goes wrong, you only

have a small amount of code that you need to check. If you start doing a change and realize that it is bad, you won't lose much work by reverting to the last commit. If you notice after a while that something has gone subtly wrong, a binary search in the revision history will let you find the small change that introduced the problem.

It is a common mistake to try to do more than one thing at the same time. For example, while you're getting rid of an unnecessary level of inheritance, you might notice that the API methods are not as orthogonal as you would like them to be and start to rearrange them. **Don't do this!** Get rid of the inheritance first, commit that, and then fix the API.

At first, this often feels like a slower way to work, but it is actually a lot faster. Smart programmers organize the way they work so that they don't have to be that smart.

Try to find a path that takes you from what the code is now to what you want it to be in a sequence of small steps. For example, in one step you might rename the methods to give them more sane names. In the next, you might change some member variables to function parameters. Then you reorder some algorithms so that they are clearer. And so on. If you start doing a change and realize that it was a bigger change than you originally thought, don't be afraid to revert and find a way of doing the same thing in smaller, simpler steps.



## STEP 5

### DON'T CLEAN AND FIX AT THE SAME TIME

This is a corollary to #4, but important enough to get its own point.

It is a common problem. You start to look at a module because you want to add some new functionality. Then you notice that the code is really badly organized, so you start reorganizing it at the same time as you are adding the new functionality.

The problem with this is that cleaning and fixing have diametrically opposite goals. When you clean, you want to make the



code look better without changing its functionality. When you fix, you want to change its functionality to something better. If you clean and fix at the same time, it becomes very hard to make sure that your cleaning didn't inadvertently change something.

Do the cleaning first. *Then*, when you have a nice clean base to work with, add the new functionality.



## STEP 6

### REMOVE ANY FUNCTIONALITY THAT YOU ARE NOT USING

The time it takes to clean is proportional to the amount of code, its complexity, and its messiness. If there is any functionality in the code that you are currently not using and don't plan to be using in the foreseeable future, get rid of it. That will both reduce the amount and complexity of the code you will have to go through (by getting rid of unnecessary concepts and dependencies). You will be able to clean faster, and the end result will be simpler.

Don't save code because "who knows, you might need it someday." Code is costly. It needs to be ported, bug-checked, read, and understood. The less code you have, the better. In the unlikely event that you do need the old code, you can always find it in the source repository.



## STEP 7

### DELETE MOST OF THE COMMENTS

Bad code rarely has good comments. Instead, they are often:

```
// Pointless:
// Set x to 3 x = 3;
// Incomprehensible:
// Fix for CB (aug) pos +=
vector3(0, -0.007, 0);
// Sowing fear and doubt:
// Really we shouldn't be doing
this t = get_latest_time();
// Downright lying:
// p cannot be NULL here
p->set_speed(0.7);
```

Read through the code. If a comment doesn't make sense to you and doesn't further your understanding of the code, get

rid of it. Otherwise you will just waste mental energy on trying to understand that comment each time you read the code in the future. The same goes for dead code that has been commented or #ifdef'ed out. Get rid of it. It's there in the source repository if you need it.

Even when comments are correct and useful, remember that you will be doing a lot of refactoring of the code. The comments may no longer be correct when you are done. And there is no unit test in the world that can tell you if you have broken the comments.

Good code needs few comments because the code itself is clearly written and easy to understand. Variables with good names do not need comments explaining their purpose. Functions with clear inputs and outputs and no special cases or gotchas require little explanation. Simple, well-written algorithms can be understood without comments. Asserts document expectations and preconditions. Comments should only be used when the code is not obvious to an experienced professional. And whenever possible, it is better to rewrite the code so that it is obvious than to add a comment.



!! When you have cleaned the code, how do you keep it clean? I don't think that is completely possible. Rather, you should regard code cleaning as a continuous activity, just like weeding a garden—but there are some things you can do to minimize the problem. !!

In many cases, the best thing to do is just to get rid of all old comments, focus on making the code clear and readable, and then add back whatever comments are needed—comments that reflect the new API and your own understanding of the code.



## STEP 8

### GET RID OF SHARED MUTABLE STATE

Shared mutable state is the single biggest problem when it comes to understanding code, because it allows for spooky "action at a distance," where one piece of code

changes how a completely different piece of code behaves. People often say that multithreading is difficult, but really, it is the fact that the threads share mutable state that is the problem. If you get rid of that, multithreading is not so complex.

If your goal is to write high-performance software, you won't be able to get rid of all mutable state, but your code can still benefit enormously from reducing it as much as possible. Strive for programs that are "almost functional," and make sure you know exactly what state you are mutating where and why. Shared mutable state can come in different shapes:

#### Global variables.

The classic example. By now everybody surely knows that global variables are bad. But note (and this is a distinction that people sometimes fail to make), that it is only shared mutable state that is problematic. Global constants are not bad.  $\infty$  is not bad. `Printf` is not bad.

#### Objects: Big bags of fun.

Objects are a way for a large number of functions (the methods) to implicitly share

Broken programmers talk about them in dusky bars, their sanity shattered by their encounters: "I just kept scrolling and scrolling. I couldn't believe my eyes. It was 12,000 lines long." When functions are big enough, their local variables are almost as bad as global variables. It becomes impossible to tell how changing a local variable might affect a chunk of code 2,000 lines down.

#### Reference and pointer parameters.

Reference and pointer parameters that are passed without `const` can be used to subtly share mutable state between the caller, the callee, and anyone else who might be passed the same pointer.



## STEP 9

### GET RID OF BAD CONCEPTS

Bad code doesn't have to come from bad coding. It can also come from bad thinking.

Sometimes, the programmer just doesn't have a clear picture of the problems the code needs to solve, and ends up coding around basic concepts that are off-kilter, with a lot

of extra code that just compensates for that fundamental wrongness. For example, a horrible system could use strings to represent dates and have a lot of weird code for dealing with the fact that "2012-10-01," "2012-10-1," and "1/10 12" all refer to the same date. Often, the problem stems from a single concept that has been given multiple responsibilities that don't quite gel. For example, consider the date struct:

```
struct Date { int year;
int month; int day;
bool is_repeating; int repeat_
interval; };
```

#### Megafunctions.

You have heard about them—mythic creatures that dwell in the deepest recesses of the darkest code bases.

Here are some practical ideas for getting rid of shared mutable state:

- » Split big functions into smaller ones.
- » Split big objects into smaller ones by grouping members that belong together.
- » Make members private.
- » Change methods to be *const*, and return the result instead of mutating state.
- » Change methods to be *static*, and take their arguments as parameters instead of reading them from shared state.
- » Get rid of objects entirely, and implement the functionality as pure functions without side effects.
- » Make local variables *const*.
- » Change pointer and reference arguments to *const*.

Some practical ideas for cleaning out unnecessary complexity:

- » Remove the functionality you are not using (as suggested above).
- » Simplify necessary concepts, and get rid of unneeded ones.
- » Remove unnecessary abstractions, and replace them with concrete implementations.
- » Remove unnecessary virtualization and simplify object hierarchies.
- » If only one setting is ever used, get rid of the possibility of running the module in other configurations.

Here the simple concept of a date has been muddled together with the completely different concept of a repeating calendar event. The result is a complete mess.

Finding bad concepts is not always as straightforward as in these examples. Often, you need to work with the code a lot to really understand what it does—and what problems it tries to solve. Only then can you see that some of its fundamental concepts are not very well thought out. Sometimes it only manifests itself as a nagging feeling that something is wrong. I've found that discussing these things with other programmers is a good way of clarifying thought and finding where the real problem lies.

When you have identified a bad concept, you want to replace it with something better. Again, don't try to do that all at once as a single monolithic change. Instead, start by using the new concepts in the low-level code. Let the high-level code continue to use the old concepts, and write helper code to translate between the old and new representations. Then piece by piece, move more and more code over to the new concepts.



## STEP 10

### GET RID OF UNNECESSARY COMPLEXITY

Unnecessary complexity is often a result of overengineering. The

code's support structures (for serialization, reference counting, virtualized interfaces, abstract factories, visitors, and so on) dwarf the code that performs the actual functionality.

Sometimes overengineering occurs because software projects start out with a lot more ambitious goals than what actually gets implemented. More often, I think, it reflects the ambitions/aesthetics of a programmer who has read books on design patterns and the waterfall model, and believes that overengineering makes a product solid and high-quality.

This kind of thinking often leads to a heavy, rigid, overly complex model that can't adapt to feature requests the original designer didn't anticipate. Those features are then implemented as hacks, bolt-ons, and backdoors on top of the ivory tower, resulting in a schizophrenic mix of absolute order and utter chaos.

