



R E P O R T E D I N M A G

O C T O B E R

gd

BULLETSTORM  
POSTMORTEM

20 GAME CHANGERS

THE LEADING GAME INDUSTRY MAGAZINE VOL.18 NO.9  
OCTOBER 2011 INSIDE: REACTIVE GAME ARCHITECTURES WITH RIX

UBM

# Some say he made a deal with the Devil.

Others say he's some kind of wizard. They say it's the only way he can consistently turn in quality code on time...and still be able to take a vacation every year.

He's not using Black Magic. He's using Seapine Software's application lifecycle management tools.

Seapine ALM tools help you deliver products faster by eliminating steps, streamlining processes, and automating repetitive tasks. With Seapine, you can find issues faster, fix bugs sooner, and deliver higher quality games.

Seapine's solutions help ensure software quality at notable studios such as Epic Games, 2K Games, Microsoft Game Studios, and more.

Visit [www.seapine.com/gamedev](http://www.seapine.com/gamedev) today, and say goodbye to long hours, sleepless nights, and skipped vacations!

 Seapine Software™



HEADSHOT +50

gd

DEPARTMENTS

- 2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]  
Interactive History
- 4 **HEADS UP DISPLAY** [NEWS]  
New games for vintage consoles, Michael Jackson visits Sega, and ASCII Animator released.
- 27 **TOOL BOX** *By David Hellman* [REVIEW]  
Corel Painter 12
- 34 **THE INNER PRODUCT** *By Peter Drescher* [PROGRAMMING]  
Programming FMOD for Android
- 40 **DESIGN OF THE TIMES** *By Soren Johnson* [DESIGN]  
Taking Feedback
- 42 **PIXEL PUSHER** *By Steve Theodore* [ART]  
Get The Memo
- 44 **THE BUSINESS** *By Kim Pallister* [BUSINESS]  
Efficiency...For Whom?
- 46 **GDC JOBS** *By Mathew Kumar* [CAREER]  
Recruitment at GDC Online
- 47 **AURAL FIXATION** *By Jesse Harlin* [SOUND]  
Separation Anxiety
- 48 **GDC NEWS** *By Staff* [NEWS]  
2011 Game Developers Conference Europe Hosts More Than 2,100 Attendees, GDC China Honors Winners Of Shanghai, Beijing Game Jams
- 49 **GOOD JOB** *By Brandon Sheffield* [CAREER]  
Q&A with Ken Taya, Who Went Where, and New Studios
- 50 **EDUCATED PLAY** *By Tom Curtis* [EDUCATION]  
A FLIPPING GOOD TIME
- 56 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]  
Sandwich Pitches

CONTENTS.1011  
VOLUME 18 NUMBER 09

POSTMORTEM

- 20 **BULLETSTORM**  
BULLETSTORM is a colorful skillshot-fest that took 3.5 years to make. It didn't perform quite to expectations at retail, but the experiment was by all other metrics a success. This straight-shooting design-focused postmortem discusses everything from emergent feature discoveries to downloadable demo woes. *By Adrian Chmielarz*

FEATURES

- 6 **GAME CHANGERS**  
The game industry is a dynamic and fluidly-changing one. But who (and what) are the companies and concepts that are shaping the game industry today? Our answer to this question is 20 companies, processes, and concepts that are changing the game. *By Brandon Sheffield*
- 13 **REACTIVE GAME ARCHITECTURES**  
Reactive programming represents a different way of thinking about code, moving toward a model that focuses more on expressively describing when things happen in games. Gary Dahl illustrates these ideas in Microsoft's Rx. *By Gary Dahl*
- 30 **THE IRON FIST**  
TEKKEN is a venerable 3D fighting game series, dating back to 1994. As the TEKKEN team looks to complete its mashup with the STREET FIGHTER series, we spoke with game director Katsuhiro Harada about the first game's origins. *By Brandon Sheffield*



# INTERACTIVE HISTORY

ON THE IMPORTANCE OF PRESERVING GAMES' RICH PAST

## I THINK HISTORY IS IMPORTANT. AS

George Santayana famously wrote in *The Life of Reason*, "Those who cannot remember the past are condemned to repeat it." If you look around you, it's clear that we are actually doing a lot of repeating nowadays. As the game industry prepares for 3D in the browsers, and we watch the increase in quality of games on smartphones, and even as one looks at the increasing complexity of indie games, we can also see a lot of people making mistakes that were solved during the 32 bit console era, or earlier.

While we can't stop everyone from re-learning all these lessons—there are tons of pitfalls in game development, after all—helping more people to understand the origins of games and game development is a worthwhile endeavor. Postmortems go a long way, but aside from those on *Game Developer* and *Gamasutra*, not that many are public. There are technical manuals and theory books, but when you come down to it, most of the game industry's collective knowledge is stored up in the thoughts and discussions of those who lived it.

Ever since my first GDC I have been privy to panels, discussions, and proposals suggesting how to preserve games. Should we keep the physical media? Do we try to find all the source code? Should we emulate everything? What do we do for game consoles that no longer exist in the world, or of which there are very few? How do we present this content? There are more questions than answers, to be sure.

Many preservation efforts are happening on the hobbyist level, scattered according to the interests of the person undertaking them. The folks at [www.visual6502.org](http://www.visual6502.org) are a great example. They've taken the venerable 6502 chip, which was used in the Apple IIe, the Atari 2600, the Commodore 64, and many others, and burned down each layer of the chip with acid, mapped the silicon die and its substrate, then

created polygon vector models of the chip's physical components. This allows chip-level accuracy in emulation, and at the very least enables the chip to be recreated if necessary. And this is fantastic, but this is just one chip! The group is working to save more chips, especially the rarer ones, but they can only do so much.

## THE LIVING CANVAS

As the preservation movement picks up speed, so too does the desire to show off these collections, or at least make them available to the right people. This, of course, leads to museums. One of the largest is the International Center for the History of Electronic Games (ICHEG) at the Strong Museum of Play, which has a huge library of games, but also has a trove of documents and artifacts from persons like Ralph Baer, Dani Bunten Berry, Don Daglow, Ken and Roberta Williams, and Will Wright. These documents are priceless windows into the thought processes of the time, and it's fantastic that they're being preserved.

Then there's the Video Game History Museum, put together by a few collectors from the Digital Press forums. Their collection includes full runs for multiple consoles, collections that truly could not exist again without the help of extreme investments. There are several other minor museums cropping up here and there, and even the Smithsonian has an upcoming exhibition on the art of video games.

These collections are all laudable, but once they're established, a new problem emerges. How do people get to see this stuff? The ICHEG is off in Rochester, NY, and the Video Game History Museum is setting up in Silicon Valley, which is convenient for Californians, but not rest of the world. And even if these objects are on display, how do people interact with them? You don't want someone touching that sealed copy of STADIUM EVENTS for NES [one recently sold for \$41,300], or rifling through Ralph Baer's papers with their taco hands.

And just looking at these things from afar is not very engaging. To that end, they really need what amounts to a community manager. Preservation is only half the battle. What good is a gorgeous collection that nobody can play with?

## FACE THE MASSES

Museums can often be static and sterile, but games are vibrant and interactive. There needs to be some care put into their presentation. In the case of game collections, rare artifacts can be presented with videos, screenshots, and most importantly context. There's no easy solution that would allow people to play them, but putting a game in the context of its time period (i.e. it influenced this, popularized this graphical technique, broke this unspoken rule) would help preserve its legacy, or perhaps introduce it to those who didn't know the game in the first place.

In the case of documentation, it can be scanned and retyped, but to really make it resonate with people, the important elements need to be highlighted. The Nth assembly code revision is likely going to be less exciting than a breakthrough design revelation or critical algorithm. Key creator interviews would be an added bonus to any of these scenarios.

Without context, these collections are just anonymous piles of stuff. We don't have scads of critics and curators out there to illuminate why such and such a work is important, like we have in the art world. So in order to popularize and continue the momentum of the preservation, these organizations need spokespersons; arbiters of quality and content for the rest of the world.

If that person does their job well, more collections will be donated, there will be more awareness of preservation efforts in general, and more people will be able to experience our deep and important history. 📺

—Brandon Sheffield  
twitter: @necrosofty

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)

FOR DIGITAL SUBSCRIPTION INFORMATION  
[www.gdmag.com/digital](http://www.gdmag.com/digital)

## EDITORIAL

**PUBLISHER**  
Simon Carless | [scarless@gdmag.com](mailto:scarless@gdmag.com)  
**EDITOR-IN-CHIEF**  
Brandon Sheffield | [bsheffield@gdmag.com](mailto:bsheffield@gdmag.com)  
**PRODUCTION EDITOR**  
Jade Kraus | [jkraus@gdmag.com](mailto:jkraus@gdmag.com)  
**ART DIRECTOR**  
Joseph Mitch | [jmitch@gdmag.com](mailto:jmitch@gdmag.com)  
**DESIGNER**  
Jessica Chan  
**CONTRIBUTING WRITERS**

Tom Curtis  
Jesse Harlin  
Mathew Kumar  
David Hellman  
Peter Drescher  
Steve Theodore  
Kim Pallister  
Soren Johnson  
Matthew Wasteland  
**ADVISORY BOARD**  
Hal Barwood Designer-at-Large  
Mick West Independent  
Brad Bulkley Microsoft  
Clinton Keith Independent  
Brenda Brathwaite Lolapps  
Bijan Forutanpour Sony Online Entertainment  
Mark DeLoura THQ  
Carey Chico Independent  
Mike Acton Insomniac

## ADVERTISING SALES

**GLOBAL SALES DIRECTOR**  
Aaron Murawski e: [amurawski@think-services.com](mailto:amurawski@think-services.com)  
t: 415.947.6227  
**MEDIA ACCOUNT MANAGER**  
John Malik Watson e: [jmwalton@think-services.com](mailto:jmwalton@think-services.com)  
t: 415.947.6224  
**GLOBAL ACCOUNT MANAGER, RECRUITMENT**  
Gina Gross e: [ggross@think-services.com](mailto:ggross@think-services.com)  
t: 415.947.6241  
**GLOBAL ACCOUNT MANAGER, EDUCATION**  
Rafael Vallin e: [rvallin@think-services.com](mailto:rvallin@think-services.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

TYSON ASSOCIATES Elaine Tyson  
e: [Elaine@tysonassociates.com](mailto:Elaine@tysonassociates.com)  
LIST RENTAL Merit Direct LLC  
t: 914.368.1000



UBM

WWW.UBM.COM

# BILZARD®

ENTERTAINMENT



WORLD  
WARCRAFT®



STARCRAFT®



DIABLO®

## BLIZZARD IS HIRING

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

IRVINE, CALIFORNIA | AUSTIN, TEXAS | VELIZY, FRANCE | CORK, IRELAND  
SINGAPORE | SHANGHAI, CHINA | TAIPEI, TAIWAN | SEOUL, SOUTH KOREA  
SAO PAULO, BRAZIL | BUENOS AIRES, ARGENTINA | MEXICO CITY, MEXICO

[jobs.blizzard.com](http://jobs.blizzard.com)



## rise from your grave!

HOBBYISTS CREATE NEW GAMES FOR VINTAGE HARDWARE

While some independent developers have found success making “boutique” titles for smaller markets like Xbox Live Indie Games, WiiWare, or self-distributed computer games, some go even further, making games for the few people still playing vintage consoles! These folks actually manufacture and release their games on discs or cartridges that are playable on the original systems, and the games tend to be labors of love, with the words “profit margin” never crossing their makers’ lips. New “homebrew” games are coming out all the time, but here are five notables that we’ve had our eyes on.

### BATTLE KID 2: MOUNTAIN OF TORMENT

(SIVAK GAMES, NINTENDO ENTERTAINMENT SYSTEM)



2008’s notoriously difficult BATTLE KID: FORTRESS OF PERIL was the first significant platformer released for the NES since it was on shelves, and is still arguably the best homebrew game on the system. Developer Sivak is following up the

inventive original (which is part MEGA MAN, part METROID, and part independent Flash game I WANT TO BE THE GUY) with a sequel that adds more huge bosses, a larger map (638 rooms at press time, versus 550 in the original) and a bunch of new gameplay features and abilities like wall gripping, conveyor belts, and slippery ice. Like the original, the entire game is a solo effort, with Sivak using all his self-taught Assembly language chops to code the game, as well as composing a robust thirty-track soundtrack and drawing all of the graphics. Like the original, this one will be available to buy on an actual cartridge at RetroUSB.com.

### STAR ODYSSEY

(SUPER FIGHTER TEAM, SEGA GENESIS)

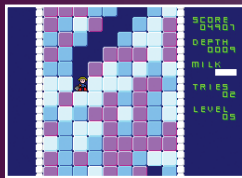


STAR ODYSSEY isn’t a new game so much as it is a new localization. This is the third RPG to be officially translated and released on cartridge by Super Fighter Team for the Sega Genesis. The original game, a turn-based space opera RPG called BLUE ALMANAC, was released

exclusively in Japan in 1991 by long-defunct publisher Hot-B. The game was planned for release in the United States at the time—and was even advertised in magazines—but was ultimately scrapped. In 2007, Super Fighter Team acquired an incomplete (and we’re told quite buggy) build of that English translation and, through a licensing partnership with its current rights holder and a lot of long nights of disassembling the ROM, was able to polish the game into a commercial state. Interestingly, this marks the first time in recent years that a Japanese rights holder has been involved with a re-release for defunct hardware.

### DIGGER CHAN

(AYPOK AND PLAYGENERATION, SEGA MASTER SYSTEM)



DIGGER CHAN is one of the first games to be released for Sega’s 8-bit Master System since about 1998—yes, the system was long dead in most of the world by that point, but it found a niche appeal and cultlike following in Brazil despite alternatives that included Sony’s PlayStation! DIGGER CHAN

brings to mind Namco’s action-puzzler MR. DRILLER. Our hero, DIGGER CHAN himself, must dig down through colorful underground tiles (preferably several matching tiles in a row) and find hidden bottles to fulfill his constant need to drink milk (he’s a milk tap repairman, you see), being sure to avoid nasty traps set by rival dairy companies. A prototype of the game was initially put together as part of a 2006 coding competition at Master System preservation web site SMS Power!, but was polished casually over the last five years into a commercial product, complete with original packaging and clamshell case.

### CALM, MUTE, MOVING

(SONNY RAE TEMPEST, ATARI 2600)



This isn’t really a game: As author Sonny Rae Tempest describes it, CALM, MUTE, MOVING is actually a “game poem”; a brief interactive work of art. The premise sees “players” taking on the role of a blue-collar worker driving during his morning commute, trying to deal with morning traffic while maintaining his cool. Our easily irate driver can steel his nerves by smoking a cigarette. You can do that with Player 2’s fire button just fine, but the game is actually meant

to be played with a special cigarette-like controller that accepts input via the player taking a drag off of a plastic tube. Unlike the other games on this list, CALM, MUTE, MOVING isn’t for sale: Those interested should head over to SonnyRae.com and download the plans to build their own cartridges and cigarette controllers. While you’re there, check out his interactive postmortem, in the form of a text adventure game!

### STURMWIND

(DURANIK AND REDSPOTGAMES, SEGA DREAMCAST)



STURMWIND is actually the fourth posthumous Sega Dreamcast release from German publisher redspotgames, a company founded in 2005 by diehard Dreamcast champion Max Scharl. Past titles include 2009’s RUSH RUSH RALLY RACING, 2007’s puzzle game WIND AND WATER: PUZZLE BATTLES, and 2005’s Neo Geo port LAST HOPE. STURMWIND is a horizontal shoot ‘em-up using a hybrid of pre-rendered 2D and 3D graphics. The game actually started life about six years ago as a demo for another obsolete console, Atari’s 64-bit Jaguar (developer Duranik has also made games for Atari’s Lynx portable and Falcon030 computers), but has since shifted to the more popular Dreamcast. Like any good shooter, the game has a ton of huge boss enemies, a robust power-up system and a rock and roll soundtrack. Initially planned to be released around now, STURMWIND was delayed to a new release date of November 11, 2011—look out, SKYRIM!

—Frank Cifaldi



## michael jackson visits sega of japan

\\ Here's a nice little image that was recently unearthed from the archives, depicting Michael Jackson visiting the Sega offices in Japan in 1988, prior to the release of the Genesis. That fellow showing him around is Mark Cerny of MARBLE MADNESS and SONIC THE HEDGEHOG 2 fame, who was integral to the early success of Sega's big foray into America with STi. The young child with MJ is Jimmy Safechuck, with whom Jackson acted in a Pepsi commercial around that time.

Says Cerny of the visit: "For what it's worth, the monkey stayed in the limo and did not take part in the tour. Not a joke!"

## ascii animator released

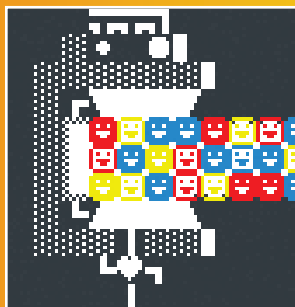
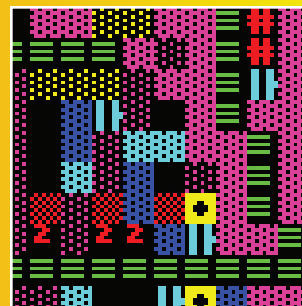
\\ ASCII art has been a staple of the internet for years now, mostly used to illustrate amusing snarky forum posts. But in the early days of computing, ASCII was one of the only ways to get graphics moving on a screen, and thus formed the building blocks of games.

Perhaps unbeknownst to many, a robust ASCII painting program was released last year by Melly of the TIGSource forums (<http://forums.tigsource.com/index.php?topic=14588.0>). But why stop there? Branching from another ASCII painting program (<http://code.google.com/p/ascii-paint/>), another TIGSource forum member, eigenbom, has created a free ASCII animation program. It's still a work in progress, with actively-updated code, but is impressive even in its early days.

One of the best features is an export to gif option. As eigenbom explains, "The animation exporter will slice the canvas up into a grid, and read the frames from left-to-right, top-to-bottom, based on the sprite dimensions you supply, and the number of frames you need. It will probably explode if you give it dodgy numbers."

While it's probably too early to try using this in a game (ASCII images can't be explicitly isolated yet), the future implications certainly spark the imagination.