The cure against overengineering is YAGNI—You Ain't Gonna Need It! Only build the things that you *know* you need. Add more complicated stuff when you need it, not before.

just like weeding a garden—but there are some things you can do to minimize the problem:

*Encourage a culture of responsibility and professional pride.*

*Maintain a living discussion about design issues. Make sure it stays focused on the big picture, and doesn't get bogged down in pointless debate about trivialities.*

*Remember there is no such thing as throwaway code. Code always lives longer than you expect.*

*Keep the code simple, decoupled, minimal, and isolated.*

*Avoid the temptation of complex, heavy, coupled abstract systems.*

*And don't forget to minimize shared mutable state.*

That is all. Now, go forth, my minions, and clean that code! 🧹

### A CLEANER FUTURE

» When you have cleaned the code, how do you keep it clean? I don't think that is completely possible. Rather, you should regard code cleaning as a continuous activity,

**NIKLAS FRYKHOLM** is one of the founders of Bitsquid AB, where he is architecting a high-performance multiplatform engine for licensing. A recently released title based on the engine is *WAR OF THE ROSES* from Swedish developer Fatshark.





# SEEING THROUGH

# “Goggles”

pyroland surrealism, and how your art can  
*mess with players' minds*

**“One shudders to think what inhuman thoughts lie behind that mask.. what dreams of chronic and sustained cruelty...”**

Valve's brilliant “Meet the Pyro” trailer for TEAM FORTRESS 2 is a masterful bit of black humor. The juxtaposition of fiery chaos and chirpy cheer deftly eviscerates our industry's endless appetite for faux-operatic “epicness” with a well-sharpened lollipop. As a game artist, however, I think there's more beneath “Meet the Pyro” than just a gimmicky short (and its corresponding in-game Pyrovision mode). The contrast between the world the Pyro sees and the “real” world gives us a rare glimpse at one of the most interesting and least well-mined aspects of game art: the inherent surrealism of everything we do. If we can examine and embrace this surrealism, we may just find an extra tool in our palette to engage players with our art.

## GAME ARTISTS AS PRACTICING SURREALISTS

» The original surrealists believed in subverting the realistic illusions of traditional art, even though they retained realist trappings that other avant-gardists disdained. The goal was to convince people that reality itself was a

shared delusion with no solid existence of its own. Almost a century later, games are almost perfect embodiments of this basic idea—a set of artificially constructed realities that appear solid and consistent, and yet are completely illusory.

I don't mean that games are surreal because they offer startling imagery. Sure, after two decades of real-time 3D we've finally gotten over the worst of our obsession with photorealism, and now it's easy to find games that toy with the conventions of realistic graphics: exaggerated or cartoon-style shading, color manipulation, and painterly rendering, for example. Once-skeptical publishers, reassured by hits ranging from WORLD OF WARCRAFT, to BORDERLANDS, to TEAM FORTRESS 2, no longer demand that every game look exactly like a TV show. And of course, the renaissance of traditional animation that's accompanied the rise of casual, social, and mobile games has done a lot to broaden our palette.

But even though we've outgrown slavish devotion to realism in rendering, it's a stretch to suggest that MODERN WARFARE 3 has much in common with René Magritte from a visual

standpoint. The surrealism of games lies less in their visual styles than in their essential insubstantiality; in the end, it's all just pixels, which we can splash on screen with all sorts of fancy tricks. We have the power to completely rewrite game reality every 16.7 milliseconds.

What's remarkable is that we almost never actually use this power. Models and textures and animations become, after a fashion, as solid and persistent as the objects and actions they represent, even though this continuity is entirely voluntary on our part—and even when the models or actions are inherently fantastic. We worship consistency. We have the option of suddenly and radically rewriting reality whenever and however we'd like. We just choose not to—at least, most of the time.

## PERSISTENCE OF MEMORY

» André Breton would have salivated over being handed the chance to repeatedly rewrite reality. If Dalí had made a video game, it probably would have been a lot like Pyroland (or CATHERINE). If one were feeling like an art critic, this would

be the place to point out that voluntarily surrendering the pretense of a stable external reality is a powerful tool for deconstructing the illusion of the game world, and inviting the player to recognize the hopelessly problematic nature of game “reality.”

But in the more philistine world of commercial entertainment, we don't usually have the option to pursue those kinds of ideas too far. Occasionally, a masterpiece of postmodernist play also turns out to be a great (and commercially viable) game—PAPER MARIO was a brilliant meditation on 3D graphics, and BIOSHOCK played plenty of games with your head. Most of the time, however, working hard to yank the rug out from under your players usually works the same way avant-garde tricksterism does other media: A few people will love it, but most will stick to safer fare and keep their philistine dollars to themselves.

For developers who are more interested in mainstream success (and paying the mortgage) the illusion of a stable, externalized reality remains a more or less obligatory part of the fantasies that we're selling. Few players would be happy if they plop down \$60 for the chance to smite the minions of the Dark Whatchamacallit and suddenly found themselves bathing them in rainbows and candy instead. Not many players (except, perhaps, ardent fans of LIMBO) will be happy to discover that the timing of their platform jump is hopelessly borked because the laws of motion or perspective have changed midframe.

Nevertheless, even games with no postmodernist theoretical pretensions can find a lot of artistic opportunity in the ability to subvert, circumvent, or just plain muck about with the stability of the player's world. To return to Pyrovision for a moment, it's obvious on reflection that the trick is not really new: Pyrovision is just a variation on familiar gimmicks like night vision, thermal vision, X-ray vision, and the host of other special view modes that games have served up over the years. What's radical is neither the sudden change of player perception, nor the tech behind it (though Valve provides a detailed walkthrough of the technique on its developer blog at [www.teamfortress.com/post.php?id=8502](http://www.teamfortress.com/post.php?id=8502)). The innovation in Pyroland is purely artistic: a technique that's usually deployed to support a familiar shooter mechanic is instead used for aesthetic effect, turning the already campy TEAM FORTRESS world into a brutal full-body takedown of FPS pomposity.

## EXQUISITE CORPSE

Of course, satire isn't the only way in which clever teams can subvert game reality to make an artistic point. Many titles with darker themes blur the boundaries of the “real” game world by overlaying ghostly or possibly imaginary characters—the on-again, off-again specter of Alma in the FEAR series is a classic example.

BIOSHOCK occasionally used “ghosts” to lead the player to plot points and set mood. Even HALF-LIFE'S mysterious G-Man, who crops up inscrutably in inexplicable locations, could be considered an example of the same genre.

In the film world, hallucinatory flash-cuts have a long history—*The Shining* is practically a textbook on the uses of imaginary characters, temporary visions, and impossible spaces. Because the tradition is so well understood, hallucinatory visuals in games are really more like cinematic vignettes or scripted sequences than a radical break in the rules of the game world—particularly when it's accompanied by a burst of discordant audio.

This level of comfort with said hallucinations makes this technique a very lightweight form of exposition, and a very efficient storytelling mechanism. The here-and-gone again “ghost” visuals don't require you to break in-game flow like a cutscene, and since they tend to draw attention to themselves, they don't require the careful staging of a scripted sequence. Despite this long track record, however, this technique is rarely seen in games outside the horror and thriller genres, which is a shame. It's easy to see how flash-cuts, in the form of conventional psychological flashbacks rather than supernatural events, could fit into a conventional military shooter, a story-driven cop game like SLEEPING DOGS, or character-driven RPG.

## THE LUGUBRIOUS GAME

The inherent instability of game worlds is even more obvious in multiplayer games. Despite our labors, shared online spaces are full of small paradoxes, errors, and lacunae. The basic building blocks that we ordinarily use to create a reality for our players—continuity in space, continuity in time, and the laws of game physics—all become unreliable when they have to be synchronized between multiple clients over the imperfect medium of the Internet. We're unhappily injured to teleporting characters, unsynced animation, and conflicting views in online games—though I'm sure that somewhere in academia there are eager critical theorists gleefully outlining how radically post-modern all those damn glitches are.

Be that as it may, the imperatives of online play have tended to make us look at online reality as a problem to be solved, rather than an opportunity for artistic exploration. If you're an animator tasked with multiplayer animations, you'll be hemmed in on all sides by technical requirements meant to prevent the sort of situation where one player thinks they're safely under cover while an opposing player has their noggin in the crosshairs. Artistry and nuance get a lot less attention than network prediction and reliable hit resolution, which is why multiplayer animation is one of the toughest jobs in game art.

That said, a surrealist approach to online games has fascinating possibilities. Modern game design has many sophisticated tools for encouraging different kinds of interactions between players—to make them cooperative or mutually suspicious, to encourage them to share information or to work in secret, and so on. However, many of these designs are not well supported by visuals. Instead, they're being communicated through ham-handedly “gamey” mechanisms like points or multi-key puzzles.