—Brandon Sheffield



# 20 COMPANIES, SERVICES, GAMES, AND CONCEPTS THAT HAVE CHANGED THE GAME BUSINESS

# CHAN

////////// We work in a rapidly changing creative industry, where trends can rise and fall inside of a year, or move on to become new standards that all shall follow. This kind of rapid growth and change doesn't come from nowhere, though. There are catalysts to every big industry shift. Where would we be without the business and game architecture of D00M? If Facebook hadn't come along, would we be employing thousands of people for MySpace games? What about Unity, or Unreal Engine 3?

We've decided to drill down and look at some of the recent concepts, games,

companies, and services that are changing the game industry, for better or for worse.

## Mojang

(STOCKHOLM, SWEDEN)

\* Much has been said about Mojang and its monster hit MINECRAFT. At over 3 million paid accounts and 10 million users total, the game is a massive financial and critical success. It takes the concept of user-created content to new extremes, making the gameplay and the creation one and the same.

But the reason Mojang makes our list is not just the money. Any company can make

money with underhanded tactics, but Mojang has done so with absolute transparency. For one thing, it proved the viability of the "pay at alpha" model of self-funding. Companies have tried it before, and others have done it since, but MINECRAFT wrote the book on the concept. Essentially, let people pay for something they like as early as possible—but make sure you





# GAME DEVELOPERS

keep supporting them, listening to them, fixing bugs, and being as honest with them as you can. MINECRAFT creator Markus Persson makes most of his announcements via Twitter rather than through press releases, and does his best to answer most emails and comments directly (though that's impossible with 10 million users), which puts Mojang at the forefront of company messaging as well.

On top of that, as the company gets threatened by Bethesda for using SCROLLS as the title of its upcoming game (Bethesda thinks Mojang's SCROLLS sounds too much like its own THE ELDER SCROLLS), Mojang is turning a blind eye to the blatant copies of its game that have cropped up on XBLIG and PC, some of

which have made over a million dollars.

Mojang should be changing the way companies think about the game business. The Swedish company proves you can be honest, transparent, and responsive to your fans, and still make a massive profit.

## Kickstarter (NEW YORK, NY)

\* Kickstarter is likely universally known by readers of *Game Developer*, but on the off-chance there's someone among us who's unaware, Kickstarter is a company that takes donations on behalf of a fledgling (or finishing) project, offering incentives for buyers, and

general goodwill for the company that needs a boost. Though Kickstarter is certainly not the only game in town (see *Game Developer*



issue May 2011, pg. 4), it is the largest, and has funded the most successful game projects to date. Kickstarter takes a small cut of the donations (5%, Amazon takes another 3–5% for use of its payment service), but this is a small price to pay for a company looking for funding. Most of these groups wouldn't get anything otherwise, and the crowd-funding model has

# GAME

turned out to be a big deal for the indie game community in particular. Games such as CTHULHU SAVES THE WORLD and BLADE SYMPHONY got their funding from Kickstarter, and OCTODAD got a sequel due to its successful campaign on the service.

The great thing about crowd-funding versus getting funds from publishers or angel investors is that Kickstarter owns no part of submitted projects, and (for better or for worse) does not hold them accountable for their completion. Most successful projects seem to be nudges to the finish line rather than actual kick-starts, but a publisher-free funding model is a blessing to any independent game developer, and Kickstarter is currently the leading way to make that happen.

## Gameloft (PARIS, FRANCE)

\* Gameloft splits its time between making mobile versions of licensed game properties, like ASSASSIN'S CREED and SPLINTER CELL, original titles like ASPHALT, and blatantly similar titles to popular games like UNCHARTED and POKEMON.

For better or for worse, Gameloft has pushed the envelope when it comes to making



POKEMON (top), CRystal MONSTERS (bottom).

games that draw on the success of other titles. The company makes entirely competent, great-looking games for mobile devices (and occasionally consoles) that leave absolutely no question as to their origin. N.O.V.A.'s enemy, weapon, and environmental designs look suspiciously like those from HALO. SHADOW GUARDIAN borrows themes and gameplay elements from UNCHARTED. CRYSTAL MONSTERS uses the themes, gameplay, and even battle perspectives from POKEMON.

ETERNAL LEGACY calls to mind FINAL FANTASY XIII. STARFRONT: COLLISION does not hide its STARCRRAFT allusions. You don't have to stretch your brain very much to see the similarities.

Now, this isn't stealing, but it is a case of extreme influence. As battles around IP and gameplay concepts rage, Gameloft's studios have managed to consistently skirt the issue. And it seems to be working for them, because whenever a company like Naughty Dog or Nintendo doesn't release a game for

iOS, Gameloft is there to pick up the slack, and make a decent game that scratches a similar itch. While one certainly wonders what the company's design meetings are like, there's no question that Gameloft is changing the business. This is especially interesting when you examine companies like Nintendo, which says it will never release a game on iOS. Gameloft is forcing companies to think about their mobile strategies a bit earlier, before Gameloft decides to think of it for them.

## Rovio (ESPOO, FINLAND)

\* ANGRY BIRDS is not a tremendously innovative game in itself. It's certainly a massive hit,



with more than 42 million downloads as of this writing, but when you break it down, it's nothing more than a standard action physics puzzler using a formula and play-style that has existed for many years. It's not the game that puts Rovio on our list—it's how the company has supported it.

Once it was clear that ANGRY BIRDS was going to be a success, Rovio didn't start planning a sequel, or even a new game. It took the "games as a service" model touted by MMO developers, and shrank it to mobile size. The company has released extra levels, holiday-themed versions, and other updates and upgrades (including some item purchasing) consistently throughout the game's now nearly two-year lifespan. People are still buying ANGRY BIRDS even now, because Rovio knows when people are playing a game, they talk about it. And when people talk, others become interested. Consistently building buzz has been critical for the title, but so has a massive campaign of porting to every device under the sun, including upcoming versions for Nintendo's Wii U and 3DS, as well as older phones and operating systems.

Rovio began as a mobile company doing J2ME games and working from contract to contract, and some of that shows in its porting lust. But the clever bit is that when

they found a hit, they stuck with it, instead of moving on to the next contract. The company also used new platforms to prove new business models (the first version on the Android OS was free-to-play with ads). Is this sustainable? Rovio certainly thinks so, bragging that when it goes public, its IPO will be worth more than PopCap's. This remains to be seen, but the company is doing a fantastic job of pushing ANGRY BIRDS out to as many people as possible, without a huge backlash saying that it's just milking one franchise. And that takes real ingenuity.

## Humble Indie Bundle (INTERNET-BASED)

\* The Humble Indie Bundle was an intriguing experiment—pack several indie games together, and give people a "pay what you want" model for downloading them. The profits were to be divided up among the developers. There had been attempts at models like this before, but not on this scale. The quality of the titles as well as the buzz generated meant that the first bundle went on to generate almost \$1.3 million. Subsequent bundles have done even better, helping all companies involved generate additional income without the bundle claiming any ownership over the products themselves.

One of the project's additional successes comes from its ability to retain that indie feeling while growing massively. As the bundles have gotten more successful, they attracted the attention of investors. Sequoia Capital provided venture funding of \$4.7 million to the bundle's future growth, which is a decidedly un-humble amount of money. Even so, the third indie-branded bundle has surpassed the previous two in sales, and only a minimal amount of ill will has been generated from fans decrying the less-than-indie funding source. So long as the games are indie and the cut the bundle makes remains low, it appears the Bundle will continue changing the way indies look at their own post-release business.

## Microsoft's Kinect (KIRKLAND, WA)

\* The Kinect was Microsoft's answer to the motion-control craze in games that started in earnest with Nintendo's Wii. Through the power of a 3D camera, Kinect makes your full body the controller, and early numbers looked good. Though Microsoft hasn't released any statistics in the last several months, as of March 2011, the peripheral had sold over 10 million units. The Kinect camera was instantly the cheapest 3D camera on the market, and the device was quickly modified by hobbyists for nonstandard

# STARTUPS

use, with early demos showing some amazing technologies, from 3D rendering of a space in real time, to curious visualizers.

It quickly became clear that Kinect was a hit among not only game players but the tech community at large, and if Microsoft didn't get



in front of the bus, the hobbyists were going to drive it away. So in February 2011, MS released a non-commercial SDK for Kinect for PC, and while the third-party market for PC-oriented use has only begun, a great number of impressive strides are already being made. Scripts exist in Google Chrome to control the browser with hand gestures, MotionBuilder is using Kinect for cheap motion capture (as are some hobbyists), and others have found virtual reality game applications for the hardware. Outside of games, Kinect has been used in video surveillance, for trying on new clothes in retail stores, and medical imaging.

Kinect is proof positive that if you provide intelligent people with an affordable and intriguing product, it will take on a life of its own. While the Kinect's greatest success will likely be in games, when our world crosses over into other spheres, even greater things can happen.

## Pixologic's Sculptris (LOS ANGELES, CA)

\* It wasn't too long ago that ZBrush and later Mudbox took the game art world by storm, offering 3D modeling environments that were closer to sculpting than they were to traditional Maya modeling. The high-polygon models couldn't be beat—but for some, the software was too complex and labor-intensive.

And so it was that hobbyist Tomas Pettersson set about developing Sculptris in his spare time in 2009. The software is still in alpha, but already has artists excited, with its simpler user interface and speedier entry into the world of digital sculpting. Though



some call it a "ZBrush lite," the software is now under the guiding hand of the ZBrush company Pixologic, and packs nearly as much



Zynga's  
ADVENTURE WORLD.

power into a more user-friendly package. Artists, indies especially, have gotten excited about the development of the software, which looks to open up the world of digital sculpting to a whole new audience. What's more, it's free to download, though of course Pixologic hopes to transition users into ZBrush and its more robust, deeper toolset, allowing interoperability between both packages.