If we start from the surrealist position—that there's no single authoritative view of the world that all players must share, but rather a shifting collection of individual perspectives (each of which we can manipulate for effect), then many interesting possibilities arise. Allowing some characters fleeting glimpses of enemies (real or imaginary) that aren't visible to others could promote communication—or paranoia, depending on the mood we want to establish. NPCs who appeared subtly different to different players in the same party could provoke meaningful discussions about who could be trusted, as well as priming players with different emotional cues for events to come. And of course, asymmetrical views of the world are a very powerful tool for shaping the perceptions of different player races and factions. Does the world really look the same to elves, dwarves, and halflings? Or can we encourage a wider range of experiences and interdependence among players by visually rewriting the world in the idiom of each type of player character?

Game artists love to complain that, unlike our cousins in the film business, we have no control over the player's point of view. An animator I know once described his job thusly: “I'm supposed to make a great movie, except my director of photography is a 13-year-old on his fourth can of Red Bull who insists on crouch-jumping during key scenes.” It's true that we don't have the kind of compositional control and framing that make movie stills so appealing, but it's also worth remembering that we have all sorts of powers that can compensate. We can change the laws of the universe any time we want to, in pursuit of the stories we want to tell and moods we want to evoke. You don't need to be a surrealist or a theoretician to want to grab every one of those advantages when you can. And in the case of control over the player's perceptions, it's mostly a matter of changing your own mental filters. Just put on these goggles and you'll see what I mean—it's magical! 🌀

---

**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*.



# HOW TO BECOME A GAME DESIGNER

## FIVE TIPS FOR SWITCHING TO THE GAME DESIGN TRACK

People enter the game industry for many different reasons. For some talented artists, programmers, and musicians, a game job is a great way to employ their talents in a vibrant and creative field. Others simply enjoy being involved with one of their own hobbies and personal passions. However, for many, there can be only one reason to join the industry—to become an official game designer.

The simplest way to become a designer, of course, is simply to start making games. Individual developers can take advantage of more tools and distribution channels than ever before to make great games. Andreas Illiger made *TINY WINGS*. Brendon Chung made *ATOM ZOMBIE SMASHER*. Vic Davis made *ARMAGEDDON EMPIRES*. Jonathan Mak made *EVERYDAY SHOOTER*. You do not need anyone's permission to become a game designer.

Nonetheless, not everyone has the resources, or simply the guts, to go it alone. Unfortunately, for established companies, starting game design jobs are nearly mythical; the job simply requires too much experience, and the competition is too fierce. Most game companies are already full of developers who want to be designers, so most new recruits are hired because they possess a specific skill, such as coding or art. One needs to earn the position of game designer, and one earns that position on the job. If you position yourself

correctly to do design work, design work might just find you.

### 1/ LEARN TO PROGRAM

» Games are a very broad category, often encompassing multiple art forms (words, music, and visuals). Some games have strong story elements. Some are almost pure abstractions. However, the one aspect they all share is that they are all based on algorithms. Code is the language of games, and knowing how to code will help qualify you for a great variety of roles.

Maybe someone needs you to script enemy behavior, or your team needs a scenario editor but no one has time to build it. Perhaps your game needs more random map scripts, or a senior designer needs someone to prototype a new idea for a game. All of these tasks could grow into more established game design roles, but only a programmer can undertake them.

### 2/ WORK ON THE UI OR AI

» There are two areas of game development that are

not strictly thought of as "game design" but actually are—user interface and artificial intelligence. Artificial intelligence controls the behavior of non-human agents in the game world, and as such it is so inseparable from gameplay that working on AI is impossible without daily interaction with the designers. If an AI coder does a consistently good job and keeps asking for extra responsibility, game design is the obvious next step.

This path is even more clear for interface work, which is on the very forefront of the user's experience. Game mechanics are useless if they cannot be communicated to the player, and UI is the most important tool for solving that problem. Thus, interface design is game design. The best part of the "interface track" to game design is that very few game developers want to work on the interface. Senior artists and programmers often view interface work as only suitable for junior developers.

Use this prejudice to one's advantage and volunteer for the job; game companies are always looking for capable developers excited to work on interface design.

### 3/ VOLUNTEER FOR DLC

» Downloadable content can be an easier way to work directly on game design roles. The stakes are inevitably lower for these smaller releases, and a game's official designers are usually too burned out from the final push to even want to think about the DLC, so it can be a good opportunity for aspiring designers to step forward and demonstrate their ambition and potential. Companies want to see their employees grow into the role, as hiring new designers is a huge gamble; DLC provides a great, low-risk opportunity to train them internally.

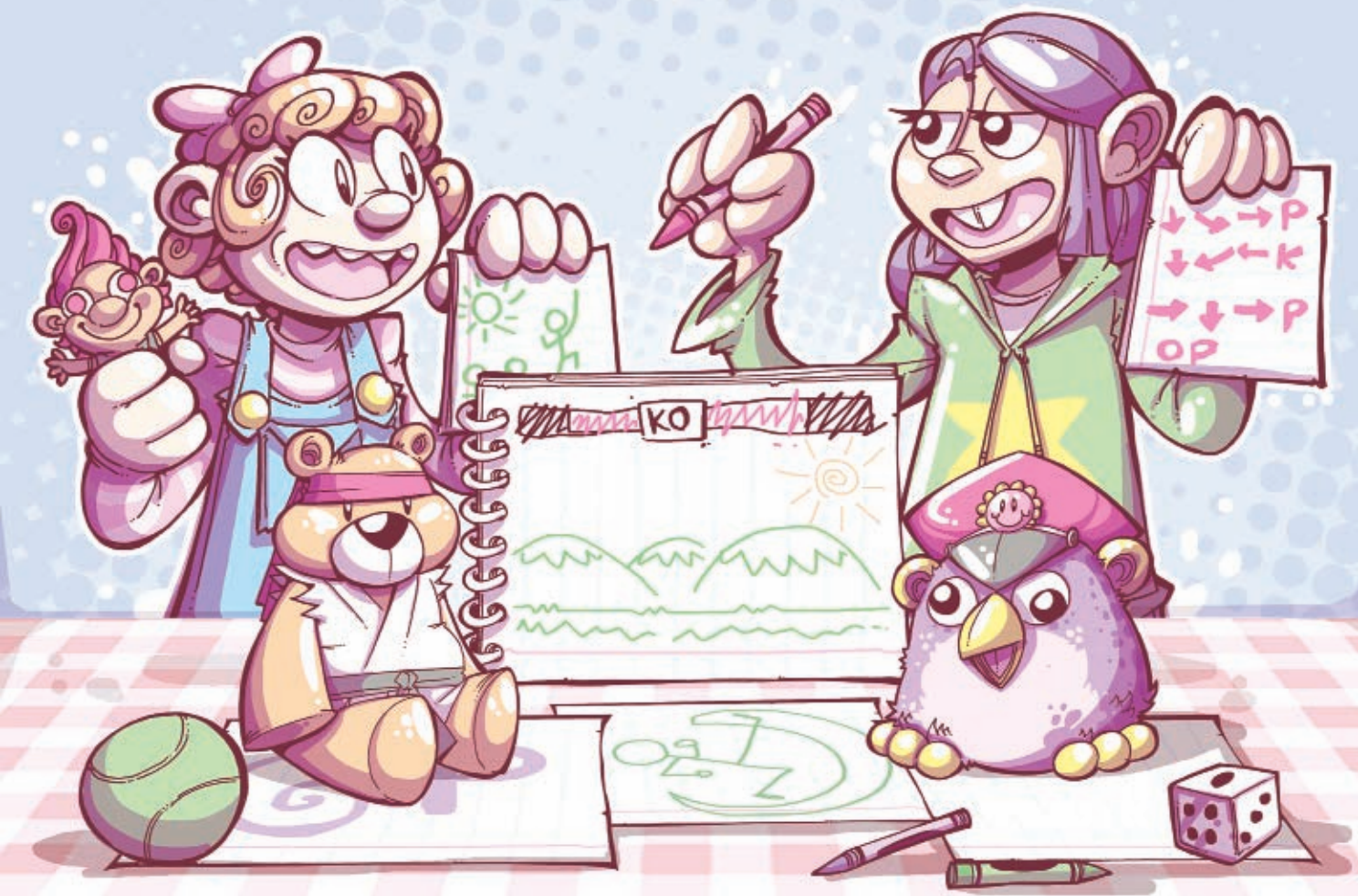
Working on DLC design also has a huge benefit for the aspiring designer—namely, you don't have to start out trying to create fun from a blank slate, which can trigger crippling pressure for a new designer. Instead, you can simply continue to iterate on the core design, while applying lessons you've learned after seeing your game in the hands of thousands of players. Most games have plenty

of low-hanging fruit that only becomes obvious after release; focus on these improvements, and players will respond positively.