## Valve's Steam (KIRKLAND, WA)

\* Steam has more than proven itself to be the digital publishing platform of choice for PC games. With more than 30 million users as of this writing, Steam commands a huge chunk of the digital game distribution marketplace. Downloadable console developers of games such as SUPER MEAT BOY have reported making significantly more money on Steam than on consoles, and cross-platform development across console and PC is becoming more common as a result.



Valve's platform has become the de-facto standard for independent game companies looking to publish on PC, and companies such as EA and GameStop have tried to make inroads with their own systems, with Origin and Impulse respectively. Regardless of whether the future of PC games will be fragmented across multiple services, it was Steam that proved the model, and remains a game changer for companies across the industry. Steam continues to update, with Steam Cloud, which allows some storage of game data on a cloud service, built-in DRM solutions (for better or for worse), and Steam Guard, a safeguard against account hacking.

Though Steam has been available for many years, its continued and increasing relevance keeps it on our list.

## Zynga (SAN FRANCISCO, CA)

\* Here we have the 200-pound gorilla of the social space. Zynga is huge, to be sure, with 275 million active players as of September 2011, and more than 2,000 employees—but the company is also leading the social industry on multiple fronts. For its huge corporate anonymity, Zynga has been rather open with its development practices, sharing best practices for web game development at conferences, and discussing the use of social metrics in games.

In terms of its actual games, Zynga has also made big strides when it comes to trying to get the core gamer into the space, with games like EMPIRES & ALLIES and ADVENTURE WORLD. Others have made inroads, to be sure, but it's nice to see when a larger company doesn't play it totally safe. Zynga also runs Zynga.org, a charity outlet that has donated thousands to worthy causes, based on in-game item purchases. Though some question the Machiavellian tinge to Zynga's practices, it is doing some good while up there at the top.

## Apple's iOS (CUPERTINO, CA)

\* Though the revolution came some time ago, Apple deserves to make our first list of game changers for iTunes, and its supported iOS platforms. Since their inception, the iPhone, iPad, and iPod Touch have collectively become a major force in the game industry, and a (relatively) cohesive platform in their own right.



Apple's devices have not only skyrocketed the company to the top of the technological heap, they have also launched the careers and assured the fortunes of a great many independent developers. Apple's 70/30%

revenue-share has become the industry standard, and the platform shows no sign of slowing down.

While Apple hasn't put as much focus on facilitating games on its home computers, many expect some manner of convergence across iOS in the near future.

## Cloud gaming services

\* Developers in general seem to agree that cloud-based gaming is an important step in the advancement of the digital medium. It can free players from having to keep their PCs or consoles up-to-date, and could pose a platform-agnostic model for game development. But at present, there are two major players vying for the biggest slice of the pie—OnLive (Palo Alto, CA) and Gaikai (Orange County, CA).

OnLive is currently banking on its physical device, the OnLive Game System, which streams game content to a TV-connected box that allows the direct use of controllers. So far, more than 50 companies have signed on with OnLive, which uses proprietary chips to display its content.

Gaikai, on the other hand, is a browser-based service, with no specific game console, and the ability to embed in web sites. Many leading games are already available on the service.

Whether one of these companies wins or loses is not the point—the game changer is the cloud service itself, which frees consumers from console cycles, game-based PC upgrades, and in some cases, installation or downloading of software. Gaming on the cloud is not a totally proven model yet, as the servers are quite expensive, but as costs go down, prospects certainly look up.

## Mozilla/Khronos Group's WebGL

\* As the next generation of web content starts to become a reality, 3D games in browsers become more common. And for that to happen, we needed something better than standard Java. Mozilla's WebGL, among other 3D web libraries, has risen to fill that need.

Though WebGL is far from perfect, the fact that it provides a 3D graphics API without the use of plug-ins is an extremely important step toward 3D games in the browser. There are some competitors out there, but at the moment, WebGL is the (slightly fickle) darling of the browser game development community. As the library expands and best practices start to emerge, trends indicate that we'll be playing a lot more



plugin-free 3D games in our browsers than ever before, further reducing the barrier to entry for players. And who can argue with that?

## Depth Analysis's MotionScan

(SYDNEY, AUSTRALIA)

\* LA NOIRE may have shuttered a studio (Team Bondi) and not fully pleased its parent (Rockstar Games) in the sales department, but it also brought us one heck of a piece of tech. Depth Analysis's 32 high-definition camera setup allowed full capture of all aspects of actors' faces, mapping them to their digital counterparts for an incredibly lifelike performance. Though the characters



were clearly still made in game engines, the animation was a major breakthrough. Since the game hoped to allow players to gauge whether characters were telling the truth, precise performance was incredibly important.

After many years of R&D, Depth Analysis's work appears to have paid off, as the performances in LA NOIRE have been majorly lauded. The company claims its setup can capture up to 50 minutes of final footage, processing up to 20 minutes of facial animation automatically per day. This technology is available only from Depth Analysis so far, but now that the technique has been illuminated, it's likely that others will follow. The only problem now is that with such lifelike facial animation, the rest of the computer-generated body begins to look even less realistic by comparison—but that's a problem for another day.

## Google's productivity services

(MOUNTAIN VIEW, CA)

\* Google's Android platform is currently the only serious contender to iOS in the smartphone game space, and has shipped on millions of devices, yielding massive sales for some of the developers on the platform. Even Sony is using it for its upcoming tablets, and a set-top box is in the works to serve games to televisions like a standard game console. But in terms of

game development, Google has arguably made an even greater impact in free collaboration software. While Google Docs may not be the perfect place to keep that game design document or store spreadsheets, it's free, and certainly useful in the prototyping phase.

The company continues to push the envelope in the collaboration, and though some may decry the fact that through data mining, their users are their product, few can deny the services' usefulness.

Looking forward, Google is making good strides with its Native Client solution. The intent is to get ARM native code running safely in browsers, allowing web programs to run at near-native speeds. The implications on this for browser-based games are pretty clear. Faster is (almost) always better!

## Web development pipelines

\* Though much of this is still in its infancy, integration of browser tech into game development pipelines is looking to be a big deal in the near future. Some companies, like Insomniac, are building their own solutions, integrating browsers with their engine for things like level editing.

Other companies have begun using cascading style sheets 3D transforms to build UI and HUDs even in non-browser games—and let's not forget client-side storage solutions.

In some cases, groups like Fabric Engine are so convinced of the future of web pipelines that they've built their entire business around it. Though a lot of the current tech is primarily for building web applications, even companies like Blizzard have found uses for the web in their more traditional pipelines. Expect to see more of this as the years wear on.

## Riot Games

(SANTA MONICA, CA)

\* Though the microtransactions model has been proven in Korea for years, it had some difficulty making inroads with the core gamer in the U.S. and Europe. More and more games from Western developers have been adopting the model, but LEAGUE OF LEGENDS is truly knocking it out of the park. The game uses intelligent microtransaction-based game design that doesn't make players feel like they're playing a partial game if they don't pay, and gives those that do pay something significant to crow about.

As an online player-versus-player game, LOL has also been intelligently built for competitive play, which has given the game extra legs in other countries. The game is one of a handful to be brought to China, and distributor Tencent Holdings went so far as to



purchase a majority stake in Riot because of its success.

The game's smart design, its democratic moderation system, and metagame overlay exemplify the future of Western free-to-play game development—and some might say, the PC game industry as a whole.

## Indie Fund

\* The indie fund was put together by a host of indie game development notables, including Jon Blow (BRAID), Ron Carmel and Kyle Gabler (WORLD OF GOD), Kellee Santiago (FLOWER), and others. Its aim is to “fix a system which never worked,” that is to say the relationship between indies and publishers. The fund currently supports four announced projects; Steph Thirion’s FARAWAY, Dan Pinchbeck’s DEAR ESTHER, Andy Schatz’s MONACO, and Toxic Games’s Q.U.B.E.

The fund’s overarching goal is to help products that are markedly different from the norm come to release. As the fund says on its official site, “We make smaller investments and ask for less in return. The hope is that developers see enough revenue from their game to self-fund their next project. And voilà, one more developer that is free to make whatever crazy game they want.”

The fund promises a flexible budget with no milestones, proportional repayment based on the amount borrowed, and no long-term obligations if the game fails to make its money back. And it’s debatable whether the funded games would even be possible without this financial backing. It’s an interesting experiment, and the fund seems like a model to watch as the games start to roll out and developers give feedback about their experiences.

## Patent litigation

\* Patent lawsuits appear to be here to stay, and they’re definitely changing the face of games. It feels like every week someone is crawling out of the woodwork to sue Sony, Nintendo, Microsoft, and any other company they can think of for violation of their patent for “moving objects on a digital screen.”

The U.S. has a particularly litigious culture, and it was perhaps a matter of time before greedy eyes turned toward the game industry, but we can’t help but decry most of these patents as mere cash-grabs. Many of these suits came from outside the industry—engineers here or there who saw fit to patent an algorithm, technique, or process. For example, in recent months, Nintendo was sued over its Wii Remote design, with ThinkOptical, maker of the “Wavit Remote” saying that the Wii as a whole violates its patent entitled “Electronic equipment for handheld vision based absolute pointing system.”