### 4/ FOCUS ON FEEDBACK

» Game design is part talent and part skill. Noah Falstein once postulated that a disproportionate number of designers are INTJ on the Myers-Briggs scale (meaning Introverted, iNtuitive, Thinking, and Judging), which suggests that some personalities are better suited to game design than others. However, talent will never be enough; you should actively develop your design skills, and there is only one true way to do that—implementing a design and then listening to user feedback. My own design education didn't really begin until the day *CIVILIZATION III* was released, and players proved that many of my assumptions about how the game played were completely false.

A game is not an inert set of algorithms; it is a shared experience existing somewhere between the designers and the players. Unless a game is constantly exposed to a neutral audience, its design is only theory. Games should have as much prerelease public testing as possible; the designer's skills will only



grow stronger with each successive exposure. Aspiring designers must find some way to experience this feedback loop. Releasing a simple mobile game or a mod to a popular game and then learning from the public feedback is much more valuable than working on some mammoth project which is unlikely to ever gain an audience before release. Even creating a simple board game can improve one's skills as long as the designer can find a testing group for feedback.

### 5/ BE HUMBLE

» Personal humility is a key attribute for success in today's game industry. A designer must accept that *a majority of their ideas are not going to work*. Indeed,

the game designer's job is not to follow one's muse or ego, but to choose a vision and let the team lead the way. Designers need to be humble listeners, not persuasive orators. If a designer ever finds themselves arguing why a playable game mechanic is fun to a skeptical audience, then the game might be in big trouble. Designers still need to be assertive and confident—or else no one will ever take them seriously—but they also need to be humble enough to be able to see things as they are, not how they wish them to be.

For aspiring designers, of course, this rule counts double. Coming across as arrogant or too certain of one's ideas is a sure way to appear unready for the job. Having a great idea that

no one takes seriously can be immensely frustrating, but the key is to maintain the right attitude. If your idea gets implemented, don't think that your idea has *won*, think of it as putting it to the test. The real work begins once the idea is playable, and then it belongs to everyone. All game teams have more ideas than they will ever be able to implement, so developers should all ensure that the best ideas are pursued, regardless of their origin. Indeed, the origin of an idea is usually forgotten; what is remembered is who put in the hours to get it right.

### SHOULD YOU BE A DESIGNER?

» Finally, all aspiring game designers should answer these simple questions:

Have you ever made a video game? A scenario or a mod? A board or card game?

If you answered no to all of these, then you should ask yourself if you really are meant to be a game designer. Painters start drawing when they are young. Musicians learn to play instruments in grade school. Writers start to write. Actors act. Directors direct. Young game designers make games. If it's a passion—and it has to be a passion to succeed—then designing games is something that you absolutely have to do, not just want to do. A true game designer cannot be stopped from creating games.

Designing games is not the same thing as

playing them. The set of people who enjoy making games is much, much smaller than the group of people who enjoy playing them. Designing a game can mean years and years perfecting a single concept and demands the strength to learn from all the criticism that will be heaped upon the design. Ultimately, the mark of a true game designer is that, given free time on some random weekend, that person will sneak in a few hours of what he or she enjoys most—making a new game. 🐼

**SOREN JOHNSON** was the co-designer of *CIVILIZATION III* and the lead designer of *CIVILIZATION IV*. He is a member of the GDC Advisory Board, and his thoughts on game design can be found at [www.designer-notes.com](http://www.designer-notes.com).



MONTREAL  
INTERNATIONAL  
GAME  
SUMMIT

---

NOV.  
13  
14  
2012

---

HILTON  
MONTREAL  
BONAVENTURE  
HOTEL

# MONTREAL INTERNATIONAL GAME SUMMIT

## The Future. Unknown.

---

LET'S SEE WHAT IT MEANS FOR GAMES

Over 80 lectures  
from renowned  
experts of the  
game industry

An exhibition  
zone to discover  
and play

A Business  
Lounge to deal  
and network


Many cocktails  
& activities



Part of



Presented by

CMF  FMC

Visit us at [MIGS.CA](http://MIGS.CA)





# IT'S JUST A JUMP TO THE LEFT

## DYNAMIC ANIMATION SOUND NOW

The art of animation sound tagging in games has blossomed from single-sound playback to a multilayered mashup of dynamic sounds. As game developers seek to match player interactions with ever-more-responsive character representations, sound is working toward the same dynamism. However, if we want sounds to take in-game variables into account, we'll need to move the sound-authoring process out of the Digital Audio Workstation and into the development pipeline, which means that you'll need to have a strong understanding of how these variables work so you can pull your sound design together and make it sound like one cohesive whole.

### AND THEN A STEP TO THE RIGHT

» Fundamentally, a sound for an animation is played (or triggered) based on a timing reference (either seconds or frames) corresponding to the moment when the sound is intended to be heard, though how developers author this depends on their tools and pipeline. Sometimes, this information is added as tags or flags within an animation tool that are then communicated from the animation, and that animation tool simply saves that information to a text file that tells the engine which sound file should play and when at runtime.

Most of the time, playing a single sound is enough, but there are a handful of regular in-game interactions that require additional considerations:

### WILL THE SAME ANIMATION BE USED WITH DIFFERENT CHARACTERS, AND IF SO, SHOULD THE SAME ANIMATION PLAY DIFFERENT SOUNDS?

### SHOULD THE SOUND CHANGE BASED ON SURFACE MATERIAL OR CLOTHING TYPE?

### CAN THE MASS OR SPEED OF THE ANIMATED OBJECT CHANGE?

### WHAT PERSPECTIVE WILL THE PLAYER HAVE WHEN HEARING THE SOUND?

### SHOULD THE SOUND BE DIFFERENT WHEN PLAYED INDOORS VS. OUTDOORS?

By leveraging dynamic aspects of the game and designing sounds that can react accordingly, you can better define your palette of animation-based sounds and make sure the player feels your sound design is fresh, reactive, and diverse.

Once your sounds are ready for animations, you'll need the ability to audition sounds within your animation tool to see how your sound-animation interactions play out in theory. Current-generation game development often depends on a developer's ability to iterate quickly toward a final result. In the frame-to-frame immediacy of the animation pipeline, this means you'll need to play sounds from an animation with different conditional combinations such as footstep type vs. surface type, spatial or positional considerations, and character or outfit type, among others.

### SPACED OUT ON SENSATION

» But the fun doesn't stop there. Other game design elements can expose different aspects of your game's animation pipeline and thus, its relation to sound. For example, think about how your sounds would play differently with factors like time dilation, dynamic player movement speed, and extreme camera angles.

You probably won't ever forget the time you first saw the slow-motion bullet-time sequence in *The Matrix*. The first *MAX PAYNE* turned bullet-time (that is, the concept of adjusting in-game playback speed) into a proper game mechanic that we would later see echoed

by *FALLOUT 3*, *MIRROR'S EDGE*, and several *PRINCE OF PERSIA* titles, among other games.

This time manipulation is often reflected as a percentage variable plus or minus the normal playback speed of the game engine or animations, and that variable is often used (in addition to any other

The sound and position of these same feet can also wreak havoc in certain situations specifically when an in-game cinematic features feet moving up close. Imagine a scene of the player's feet moving stealthily toward cover, or a cinematic that focuses on a player climbing a ladder, for example, you'll immediately see



specific audio mix-related changes) to pitch specific parts of the in-game sound in accordance with the timing change. That means that for animations playing back at a slower speed, the sounds triggered via animation often unfold under a sonic microscope of detail that is usually undetected during normal playback. Depending on how you've designed your sound, this could be a very good thing or a very bad thing.

For example, when the player is in control and able to creep along at a snail's pace, the need for separate heel and toe sounds can make or break the player's feeling of immersion during these highly focused situations. Additionally, there is always the possibility of walk and run animations that "blend" from one to the other based on a parameter from the game defining the player's speed. In these cases, the footstep sound should ideally be exchanged when the player transitions to a running state in order to represent the increased impact of feet pounding the pavement.

that sounds for explicit left- and right-foot positioning should be translated to the correct speaker. This means that during animation tagging, you'll need to indicate the appropriate footstep sound and playback position for a specific foot or foot bone.