Peripheral industries aside, we’ve started to see lawsuits from within as well, especially those which question reuse of code in subsequent games. Some of these lawsuits are legitimate misuses of intellectual property, but many more are simply posturing matches between companies looking to stake out their marketing arenas. Unfortunately, the only people that win in these cases are the lawyers, and quite often the shady engineers with their vague patents. That means less money for game development, and less money for developers, and certainly changes the industry for the worse.

## Mobile social platforms

\* OpenFeint was the dominant mobile social platform on iOS upon launch, serving achievements and persistent leaderboards across multiple games. But then Apple came out with its own solution, Game Center.

Now OpenFeint exists for Android as well, while DeNA has taken control of most of the Japanese market’s standard platforms.

While all these companies and platforms duke it out for first place, it’s clear that this is an important area. Players want achievements, and they want social elements in their games, and that’s the real game changer here. Mobile games, even a few years ago, were largely solitary experiences, but that has changed completely with the advent of these sorts of platforms.

Will one solution rule them all, or will the market fragment? As long as consumers’ interests are served, and there some interoperability for developers, the revolution matters more than who’s fighting.

## The U.S. Supreme Court

(WASHINGTON, D.C.)

\* In a landmark decision, the U.S. Supreme Court ruled this year that video games are protected under the auspices of the First Amendment, striking down a California law that would have banned the sale of violent video games to minors. As the official ruling said, “Video games qualify for First Amendment protection. Like protected books, plays, and movies, they communicate ideas through familiar literary devices and features distinctive to the medium.”



The court also said, “Psychological studies purporting to show a connection between exposure to violent video games and harmful effects on children do not prove that such exposure causes minors to act aggressively. Any demonstrated effects are both small and indistinguishable from effects produced by other media.”

While the ruling may seem obvious to those of us in the industry, one can only imagine what might have happened had the court ruled in the other direction. This is a critical point in the future creative freedom of the game industry, and can be used to good ends when the inevitable future cases of video game censorship crop up. Until a new evil comes along to steal the hearts and minds of America’s youth, as movies, heavy metal, and comic books did before them, video games will continue to need solid defense in the courts. And the First Amendment is pretty much the best thing we could ask for. ☺

**BRANDON SHEFFIELD** is editor-in-chief of Game Developer magazine. He fully understands the internet’s obsession with cats.



# UNREAL TECHNOLOGY NEWS



## UNREAL ENGINE 3 POWERS TRIPLE-A, FROM BLOCKBUSTER RPG TO PUZZLE PLATFORMING

The *Mass Effect* franchise is a behemoth in video game entertainment, a series that has literally reshaped the way players think about gaming and its immersive possibilities. The upcoming third installment, along with its predecessors, are all built on Unreal Engine 3.

*Mass Effect 3* will feature not only improved environments and cinematics, but also Kinect integration.

"So much has changed since we began working on this series," said Casey Hudson, executive producer at BioWare Edmonton. "When we began, the Xbox 360 hadn't even come out yet, and yet we had to design a game for it. Now, looking back, we've been working with Unreal Engine 3 for quite a few years. Even with *Mass Effect 3*, we've been able to find huge new improvements to the engine's performance."

And performance is key, according to Hudson. "That's allowed us to do everything from getting much better acting with the characters to introducing better storytelling methods. In addition, the engine allowed us to create a richer world and produce more



from Papo & Yo

entertaining cinematics. "We've also utilized the improved performance of the engine to bring more enemies onscreen at once, so players will contend with a lot more stuff happening in this game."

Gameplay has also changed dramatically, said Jesse Houston, producer of *Mass Effect 3*. "If you look at *Mass Effect 1* to *Mass Effect 3*, they're almost completely different games at this point. You've seen major changes in combat. You've seen major changes in role-playing elements. You've seen massive lighting changes; now we have real graphic fidelity that is just so much better than it has been historically. You're going to see performance improvements. We're 30 frames per second yet again, locked across the board. You can really feel the difference in the controls."

"We're ultimately an Unreal Engine 3 game, and Unreal Kismet and Unreal Matinee are a major part of any kind of cinematic experience," said Houston. "Our team has been able to utilize Kismet and Matinee to create Hollywood-style cinematics that bring the story to life and enhance the gameplay experience." *Mass Effect 3* will be released for Xbox 360®, PlayStation®3 and PC in early 2012.

And while Unreal Engine 3 is the choice among many established developers, independent studios like Montreal's Minority also rely on its power.

Minority's upcoming PlayStation®Network exclusive 3D puzzle-platforming adventure, *Papo & Yo*, has already snagged impressive critical accolades, including six awards and 20 award nominations at this year's E3, including both IGN and GameSpot's Best Puzzle Game of E3. GameSpot stated that "its deeply personal subject matter, clever implantation of puzzles, and surreal art design combined to make something unique and engaging."

Minority's choice to go with Unreal Engine 3 had to do with not only the team's great track record with the engine but also a need for agility throughout development.

Julien Barnoin, lead engineer at Minority, explained, "We knew Unreal Engine 3 would get us creating gameplay mechanics and levels very quickly. We wanted to quickly start building gameplay elements and puzzles and iterating on them. UE3's material editor allowed us to achieve beautiful characters and environments without a lot of work."

"When artists or designers come to me asking for a new feature, I can often just point them to how to do it right in the editor, and can get back to coding the features that are really unique to our game."

*Papo & Yo* is scheduled for release on PlayStation Network in 2012.

WWW.UNREAL.COM



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

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

## UPCOMING EPIC ATTENDED EVENTS

**GDC Online**  
Austin, TX  
October 10-13, 2011

**Montreal Int'l Game  
Summit**  
Montreal, Quebec  
November 1-2, 2011



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

a new way of thinking  
about coding with Rx

# reactive game archi tecture s

////////// One of the exciting things about working with computers is that they provide endless opportunities for working smarter rather than harder. These opportunities have led to evolutions in game development tools, game engine architectures, and game programming paradigms. Most of these advances have been focused on modeling what happens in games rather than describing when those things happen. Consider how you would express the following rules to a computer:

- Earn a power-up after three consecutive deaths without scoring.
- Throw fireballs by pressing a button after rocking the joystick forward.
- Gain extra points by killing enemies within a short time after collecting a power-up.

Describing when rules like these take effect is often more difficult than describing their impact. Reactive programming is a programming paradigm that focuses on how programs react to change. This change might come in the form of new user input, the passage of time, or assignments to internal variables. When changes like these trigger reactions in code, those reactions often result in further changes that cascade into subsequent reactions. In a sense, changes flow through our programs. This is what the reactive programming paradigm encourages us to model. Instead of treating events as discrete isolated moments, reactive programming is focused on modeling and processing streams of interdependent events.

Reactive programming isn't new, but it has primarily been used with functional programming

>>>



Imperative Architecture



Event Driven Architecture



Reactive Architecture

ILLUSTRATION BY JUAN RAMIREZ

languages in the past. Traces of these functional roots are evident in Microsoft's new Reactive Extensions (Rx) library, even though it's available for popular object-oriented and hybrid languages like C#, JavaScript, F#, and Visual Basic. After an introduction to reactive programming using Rx with C#, we'll reflect on some of the key benefits of reactive programming for game development. In particular, we'll look for better ways of expressing when things happen in games.

### Reactive Programming with Rx

/// In Microsoft's words, "Rx is a library to compose asynchronous and event-based programs using observable collections and LINQ-style query operators." Observable collections serve as the foundation of this library. The `IObservable` interface represents a stream of event notification messages that follow the well-documented "Observer Design Pattern" used in many object-oriented applications. Of the many ways to create new observables, here is one of the simplest:

```
IObservable<long> whenSecondsElapse = Observable.Interval(
    TimeSpan.FromSeconds( 1 ) );
```

This simple observable creates a new event every second. The payload with each of these events is a simple sequence number. There are many ways to define observers (that implement the `IObserver` interface) to process these events. The following example simply passes every new `whenSecondsElapse` event to the `WriteLine` method to be printed on

the screen. There is also a sleep instruction to prevent this asynchronous observable from being prematurely terminated. We'll further discuss this asynchronous nature of observables below.

```
whenSecondsElapse.Subscribe( Console.WriteLine );
Thread.Sleep( Timeout.Infinite );
```

Subjects seem to provide one of the most general shortcuts for converting existing game events into observable streams. They are observables that you can post new events to at any time. Every time an event is posted to the following subject, it will be printed to the screen with the prefix "RX".

```
Subject<string> whenNewInputArrives = new Subject<string>();
whenNewInputArrives.Subscribe( newInput => Console.WriteLine(
    "RX> " + newInput ) );
```

We can now post new event messages to this subject at any time using the subject's `OnNext` method. Here's some code that simply echoes the user's input back to the screen by posting it to the `whenNewInputArrives` subject.

```
while ( true )
{
    Console.Write( "YOU> " );
    string input = Console.ReadLine();
    whenNewInputArrives.OnNext( input );
}
```



# MAKE YOUR GAMES MORE ENGAGING, ADDICTIVE AND PROFITABLE



**2011**  
**STORYWORLD**  
conference + expo  
san francisco • oct 31-nov 2

Learn directly from the game developers and other media pros who are using transmedia techniques to get their fans fully hooked! At StoryWorld Conference + Expo, major creative minds like ARG pioneer **Steve Peters of Fourth Wall Studios** (*Six Minutes to Midnight*; *Why So Serious*; *Year Zero*; and *Dead Man's Tale*) will show you how to avoid potential pitfalls and succeed at cross-platform entertainment.