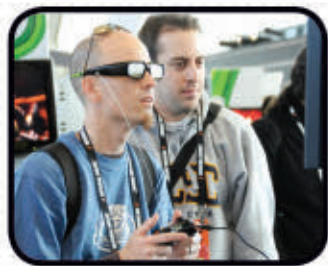
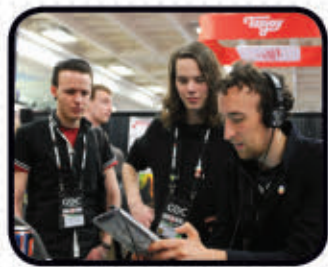
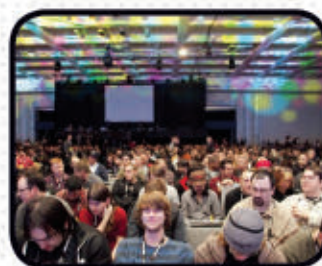
### LET'S DO THE TIME WARP AGAIN

» Our industry (and our audience) continues to demand more cinematic-looking games which means more cinematic-sounding games as well. Take advantage of your ability to tie your sound design to the gameplay itself, and this additional level of detail will help your players favorably measure their in-game experiences against their out-of-game experiences.

"With a bit of a mind flip, you're into the time slip, and nothing can ever be the same." —*The Rocky Horror Picture Show*

**DAMIAN KASTBAUER** is a technical sound design vagabond living out of a suitcase at [LostChocolateLab.com](http://LostChocolateLab.com) and is @lostlab on Twitter.

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



# GDC Vault

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

**EDUCATION**  
GROUP RATES AVAILABLE!

For more information visit: [WWW.GDCVAULT.COM](http://WWW.GDCVAULT.COM)



# TELL STORY, SELL GAME

## USING STORYTELLING TO CREATE BETTER PITCHES

In my role as the CEO of the only game-focused accelerator in the business, I listen to a lot of pitches. Between pitches from YetiZen applicants, hopeful game developers at the YetiZen pitch competitions, hardcore hackers at hackathons, and game jams I judge or organize, I'd estimate I probably hear about a thousand pitches a year.

I learned early on that a poor pitch is not necessarily a mark of a poor entrepreneur. Pitching is a learned skill—and one that most talented game entrepreneurs haven't spent time perfecting. Because of this, my team and I take significant time to get to know the people behind the pitch. In the last year alone we have had half- to one-hour conversations with a few hundred potential entrepreneurs, many of whom were often rejected from the running by our fellow judges at pitch competitions. When it comes to dealing with most potential investors and publishers, however, you won't have that much time to work with. If you want to make your pitches more impactful in less time, you'll need to start by telling a story.

### STORIES WORK

» Most pitches I hear start out with either 1) market numbers that are commonly known by those in the game industry, such as the number of mobile games or the dollar value of the virtual goods market, or 2) by mentioning

no numbers at all, leading instead with something generic like "We are a social mobile gaming company." Both of these are damaging to the pitch, since you have not presented something novel or interesting about yourself or your team.

When I point this out I am often told: "We can't share internal company numbers—that is competitive information!" or that "Our current numbers may not be as good as those you are hearing thrown around by larger companies in the press. We need investment to get there. Sharing our numbers now will only make us look bad." If that's the case, then you should...

### TELL ME A STORY

» While the writing of numbers has existed for over 40 thousand years, the writing of language has only been around for a little over five thousand years. [1] This means that for the majority of history, the way human beings communicated ideas and passed them down from generation through generation was through oral storytelling. As a result, our brains have evolved to accept and remember stories—our very survival depended on it. For other reasons on why stories work please see my essay on neuroscience and pitching (the essay is not specifically on the use of stories, but many of the neuroscience fundamentals apply to storytelling as well)[2].

But before you start telling stories all the time, keep in mind that not all stories are equal. Some



are clearly better than others. The type of story I recommend most for a pitch is the creation story.

### TELLING YOUR CREATION STORY

» A creation story is a story that explains why you started your company. These stories are generally characterized either by a painful moment, frustration, or a more positive and inspiring experience. Whichever you choose, it has to be followed by a deep desire to create the reality of your vision by forming your company. The key to an effective creation story is that the listener must understand the reason you formed your company at a core emotional level. You know you have hit this level if you see your audience respond with a "me too," or "I get it!" or are otherwise intrigued and hooked by what you will say next.

Note that this effect can be achieved with very few words. The length of your pitch is less important than the use of appropriate emotional punctuation—use pauses, silence, volume, and other gestural mechanics to transmit

the emotional moments in the story. After all, nonverbal communication is 93 percent of all communication. [3]

The mistake most entrepreneurs make after this point is that they follow up by talking about their product when they should be using the story as a launching pad into compelling facts about the market. If you can show that the story applies not just to you but many other customers like you, you have effectively made the leap from story to factual evidence. Only then can you delve into the nitty-gritty of execution and your secret sauce.

There you have it. Try it out in your next few pitches. If you want to talk about this further, you know where to find me. Do let me know how this ends up working out for you! 🙌

**SANA N. CHOUDARY** is the CEO and founder of YetiZen ([www.yetizen.com](http://www.yetizen.com)), which includes the YetiZen Innovation Lab (a game-industry community space) and the games-focused YetiZen accelerator program. You can follow her and YetiZen at [yetizen.com/blog/](http://yetizen.com/blog/) or ask her a question on Twitter at @SanaOnGames.

### FOOTNOTES

- [1] [http://en.wikipedia.org/wiki/History\\_of\\_writing\\_ancient\\_numbers](http://en.wikipedia.org/wiki/History_of_writing_ancient_numbers)
- [2] <http://yetizen.com/2012/07/04/avoid-nude-beach-pitches/>
- [3] [http://humanresources.about.com/od/interpersonalcommunication/a/nonverbal\\_com.htm](http://humanresources.about.com/od/interpersonalcommunication/a/nonverbal_com.htm)



# VIDEO GAMES IN RETROGRADE

## HOW DO YOU RECAPTURE VINTAGE MAGIC ON MODERN TOUCH DEVICES?

One of my current game projects is a 16-bit arcade-style racer for mobile devices, which puts me in the position of trying to marry an older aesthetic with newer input methods and design. Along the way, I've been playing (and watching videos of) lots of games from the era, to try to catch hold of the vibe, while also seeing what works in modern times. Through the process, I've found there's a lot that can be learned from what's good and what's bad about older games.

### HOLISTIC DESIGN

» With older games, graphical techniques were a lot less sophisticated (in certain ways—I will still never understand how those assembly wizards managed to create THE ADVENTURES OF BATMAN AND ROBIN on Genesis hardware). Because of this simplicity and unity of hardware, you could create a game that felt completely self-contained from start to finish, from its menus to its music. Everything was being drawn in the same way by the same part of the hardware, there was usually just one sound chip, and the hardware's limitations often guided what could and couldn't be done. In many modern titles, the variety of shaders, textures, and lighting arrangements that are possible make it tougher to keep menus, characters, text, and environments feeling like they're 100 percent part of the same world.

For a retro game on modern hardware, this gets more interesting. For my racing game, it would be far easier to create a 3D road—but how would that fit with our pixelated cars and backgrounds? If you look at FINAL FREEWAY 2R on iOS and Android, the 3D road feels very smooth compared to a proper 16-bit approach, and the ability to can't the camera as you turn makes the pixels of the cars behave in odd ways as it rotates in hardware instead of being redrawn. That's clearly what they're going for—they've used a variety of techniques to try to emulate that old look, but for my purposes it can't match the dirty, sketchy roads in ROAD RASH II on Genesis hardware, which make you feel like the road is the game.

Our current road approach is to slice the screen into small horizontal chunks, and as the camera moves forward, it looks for new chunks to render. This includes small bits of road, but also spaces out our sideline environmental

dressings, which live as billboards along those horizontal lines. This emulates the approach of the older games, but is done for every scanline, since modern hardware is much more powerful. While this may not wind up being our final approach, it's interesting to see how neither fully old nor fully new approaches work perfectly when recreating the vintage look on new devices.

The main lesson here is that with a smaller game, we have an excellent opportunity to create a product that looks very integrated and internally consistent, but it can be challenging because newer techniques make it easier to create something that looks almost as good but sacrifices that holistic feel. It's actually tougher to do it the old way!

SUB-TERRANIA



### FLOATY CONTROL

» You'd be surprised how many of those old 16-bit games that we remember as being so sharp and precise actually had rather fiddly controls. In good games like SUB-TERRANIA on the Sega Genesis this was purposeful, and part of the challenge. Your ship was hard to control and had a lot of inertia, but was incredibly quick to turn, meaning if you got used to the unique playstyle, you could master it.

In less-successful games, having too much inertia in a platformer or imprecise hit boxes in a beat-em-up ruins an otherwise enjoyable experience. We see this comparison today, where games like SUPER CRATE BOX on iOS have simple controls (left, right, jump, and shoot) that at first feel imprecise, but have a quick learning curve, and ultimately inform better play. This works because the environments are static, single-screen, and easy to navigate. You don't need to wall-hop or do anything complex—just jump, land on a platform that you know is always there, jump over an enemy, and shoot. On the other hand, there are games that try to be full-on platform game experiences, with triangle jumps and floor stomps, and I've yet to see a one that didn't frustrate me into wishing I had an actual controller.

The lesson here is to make sure your control method informs your design. The simple environments in SUPER CRATE BOX are a good example of this. With scrolling and complex jumps, the game just wouldn't work as well as it does. Another great example is SUPER HEXAGON, which is a twitchy, hardcore game that only lets you rotate left or right. It's precise and accurate, meaning whenever you die, you know it's your fault.

### THAT OLD-SCHOOL FEELING

» More important than anything else is the old-school vibe. It's hard to pinpoint what exactly triggers that particular feeling in a player, but there are some common threads in the more successful older games.

### Pattern learning through failure

Most good vintage games show you how to succeed by showing you how you can fail. Missing jumps, knowing that spikes show up over red pits but not blue ones, realizing you can duck to escape enemy fire—these were the teaching methods of old. (And when you die, it's back to the start of the level with you.) It might feel slightly unfair the first time, but usually if you're quick enough, you can learn the pattern as it's happening without too much actual punishment.

This kind of approach to learning and difficulty has fallen out of favor in modern games, which have frequent checkpoints and

variable skill levels for casual players. But I think the hardcore nature of these older games contributes a substantially retro vibe, and as games like *SUPER CRATE BOX*, *SUPER HEXAGON*, and *SUPER MEAT BOY* prove, good control plus punishing difficulty and pattern learning can really engage players, even today. (Also, adding “Super” to your game title boosts sales.) Repetition bonds you with a product—that’s why kids still learn SAT words with flash cards. We remember the games from our childhood so well in part because we had to play through them so many times in order to beat them.

### Distinctive music

Many games these days (especially in the triple-A space) give music the backseat, and



choose to let it linger in the background without major melodic hooks. *SUPER MARIO BROS.’S* music was fantastic because it added ragtime to an action game. *STREETS OF RAGE 3*, composed by Motohiro Kawashima, is an early study in progressive, experimental electronica. The unexpected sounds give these games an edge.

### Appropriate control

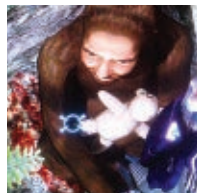
As previously hinted, precision isn’t necessarily the hallmark of good control, but the control method has to be appropriate for the game. There were six buttons (eight if you include select and pause) on the SNES controller, but you didn’t have to use them all. Also, your game’s genre



may inform your players’ preferences; I’m a particular fan of sticky, inertia-free jumps in platforming and action games, but with flight or driving games, I like a bit of wiggle room.

### Easter eggs

One thing that is more difficult to represent in larger games is the little subtle nuances that made 16-bit games so charming. Hidden paths and secrets in these living worlds led to a great feeling of discovery. In *BONK’S REVENGE* for the TurboGrafx-16, enemies on a ship cook food and wander about if you leave them alone. Sure, you’ll find a developer’s face in a portrait in



*UNCHARTED 3*, or an interesting poster in *CALL OF DUTY*, but a better example would be *HALO’S* hidden, hard-to-reach cavemen. In the arcade racer *OUTRUN 2*, when you stay at the starting line instead of driving off with your peers, the flag waver will do stretches, breakdance, and a host of other actions while you wait. There’s no reason to do this—95 percent of players will never see it—but that kind of dedication to the world really helps make it feel alive, and gives the player a great feeling of discovery.

### RETRO FUTURE

» For the last decade or so, retro-inspired developers have pondered what makes these games feel the way they do. Ultimately, the vibe may come down to individual nostalgia—what resonates with that game’s director? To rekindle that feeling myself, I’ve been playing Genesis and TurboGrafx games daily, and encouraging my team to do that as frequently as they can as well. It’s been immensely informative, and I can’t recommend this enough for anyone trying to make a retro title.

I don’t know if we’ll be successful in taking that vintage vibe and bringing it into the future, but the learning process is already showing me how much vintage titles have to teach about present and future games. 🎮

**BRANDON SHEFFIELD** is director of Oakland, California–based Necrosoft Games, and editor emeritus of *Game Developer* magazine. He has worked on over a dozen titles, and is currently developing two small-team games for PlayStation Mobile.



SUPER CRATE BOX.

## GDC NEXT, APP DEVELOPERS CONFERENCE COME TO L.A. IN 2013

The organizers of Game Developers Conference (GDC) have announced a brand-new event, featuring two market-leading conferences and a shared expo floor, debuting in Los Angeles in November 2013.

Game Developers Conference Next will be the reimagined successor to GDC Online, whose final iteration took place in Austin this year from October 9-11. Meanwhile, App Developers Conference (ADC) is a brand-new event dedicated to key takeaways in app technology, creation, business, and marketing outside of gaming.

GDC Next and the App Developers Conference will debut as a co-located event at the Los Angeles Convention Center from November 5-7, 2013.

GDC Next focuses on what's next in smartphone and tablet, social, independent, cloud, and other major forms of games. Whether you're a designer, programmer, architect, producer, artist, marketer, businessperson, or all of the above, the new show will prove vital to making great games and making money in the most vibrant new areas of the game industry.



From the creators of GDC, ADC is a brand-new event focusing on the very best development, UI, marketing, and business of apps. Organizers will be recruiting the very best app creators, whether they be on mobile devices, the web, or beyond, and getting them

to present their best practices to attendees.

Attendees will be able to sign up individually for the GDC Next 2013 and ADC 2013 conferences (November 5-7, 2013) or attend both at a discounted price. The shared expo floor for the two shows will be open to all attendees on November 5 and 6. For further information, please visit the GDC Next website ([www.gdcnext.com](http://www.gdcnext.com)) or the ADC landing page ([www.adconf.com](http://www.adconf.com)). (GDC Next and ADC are owned and operated by UBM TechWeb, as is *Game Developer*.)

## GDC CHINA 2012 ADDS BEJEWELLED BLITZ, ZOMBIE GUNSHIP SMARTPHONE TALKS

The growing session lineup for this November's GDC China in Shanghai is quickly gaining steam, and the show's organizers have revealed three new talks in the Smartphone & Tablet Games Summit.

This time, these new sessions include an examination of PopCap's BEJEWELLED BLITZ, a case study from Limbic Software on the challenges of mobile game growth, and Appy Entertainment on pivoting from "premium to freemium" in the smartphone space.

As part of the Smartphone & Tablet Games Summit, these sessions will run alongside the rest of GDC China from November 17-19 at the Shanghai Convention Center in Shanghai, China, and will offer specialized content covering the ins and outs of mobile game development.

The full details on these new Summit sessions are as follows:

In "BEJEWELLED BLITZ: One Year in the Life of a Top-Grossing Mobile Game," PopCap's Giordano Bruno Contestabile will discuss how



BEJEWELLED BLITZ made the jump from Facebook to iOS, and became one of the platform's most long-lasting success stories. Along the way, he'll explain how other cross-platform social game developers can emulate that success by combining smart game design, data analysis, and more.

Elsewhere, Limbic Software cofounder Iman Mostafavi will host "Getting Out of the Garage:

Managing Teams, Customer Relationships, Cloud Servers, and 97 Other Things to Handle Growth." During this session, Mostafavi will examine Limbic titles like TOWERMADNESS, NUTS!, AND ZOMBIE GUNSHIP, and will detail the technical and operational challenges the studio faced as its games began to grow. It'll be a great chance to learn from Limbic's experience and gain a better understanding of how to deal with rapid changes in scale.

Finally, Appy Entertainment brand director Paul O'Connor (SPELLCRAFT SCHOOL OF MAGIC, TRUCKS & SKULLS) will present "Premium to Freemium: Pivoting Monetization Method for Best-Selling Apps," detailing how his studio "converted two best-selling premium apps to freemium monetization and increased their audience, grew their revenue, and made the games better in the process."

These talks join a number of other previously announced sessions for GDC China, including

a handful of additional talks in the Smartphone & Tablet Summit, Main Conference sessions from Volition and Naughty Dog, and much more. More information on any of these sessions is available on the "Announced Sessions" page on the official GDC China website ([www.gdcchina.com](http://www.gdcchina.com)).

Be sure to keep an eye out for even more updates on GDC China in the weeks ahead, as show organizers have plenty more to announce for the upcoming event. For all the latest information on GDC China, visit the show's official website, or subscribe to regular updates via Facebook, Twitter, or RSS. (GDC China is owned and operated by UBM TechWeb, as is *Game Developer*.)



Photos courtesy of Game Developers Conference and LACC brochure.



## who went where?



/// After 20 years with the company, Cliff Bleszinski (above) has resigned from Epic Games. No word on what he'll do next, other than move on to the "next stage of his career."