Use your exclusive Game Developer discount code GDSW11 to save \$50 off full conference registration.



**SIGN UP NOW AT [STORYWORLDCONFERENCE.COM](http://STORYWORLDCONFERENCE.COM)**

**THE EDUCATION EXPERTS IN  
GAMES, 3D & VFX**

**NEW US CAMPUSES  
OPENING AUGUST 2011!**

**[www.theaie.us](http://www.theaie.us)**

As one of the first educators in the world to offer games-specific qualifications way back in 1996, the Academy of Interactive Entertainment knows how to get graduates into games, animation and VFX.

More Info: 206-428-6350 OR 225-288-5221



**AIE** ACADEMY OF  
INTERACTIVE ENTERTAINMENT

When an observable event stream ends or errors out, all of its observers are notified and unsubscribed. To demonstrate this, we can add a conditional statement to the loop above that ends the event stream as soon as the user enters "STOP." After you type the word "STOP," your input will stop being echoed to the screen through the `whenNewInputArrives` observable.

```
if ( input == "STOP" ) whenNewInputArrives.OnCompleted();
```

## Observable Operations

Now that we can create simple observables, let's look at some of the high-level operations we can perform on them. Processing and composing observable streams represent an expressive new way of describing when things happen in our games. The three basic kinds of operations that we'll look at include filtering, projecting, and combining.

For the purpose of this article, we'll make use of four subjects created in a simple XNA game, which you can download the assets for at [www.gdmag.com/resources](http://www.gdmag.com/resources). There are static `keyPress` and `keyRelease` subjects that all `GameObjects` share access to, and non-static `collisionEnter` and `collisionExit` subjects within each `GameObject`.

## Filtering

Filtering allows us to create new observables that contain only a subset of the events from another observable. Consider creating an observable of firing commands by filtering only enter key presses out of the general `keyPresses` subject. The `Where` method is used below to create a new observable that is filtered in this way.

```
IObservable<Keys> whenToShoot = keyPresses.Where( key => key == Keys.Enter );
```

We can also filter these events by any number of other criteria, such as whether there is actually ammo available for the player to shoot when this key is pressed.

```
IObservable<Keys> whenToShoot = keyPresses  
.Where( key => key == Keys.Enter && ammoCount > 0 );
```

Once we've created a suitable observable, we can subscribe our bullet-shooting method to observe it.

```
whenToShoot.Subscribe( fireKey => shoot() );
```

## Projecting

Projection involves mapping the payloads in an event stream to other values. For example, we might want to map the left- and right-key press events to  $-0.05$  and  $+0.05$  radians of rotation respectively. After filtering the key press stream down to these two events, the `Select` method below performs this projection.

```
IObservable<float> whenToSteer = keyPresses  
.Where( key => key == Keys.Left || key == Keys.Right )  
.Select( key => key == Keys.Left ? -0.05f : +0.05f );  
whenToSteer.Subscribe( steer );
```

The `keyPresses` observable above contains only a single event as each key is pressed. If we want our steering handling code to be called repeatedly while a key is being held down, we can project each key-press event into a stream of multiple events that don't stop until that same key is released. `SelectMany` projects each individual event into a new observable stream

of events. The observable it returns is composed of all of the events from each of those projected streams merged together.

```
IObservable<Keys> keyHolds = keyPresses.SelectMany( keyDown =>  
Observable.Interval( TimeSpan.FromMilliseconds( 10 ) )  
.Select( tick => keyDown )  
.TakeUntil( keyReleases.Where( keyUp => keyUp ==  
keyDown ) ) );
```

To break this composition down, every `keyPresses` event creates a new observable using `SelectMany`. That observable creates a new event every 10 milliseconds. The payload of these ten millisecond events is projected to the value of the key that is being held down using `Select`. Finally, these projected observables are terminated as soon as the held key is released. The `TakeUntil` method is taking care of this termination.

The resulting `keyHolds` observable can be used to capture events every 10 milliseconds for as long as any key is being held down. You can replace the `keyPresses` observable in the steering example above with this new `keyHolds` observable to implement more continuous steering controls. This same `keyHolds` observable can also be used to add thrust to propel our ship forward for as long as the up arrow key is held down.

```
IObservable<Keys> whenToThrust = keyHolds.Where( key => key ==  
Keys.Up );  
whenToThrust.Subscribe( thrustKey => thrust() );
```

This ability to compose and reuse new observable event streams presents a breakthrough in more expressively describing when things happen in a game.

## Combining

`Rx` contains a large collection of observable stream-processing methods. There are many more methods available than I could hope to cover in this article. I can, however, demonstrate a couple patterns that appear more generally useful in processing game events. I'll describe these patterns as serial and concurrent event combiners.

The serial combiner responds to specific sequences of serial events. As an example, this might be used to implement a combo attack that is triggered only by a specific sequence of inputs. The observable defined below will only trigger a combo attack after the user presses the sequence: up, down, up, enter.

```
IObservable<Keys> whenToFireCombo = keyPresses.Where( key =>  
key == Keys.Up ).SelectMany(  
keyPresses.Take( 1 ).Where( key => key == Keys.Down  
) .SelectMany(  
keyPresses.Take( 1 ).Where( key => key == Keys.Enter )));  
whenToFireCombo.Subscribe( combo => fireCombo() );
```

In this example, the `SelectMany` method cascades new observables to check each subsequent serial requirement. Whenever the next key entered does not match the required sequence, the nested observables are terminated. The only events that the resulting `whenToFireCombo` observable returns occur when the final enter press in this sequence is matched.

The concurrent combiner responds to events that are occurring at the same time. As an example, consider picking up ammo only when the player is both colliding with an ammo pickup and pressing the space bar at the same time. The `Join` method provides a convenient way of detecting overlap between events with duration.

```
IObservable<GameObject> whenToPickup = Observable.Join(
    collisionEnter.Where( ammo => ammo is Ammo ),
    keyPresses.Where( key => key == Keys.Space ),
    ammoEnter => collisionExit.Where( ammoExit => ammoExit ==
ammoEnter ),
    keyDown => keyReleases.Where( keyUp => keyUp == keyDown ),
    ( ammo, key ) => ammo );
whenToPickup.Subscribe( pickupAmmo );
```

In the `Join` method call above, the first and third parameters describe when collision events start and end. The second and fourth parameters describe when the enter key is pressed and released. Whenever an ammo collision overlaps with an enter-key press, an event is spawned with its payload defined by the fifth parameter of this method: the specific ammo object that the player is colliding with.

## Scheduling and Disposing of Subscriptions

/// Our examples so far have neglected a couple somewhat clerical concerns. First, where and when is the code that is asynchronously processing observable event streams running? And second, what resources need to be released when observers are destroyed?

Rx uses schedulers to encapsulate work that needs to be done, and the context it should be done in, with a clock to help regulate when that work is done. All the observables we've worked with thus far include default schedulers. Microsoft has chosen the schedulers to introduce the least amount of concurrency reasonable for the observable type. The `ObserveOn` method allows you to change the scheduler that an observable is running through. In addition to the `Immediate`, `CurrentThread`, `NewThread`, `ThreadPool`, `TaskPool`, and `Dispatcher` schedulers that are provided with Rx, you can create your own `IScheduler` implementation for more complete control. The example code includes a scheduler that performs work within XNA's `GameComponent` framework. Here's an example of subscribing to an observable using that custom scheduler.

```
whenToShoot.ObserveOn( rxGame.scheduler ).Subscribe( shootKey
=> shoot() );
```

Rx uses disposables to model observer subscriptions that can be terminated at any time. Whenever an observable sequence is completed or errors out, all related subscriptions are automatically disposed of. However, when an observer wishes to stop receiving event notifications from a live observable stream, they must manually dispose of their subscription. This disposable subscription is returned from the `Subscribe` method, and its `Dispose` method prevents further events from being pushed to the observer. The `CompositeDisposable` is a convenient collection type that allows groups of disposable subscriptions to be disposed of together. The following code fragments add a new subscription to a `CompositeDisposable` collection called `subscriptions`, and then disposes of all of the subscriptions in that collection.

```
subscriptions.Add( whenToShoot.Subscribe( shootKey => shoot()
) );
subscriptions.Dispose();
```

## Syntactic Sugar

/// The Rx team is fond of pointing out the duality between `IObservables` and `IEnumerables`. Pulling data from an `IEnumerable` collection is symmetric to having data pushed to you from an `IObservable`. This observation appears to be the inspiration behind integrating LINQ and Rx. LINQ is Microsoft's Language Integrated Query syntax, and has traditionally

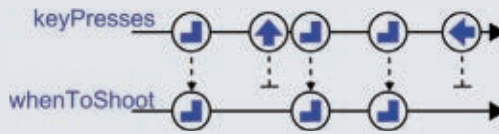


FIGURE 1 Filtering enter key presses to trigger shooting.

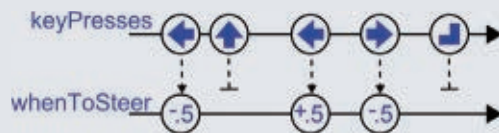


FIGURE 2 Projecting left and right key presses into +/- .5 steering angles.