/// BioWare cofounders Greg Zeschuk and Ray Muzyka have departed the RPG giant five years after its acquisition by EA. Neither have announced further game-related plans, though Zeschuk will be pursuing a long-standing passion for craft beers.

/// Former CBS TV executive Nancy Tellem is heading up a new studio at Microsoft, which will focus on creating "storytelling experiences for the Xbox brand." The recent Kinect Nat Geo TV and Kinect Sesame Street TV are two early results of this new direction.

/// Two more executives have resigned from social games giant Zynga. Infrastructure CTO Allan Leinwand and chief marketing and revenue officer Jeff Karp join COO John Schappert and vice presidents Bill Mooney and Brian Birtwistle, among others.

## new studios

/// Microsoft has created a new studio that will focus on content for Windows 8 tablets. The London-based developers, led by ex-Rare staffer Lee Schuneman, will focus on the concept of "entertainment as a service."

/// Epic Games is opening a new studio in Seattle. As of this writing, the new studio is currently unnamed and is in the early stages of organizing and staffing.

/// Mobile/social developer Mediatonic has opened a new studio in Brighton, England. The company plans to hire as many as 50 new positions over the next year.

# Designing A Whole New Type of Health Meter

GAME DESIGNER DEREK MANNING TALKS TRANSITION TO HEALTH-CARE INDUSTRY

OVER THE PAST FEW YEARS, GAME DESIGNER DEREK MANNING HAS TRANSITIONED FROM CRAFTING TOMB RAIDER LEVELS TO DESIGNING HEALTH AND WELL-BEING APPS AT ORCAS, A 20-YEAR-OLD HEALTH-CARE BUSINESS THAT IS NOW DEVELOPING INCREASINGLY GAMELIKE INTERACTIVE HEALTH APPS. WE CHECKED IN WITH DEREK TO CHAT ABOUT DEVELOPING GAMES AT WHAT MIGHT BE TERMED A NONTRADITIONAL GAME COMPANY.

**Alexandra Hall:** *What does it feel to leave triple-A development?*

**Derek Manning:** It was quite the transition. I think the biggest change was the shift in demographic. Working on a game like TOMB RAIDER or LEGIONS gave us certain expectations about the player's skill level, and you can make a lot of assumptions. When it came to Facebook, there was such a different crowd that for the most part you couldn't really make any judgments about the experience levels of your user. A lot of them may have just started playing games for the first time on Facebook.

Another big change was the smaller production cycles. You usually have a couple years at least to produce a big-budget console title, but Facebook games only grant you maybe six to nine months. However, the upward trend in the quality of Facebook games has that gap slowly closing.

**AH:** *How do your prior level-design skills apply to your current projects?*

**DM:** The focus of your job is really the same, and that is making sure that you're creating a fun experience for the user. It's all about looking at what you have to work with. I used to look at the mechanics available in a given game and then try to weave those in a fun, unique manner for each level. Now I look at the different resources available in our

company versus the time we have and try to create the best look and experience for the user for each project.

**AH:** *Were there any health-related skill prerequisites for this job class?*

**DM:** Yes, but more for my own knowledge and benefit rather than any requirements from the company. I try to keep up on the latest research on health and happiness in order to create products that will have the biggest impact.



**AH:** *Do you think we'll start seeing more crossover between the game and health-care industries?*

**DM:** I think that very soon we're going to see a lot more gaming companies entering into the health field. Over the recent years our methods of engagement with users has increased dramatically with things such as the Wii, Kinect, tablets, smartphones, and devices like the FitBit. These are allowing endless new possibilities for merging games with health. We've

started to see a few health-related games, such as Wii FIT or YOUR HEALTH, but most of them are solely focused on the exercise aspect. There are many other ways of improving your quality of life that we are just now seeing products tackle.

**AH:** *Do you have a different attitude about your work now that you're making games that aim to concretely improve your players' lives?*

**DM:** My degree is actually in psychology. I was going to school to become a psychologist before I switched to working in the game industry. Ever since I was little I have wanted to help people to be healthier, both mentally and physically. ORCAS provides an amazing opportunity for me to merge my knowledge of gaming with the science literature to make fun, enjoyable products that make life better. It's great!

**AH:** *Has working at a health-focused company changed your own self-care habits?*

**DM:** Not surprisingly, it has. Most of the people here lead very active lives. We even have an ultrarunner on the game team. Being in this kind of health-centered environment has definitely made me exercise more.

**AH:** *What's your #1 health tip for fellow game designers?*

**DM:** Get a standing desk at work. It does wonders for your back! 🏋️

[HTTP://OFLIGHTANDSHADOW.AT/](http://oflightandshadow.at/)

# OF LIGHT & SHADOW

OF LIGHT & SHADOW IS THE GAME DEBUT OF 12 ANGRY DEVS, DEVELOPED OVER 18 MONTHS BY A GROUP OF GAME DESIGN STUDENTS AT THE SALZBURG UNIVERSITY OF APPLIED SCIENCES. OL&S WEAVES PUZZLES INTO ITS PLATFORMING BY LETTING THE PLAYER SWITCH BETWEEN TWO CHARACTERS, ONE OF WHOM CAN ONLY EXIST IN LIGHT, AND ANOTHER WHO NEEDS TO LURK IN SHADOWS. WE CHECKED IN WITH 12 ANGRY DEVS (A MISNOMER ON SEVERAL FRONTS) TO CHAT ABOUT THE DEV PROCESS, WORKING WITH UNITY, AND MAKING GAMES IN AUSTRIA.

**Alexandra Hall: How did the Of Light & Shadow project come to be? Were you all students?**

**12 Angry Devs:** It started with eight MultiMediaArt students who wanted to create a game as part of their MA course. They started brainstorming, came up with the core mechanics, and pitched the project to the programmers at MultiMediaTechnology. Four of them jumped on board—and thus we were “12 Angry Devs.”

We started building a prototype in March 2011, and after that iterated regularly on the results. In January 2012 we had our first finished level, and in June we released the game with seven levels. Recently, we’ve added more difficult versions of these levels to the game—and we’re currently considering the possibility to apply for Steam Greenlight, using the resulting “deluxe version” of the game.

**AH: Did you like working with Unity?**

**12AD:** The Unity engine offers a lot of functionality out of the box. That’s something really useful if you care more about developing a game than

developing a game engine. Unity is also an engine that can actually be used by nonprogrammers as well (to a certain extent, anyway). The editor works in a similar fashion to 3D animation packages, so our artists were able to integrate assets and build levels quite easily. This of course helped us a lot with our workflow, because the programmers could focus on actual programming instead of integrating assets or tweaking values.

A downside of Unity (and its ease of use) is that if the engine doesn’t support a certain feature (like a specific rendering technique, for example), you have little chance of implementing it into the engine. We had this problem early on when we wanted to render some dynamic volumetric lighting. But eventually we managed to work around these limitations by using some shader trickery.

These issues aside, we felt that Unity was pretty much the way to go within a setting such as ours; you just have to be aware of its limitations in advance.

**AH: How’d you come up with the light/dark game mechanic?**

**12AD:** As with other teams we know of, light and consequently shadows are amongst a set of core mechanics and gameplay ideas that are obviously quite popular. That being said, we think we didn’t take an all-too-obvious approach to that particular concept and iterated several times on the idea of platforming combined with lights and shadows. Originally, Mr. Light and Dr. Shadow would have been two separate characters with unique abilities and played in turns, so you can see we’re quite far off with the final gameplay now. Early in development we started to build a playable prototype and constantly kept playing around with the mechanics, filtering out the fun parts of play, and cutting everything that wasn’t fun in our point of view.

**AH: Did you run into any particularly tricky problems during the development process?**

**12AD:** The visualization of the light cones was one issue that kept us busy for quite some time. We experimented with several approaches like volumetric lighting and manipulating meshes, but in the end decided to go with a

**Developer:** 12 Angry Devs

**Release date:** 06/06/12

**Development time:** Roughly 18 months, not full time

**Development budget:** Just enough for Chinese takeout (seriously, we had no budget)

**A fun fact:** In one of our levels the girl of THE BALLOON QUEST floats around in the background. Also, one of the first mechanics for Dr. Shadow was the ability to “flip” as seen in WWW.

**Team members:** Florian Jindra, Michael Fuchs, Markus Huber, Martin Kenzel, Martin Klappacher, Vinzenz Mayrhofer, Sophie Müller, Clemens Stangl, Patrick Topf, Stefan Wiesenegger, Michael Kenzel, Manuel Hoffmann, Vedad Siljak, Andreas Stallinger, Michael Webersdorfer

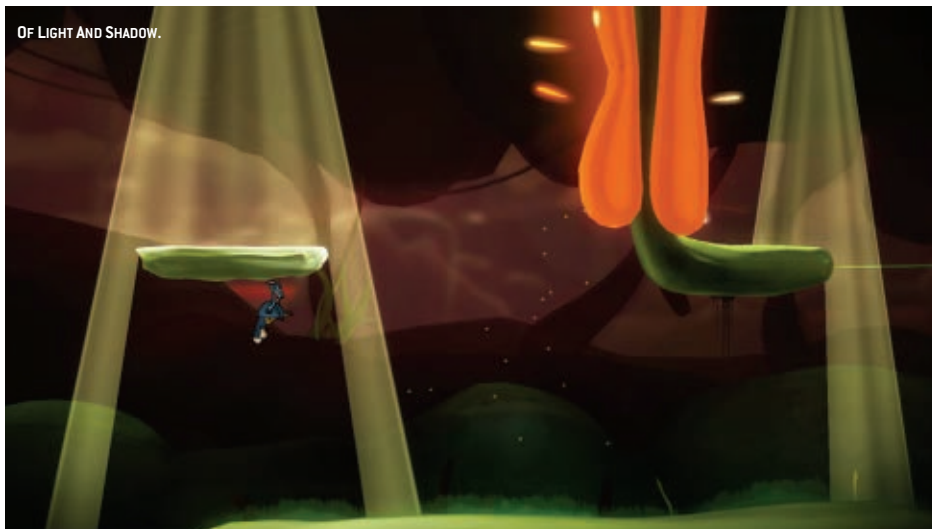
shader-based approach.

Another tough nut to crack was the movement of the Dr. Shadow character. Mr. Light has a pretty traditional movement, but Dr. Shadow needed to be able to walk on walls and ceilings, to stick to and to drop off those level parts. We had a long and tough time of tweaking and adjusting the abilities of Dr. Shadow to make it work within the setting and for each level specifically.

**AH: What’s the dev scene like in Austria? Are there many opportunities?**

**12AD:** A lot of student projects are produced at the University of Applied Sciences Salzburg because there are art and programming courses for game development. There is not a huge gaming industry in Austria, only a couple of companies that are mostly based in Vienna. So quite a few of the students end up going abroad to work after they graduate. Of course, we hope that this is going to change in the coming years because of all the young game development graduates who might try to build up their own company. AND YET IT MOVES developer Broken Rules actually started this way, and more of that is sure to happen. 🎮

OF LIGHT AND SHADOW.







CDM

## BECOME A LEADER IN DIGITAL MEDIA

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

Learn more about our MASTERS OF DIGITAL MEDIA PROGRAM at [www.thecdm.ca/programs/mdm](http://www.thecdm.ca/programs/mdm)

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

CENTRE FOR DIGITAL MEDIA | [www.thecdm.ca](http://www.thecdm.ca)



### ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
EPIC GAMES	C2	PERFORCE SOFTWARE	22
HAVOK	C3	RAD GAME TOOLS	C4
MASTERS OF DIGITAL MEDIA PROGRAM	55	TWOFOUR54	3
MONTREAL INTERNATIONAL GAME SUMMIT	46	VANCOUVER FILM SCHOOL	21

*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.



# THE REAL HOLLYWOOD MOJO

## GETTING BETTER VOICE ACTING IN YOUR GAMES

If you're reading this magazine, you know that video games have surpassed Hollywood in emotional depth, breadth of subject matter, relevance, and, most importantly, revenue (take that, Ebert!). But if there is one area left where games might not yet totally be the equal of the motion picture, it is acting. And I should know! That's the very stuff I deal with for a living as a voiceover director in the video game industry.

Yes, even though voiceover recording has been greatly improved by today's technological developments (motion capture, text-to-speech technology, and so on), there's still room for improvement. Great performances have always been a hallmark of movies (who could forget that orc who runs at Helm's Deep's drainage culvert in *The Two Towers*?)—but they haven't always been a hallmark of games.

Thankfully, the game industry's plan to win at every point-by-point comparison to Hollywood doesn't have to be stymied by the unfamiliar terms and methods of the art and science of acting. With a few simple tips from yours truly, you'll be able to get the voices in your game (or the "VO," as we folks in the know like to call it) at a level where IGN will be hailing it as worthy of a dozen Oscars and Academy Awards!

### TIPS FOR INDIES

» You're probably thinking, "Hey, we can't afford voice acting. We're indies!" But it doesn't have to cost anything if you're clever and resourceful enough. Think about how everyone on the team already has a voice. Can they can read words on a page and then speak them? Well, that's all a "professional" voice actor does, really, when you get down to it.

Now think about all the different voices you have access to. It may seem at first like most of your characters will have to be young white males. But wait! Surely you've got a couple of guys on staff who love to do hilarious voices. Ask for a Sean Connery or Werner Herzog and let diversity fly! (I suppose those examples are also white males, but hey, you get the idea, right?) Audio programs often have features



that make people sound different, anyway. They have dials and knobs that can make someone sound like a zombie or a pirate, or even a zombie pirate ghost!

Oh, yeah: Don't forget that you need a computer with a microphone. Most laptops have one built in these days.

### TIPS FOR PROS

» In most cases the approach above will work wonders, but if you're big shots and rolling in money (call me, Mr. Kotick! I can help you!), you might think about hiring some professionals. Working with actors can be intimidating at first—you've probably heard all about how they make millions of dollars, have temper tantrums on the set, and make weird demands. But a little confidence can go a long way to coaxing that perfect guttural outburst from a star's vocal chords.

The director's first job is to make sure the actors really

understand what's going on in the game. How are gamers going to be able to understand the difference between pre-Empire Maldichori GuardWalkers and post-Empire Maldichori GuardStalkers if the actress playing the Academician who explains the difference can't explain it herself? You know?

After you've walked your actress through the nuances of the different societies, political systems, histories, and mythologies of your game world, you'll begin to really "direct" her as she acts. Now, directing is truly a mysterious art. I have been doing it for over three months now and I daresay I still don't understand all of it. Some people think it consists of sitting in a folding chair and yelling "Cut!" every so often, but nothing could be further from the truth: It is highly engaged, active, and exhausting work.

By way of an example, I present a transcript from one of my most recent sessions.

*[recording begins]* "...Great. Okay. Now, you're dying from a blunt instrument to your abdomen, even though you had a 'Flesh to Iron' spell cast at the time. But it's the 'Flesh to Iron' learned from the trainer near the SkyVessel docks, not the trainer near the old guard tower that was burned in the Crisis of Hellistria riots. So you're kind of, like, sad, but also angry at the same time. But still dying—the dying is important. The player needs to feel the dying. Again." *[recording ends]*

Note how I've expertly woven in the vital contextual information that the actor needs to hear while still focusing on the outcomes. Directing is no easy task, but that's why directors are famous and get paid so much (except for me [just kidding (Bobby! Call me!!)]).

### THAT'S A WRAP

» I could talk forever describing more tips and tricks—like how to save time by making new lines out of the ones you've already recorded, how to navigate those arcane union rules, even the subtle differences between "grunts," "exertions," "exclamations," and "heaves"—but it looks like we're out of time for today. Look for my upcoming seminar to be held in the Activation headquarters parking lot! If anyone sees Bobby walking by, let me know! 📞

---

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

---

MAGNUS UNDERLAND writes about games and other topics at [www.above49.ca](http://www.above49.ca). email him at [magnus.underland@gmail.com](mailto:magnus.underland@gmail.com).

# UNLOCK THE POTENTIAL OF VIDEO GAME DEVELOPMENT



Award-winning technology.  
Unparalleled support. Flexible licensing options.

**Learn More:** [www.havok.com](http://www.havok.com)

Havok is hiring! Visit [www.havok.com/careers](http://www.havok.com/careers) for more information.

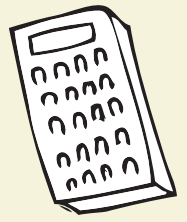
#### **Havok™ Technologies Include:**

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

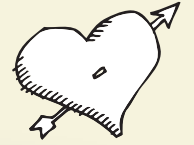
IF YOU WANT TO BE AN EVEN MORE

# AWESOME

I ♥



# GAME DEVELOPER



THEN YOU NEED TO BE USING

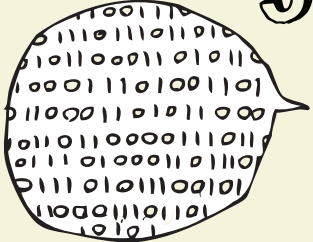
# Granny 3D

.....



You get a run-time dynamic

## 3D ANIMATION SYSTEM with



AMAZING content exporters

EASY data manipulation



# MUSTACHES

Now includes the



## Granny Animation Studio

.....



USING OUR TOOLS NOT ONLY MAKES YOU

MORE awesome, THEY MAKE YOU rad!



www.radgametools.com  
sales3@radgametools.com  
(425) 893-4300