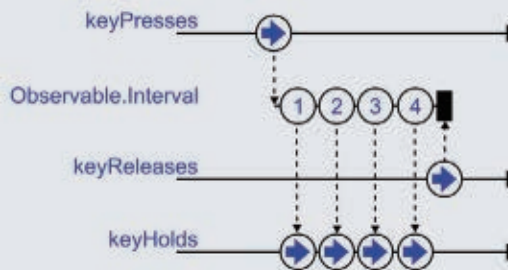


FIGURE 3 Projecting key pressing events into multiple key holding events.

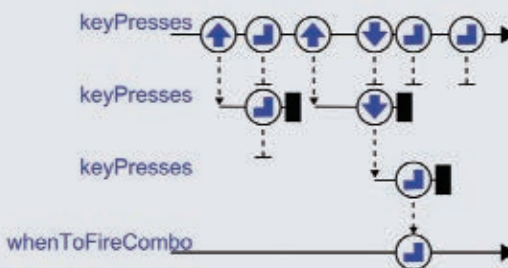


FIGURE 4 Cascading observables that detect Up, Down, Enter serial key combinations.

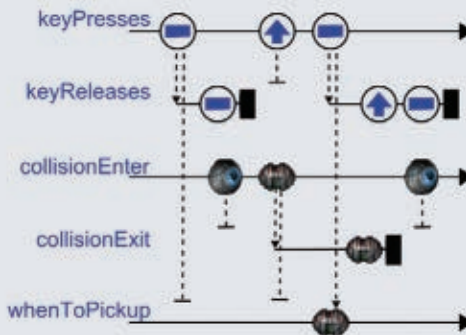


FIGURE 5 Using join to detect concurrent space bar down and ammo collision combinations.



been used to query data from object hierarchies, xml files, and databases. It can now also be used to query data from observable event streams in Rx. Here's a quick LINQ example of the steering observable discussed above.

```
var whenToSteerContinuous = from key in keyHolds
    where key == Keys.Left || key == Keys.Right
    select key == Keys.Left ? -0.05f : +0.05f ;
```

Although I don't personally find this syntactic sugar particularly advantageous, I do believe the underlying relationship between querying databases and event streams is important. The shift from thinking about events as hookable moments in time to streams of data that can be processed and queried is at the heart of what reactive programming has to offer game developers.

## A Brief History of When

/// The thing that excites me most about reactive programming in games is the increased focus on expressively describing when things happen in games. This different way of thinking about and organizing code is much more subtle than any new feature on a graphics card or in a game engine. In order to better frame this advantage, I'd like to briefly review how descriptions of when things happen are commonly implemented in games.

Code is generally executed from top to bottom in imperative architectures. When something happens it is implicitly determined by the relative position of an instruction. Basic control structures give us ways of conditionally skipping over some instructions or repeatedly executing them, but generally avoid directly addressing the question of when things should happen in a game. These implied expressions of when things should happen are powerful enough to implement any Turing-compatible game feature or algorithm, but are extremely messy and difficult to maintain.

Modern games and simulations are commonly organized around a main loop that runs multiple tens of iterations per second. This loop contains instructions implementing every possible change within a game. However, many of these instructions may be preceded by a condition that contains a true or false expression of whether those instructions should be run or skipped over. This condition is repeatedly evaluated within a loop, and serves as an explicit description of when something should happen in a game. Organizationally, this architecture is still quite a mess to deal with, as the line between event-detecting and event-handling code is so blurred.

Event-driven architectures break descriptions of what happens in a game apart from when those things happen. Many modern game engines expose similar sets of event handlers for programmers to implement. The convenience of implementing these event handlers without worrying about event detection is clearly limited to the set of events that an engine exposes. Extending these sets of events is often so cumbersome that programmers commonly filter overly general events instead. For instance, a general update or tick event that is repeatedly triggered may contain multiple conditions checking for events related to each player's health, the elapsed time of a race, and a submersed player's depth. Detecting game-specific events in this way leads event-driven architectures to essentially degenerate into loop-style now or not-now expressions of when.

There's a variation of event-driven architectures that carries an important advantage in terms of describing when things happen in games. Although this terminology is not completely standard, I prefer describing systems that serialize event messages for later processing as message-based architectures. This is in contrast to event-driven systems that integrate event-handling directly into the flow of a program's execution with constructs like delegates, virtual functions, or function pointers.

Message-based architectures' subtle shift from collecting event-handling code to event-notification data adds a valuable level of

indirection. Event notification messages can be sorted, filtered, or even deferred for later processing. Beyond helping us more expressively describe when things happen in games, message-based architectures also offer benefits in the implementation of game save and playback features, networking synchronization, and powerful diagnostic tools for developers.

## The Future of When

/// Reactive game architectures are built on top of multiple observable event streams, each of which individually resemble message-based architectures. In addition to enabling us to sort, filter, and defer the processing of individual event messages, reactive architectures encourage us to compose entirely new observables. In the previous example code, we composed a `keyHolds` observable relative to the pre-existing `keyPresses` and `keyReleases` observables. We also created new observables that represented both serial and concurrent combinations of events from other observable event streams.

Although reactive programming is an important step in the evolution of describing when things happen in games, I believe that we are still near the beginning of this journey. Many experienced programmers are so accustomed to implicitly describing when things happen in games that it's easy to overlook the problem and accompanying opportunities for improvement. I'm not sure that I would have been as sensitive to this problem if it weren't for my experience helping students figure out how to convert polled key inputs into throttled fire commands. The process of identifying mutable state to track and check is just not very intuitive to beginning programmers. In fact, even more experienced programmers often prefer to avoid such messy mutable states, as can be seen in the design of more functional programming languages like Scheme, Haskell, and F#.

Reactive libraries like Rx are still a long way from fulfilling their exciting potential. There are currently more than a hundred operations available on observables. Many of these operations are redundant, and it's not entirely clear what subset of them will form a sufficiently expressive basis. There may also be better ways of expressing common high-level processes like the serial and concurrent combiners discussed above. As a final note, Rx represents a paradigm shift in processing observable event streams. This requires time, both for programmers to become familiar with, and for tool developers to engineer specialized diagnostic and debugging environments around.

## Final Reactions

/// Expressively describing when things happen in games is hard. We have evolved from using imperative and event-driven architectures to message-based, and now reactive game architectures. But none of these are as expressive or intuitive in formulating descriptions of when something happens as in English ... yet. The composable event streams of reactive architectures appear to be a promising step forward, but only time and experience will tell.

Beyond introducing you to reactive programming and Microsoft's Rx, I hope this article encourages you to experiment with new ways of explicitly describing when things happen in your games. It's easy to take what we are used to for granted. But resisting that complacency is essential to advancing the state of our art. This is the programmer's journey, to search for new ways to work smarter rather than harder. 🎮

**GARY DAHL** is a game programmer, designer, teacher, student, and player. He has been teaching game development skills at Brown College for the last five years, and also develops indie games through <http://sugarpillstudios.com>.



learn / network / inspire

**GAME DEVELOPERS CONFERENCE<sup>®</sup>**

SAN FRANCISCO, CA  
MARCH 5-9, 2012  
EXPO DATES: MARCH 7-9

**2012**

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



# bulletstorm

postmortem

What could go wrong if your first project is an unassuming old-school PC shooter, and your second project is a big, multiplatform AAA title? Why would things be different if you grew from 15 to 70 employees in a couple of years? How does it feel to go from e-mail interviews to standing in front of the entire world as it watches your live E3 presentation? Yeah, this could be a book. Ten thousand things went right and ten thousand things went wrong during the production of BULLETSTORM. Obviously there's not enough space to talk about them all, so this article is a very subjective selection, mostly filtered through the design side. It took three and a half years to make BULLETSTORM. We've learned, we've grown, and we hope you'll find something for yourself in this story as well. >>>



**PUBLISHER** Electronic Arts  
**DEVELOPER** People Can Fly/Epic Games  
**NUMBER OF DEVELOPERS** Around 80 at PCF  
**LENGTH OF DEVELOPMENT** 3.5 years  
**RELEASE DATE** February 22, 2011  
**SOFTWARE** Unreal Engine 3, 3ds Max, Maya, Photoshop, Modo, ZBrush, Motion Builder, MS Office  
**PLATFORM** Xbox 360, PlayStation 3, PC

# bulletstorm



## WHAT WENT RIGHT

### 1 focus on core combat loop.

/// Our philosophy is that it's better to have a great game that's just about spitting than a mediocre game about spitting, screaming, and playing the banjo.

If you have ever sighed at a game's attempt to offer variety (e.g., not every game needs a driving section), you already know what I mean.

Also, stop thinking about that banjo game. It's not going to happen.

We spent months polishing our core combat loop: shooting, kicking, sliding, leashing. A millisecond here, a 1% more transparent particle there. Improve, playtest to death, rinse, repeat. We knew it all worked together well when we started missing elements of our CCL while playing other games. I cannot count the amount of times I wanted to kick or slide into an enemy when enjoying a competitor's FPS.

Of course, modern games cannot rely on the CCL alone, but we also don't think that having myriad core features is the right solution. You will never have the time to polish them all properly.

We decided to focus only on the crucial ingredients, get them to 100%, and have the gameplay variety come from a wide palette of contexts to the CCL. For example, the kick is always the same thing—the same animation, the same sound, the same button—but its purpose can change depending on what the player needs to do. Kick to push an enemy away. Kick to destroy an enemy's armor. Kick to open a door.

### 2 pacing and balancing pass.

/// As with most games, BULLETSTORM was built by multiple level designers, each working on their own fragment of the game. At a certain point we felt that every level had really great pacing: a good warm-up, varied encounters, and an interesting cliffhanger.

Most of the levels were done with "big picture" pacing in mind. For example, the Cave level started slow, because the ending of the preceding

level was a six-minute boss fight, and we wanted to let players relax a little.

But as all game designers know, theory is one thing, and the actual implementation is another. When we put the game together, some pacing problems immediately crawled out of the woodwork. The Cave's relaxed opening was a good idea, but lasted far too long. The Underground and Ulysses sections were great on their own, but both exclusively featured one type of enemy (Burnouts), and that was tiring in the long run. The Dam level felt too long, even though in isolation it was one of the best levels in the game.

We made pacing and balancing our priority. We removed and added battles. We shortened cinematics. We fine-tuned all the values for ...everything.

In the end, we got a really well-paced game. Unfortunately, it was a little too well-paced for its own good. Wait, what?!

Imagine a shooter or a horror game that takes eight hours to finish, but where the experience is so intense that you only play one to three hours daily. It means it will take you three to five days to finish the game, which is how we imagined our game would be.

However, with BULLETSTORM, we have seen time and time again that people got so engaged in the experience that they finished the game in one day. It was still eight hours, but played in only one or two sessions. That allows us to enjoy the fact that over 50% of players finished BULLETSTORM, which is higher than the industry's standard (this is easily trackable by checking "Finished the game" achievements in services like Raptr—see References for more). But at the same time, opinions started floating around that the game was short.

It was not short by current standards. But to gamers, two four-hour sessions are not the same as five hour-and-a-half sessions.

We are still proud of the pacing and balancing work, but intensity versus time is something we'll have to think about in future projects.

### 3 finding the fun through early focus tests.

/// We never planned to offer a unique gameplay hook. We just wanted to evolve the genre a tiny bit. Maybe we'd do it by unmuting the main hero, even though





it's a first-person game. Maybe we'd differentiate by making sure that the sidekicks are not automatons spewing context-sensitive comments, but actual people we can have strong feelings about. Our intentions were nothing more than that. We just wanted to offer a fun high adventure—that's it.

However, the aforementioned focus on the core combat loop, constant iterations, and—most of all—internal focus tests led to the invention of Skillshots.

I am a big fan of games that offer multiple pseudo-independent systems. That is the core of any emergent gameplay. BULLESTORM has a few systems like that: multifunctional weapons (e.g., the flail chain can wrap around enemies or objects), an interactive environment (e.g., the explosive trash cans of the future can be moved around) and the tools of war (kicking, sliding, leashing).

Players can manipulate and use these systems any way they want. That leads to a lot of emergent gameplay moments. You can kick an enemy into a trash can, which then explodes, which causes another enemy to go airborne and get pierced by a rocket fired by another enemy positioned on the rooftop.

But I would never have thought of naming these crazy actions—and indeed, I would never have thought of Skillshots at all—if not for internal focus tests.

To some developers, the focus test sounds like something to dread. They think: "There's a reason why I'm the creator, and they're the consumers." That's fair enough, and there is some truth to it; but on the other hand, there's also a reason why Blizzard wins so big after spending insane amounts of time playtesting and fine-tuning.

Our early focus tests were nothing official. We were just watching each other play. In casual conversations we debated what was fun and what sucked.

But it was during these monitored play sessions that I noticed people were using BULLESTORM's emergent combat not to be efficient, but to have fun. A headshot is a much faster kill than leashing an enemy toward you, then kicking him into a cactus. And yet still people kept on doing that, and even much more elaborate things. They were experimenting. Testing theories. Having fun.

"Hey, why don't we actually reward people for being creative?" I thought. "How about we call it a Skillshot system?" And one of our core selling points was born.

#### 4 many companies, one game.

/// You can imagine the fragmentation challenges we faced with PeopleCanFly as the main developer and creative owner, Epic as co-developer and quality enforcer, and EA as the publisher. And there were a number of additional companies from all around the world helping us as well (in Germany, USA, China, Sweden, and Poland).

This stuff is not for the faint of heart.

Oddly enough, coordinating the outsourcing was not the hardest part. With good, dedicated producers in place, it's actually something I am sure to repeat in the future.

It was the PCF/Epic/EA cooperation that caused the most problems. Everybody was equally important, and everybody could influence any part of the development process: design, production, and so forth. For example, both EA and Epic were giving us independent feedback on the milestones and playable builds. It was a mess.

Fortunately, we all noticed that very quickly. We understood that the key to fluid cooperation is a proper distribution of roles and responsibilities. In other words, sometimes you have to let go.

We immediately streamlined all these processes. Using the feedback example, EA was no longer sending feedback to PCF but to Epic. Epic then merged EA's feedback with its own, empowered to remove whatever they disagreed with, and then sent one unified chunk of feedback to PCF. This way we only had to deal with one source of feedback, were no longer prey to multiple masters, and stopped worrying about the priorities of the feedback items.

If you are working with several partners, letting go is the key. One of you knows more about the marketing than the other, another is a better

# bulletstorm



judge of quality. Clearly define who is responsible for what, and make life easier for everyone.

## 5 unreal engine 3.

/// Ha! What a surprise: a guy from a company owned by Epic, creator of Unreal Engine, praising the Unreal Engine. Good one!

In 2004, we released PAINKILLER. The game took two years to make, with a 15-person team (on the average). We had a long single-player campaign, and a fun multiplayer mode that was a CPL (Cyberathlete Professional League) game of the year. And we did all that using our own engine, which we created from scratch.

PAINKILLER was really an extremely simple game at its core, though, and yet still we struggled. We had to outsource cinematics in order to finish the game on time, for example. If it had been any more complicated—and I am not talking about big stuff like going multiplatform, I merely mean things like featuring a sidekick—the game would never have happened.

Instead of merely listing the benefits of Unreal Engine—something you either already know or can easily google—let me just tell you three facts from PCF's past.

First, we decided to switch from our own engine to Unreal well before we had any idea that one day we might be a part of Epic. It was worth it to hear Mark Rein scream, "Finally!" into the phone. Second, from the moment we got the engine, it took us only a month to prepare a demo for publishers—a demo that Epic saw and said "Hold on for a second there, let's talk." Third, it took us two months to make two levels of BULLETSTORM for a pitch demo that we showed in Leipzig to various publishers, and which got us a deal with EA. I have absolutely no idea how we could have achieved that without UE3.

## WHAT WENT WRONG

### 1 creative f-bombs.

/// BULLETSTORM was often critiqued for its seemingly endless stream of four-letter words.

Do you know any swear word in a foreign language? German, French, Polish? When you say it out loud, no biggie, right? Not a problem to use it during a family dinner, I assume?

That is how all the f-bombs sounded to us. Being Polish, all the strong language in BULLETSTORM was just exotic and fun to us. We did not feel its power. In other words, Epic thought this is what we wanted and respected our creative vision, while we had no idea this vision was a bit more than we really wanted.

It was only at the end of the development, when I read the Polish translation of the game, that I realized how dirty we were. I swear a lot. A LOT. Yet still I ... kind of blushed.

But, to be honest, the language would not have been a problem if not for its creative usage. There are games that use way more f-bombs than BULLETSTORM, and yet somehow it's less of a problem for them. Why?

That's because most of the time the language is forgettable. If you hear a character say "You f\*\*\*\*\* scared me, you a\*\*\*\*\*!" you forget about it two seconds later. Although, if he says "You scared the d\*\*\* off me!" it stays with you for a little bit longer, which creates the impression that the game is much fouler than it really is.

We tried to solve that by adding the language filter. Originally we forced the players to make a choice before they even got to the main menu. With best intentions to make the experience more fluid we moved the option to one of the submenus.

Big mistake. No one noticed it existed. It cost us a few prestigious reviews and a sea of tweets from angry gamers.

Next time, do it like BRUTAL LEGEND: a forced choice, during the game, right before the first f-bomb. Personally, just in case, the game should ask the players once more some time later.

### 2 slow beginning, debatable ending.

/// I am not a fan of action-packed openings to video games. You do get players' attention, even if the attempt reeks of desperation, but the inevitable lull after

the intro is over is a pacing killer. I think the only time I have seen a hi-octane opening work well was GOD OF WAR, but that's just because Kratos doesn't take a break, and the opening is actually the entire game.

But, by Crom, that does not mean your game can take 45 minutes before you get to the core gameplay! Which is exactly what BULLETSTORM did.

Endings are even more important. That 2–5% of the game can increase the review score if it's awesome, and kill the other 95–98% if it sucks. A bad ending is like finishing a delicious dinner only to learn there were worms in it.

Our ending is decent, but we were just trying to be too clever. We wanted the players make a certain mental choice. Bad things start happening in main character Grayson Hunt's life when revenge becomes the only thing he cares about. At the end of the game, he can choose to continue walking down that path, or let go. What would YOU do? Do you think Grayson has learned anything?

But that was a bit fuzzy, and many people treated the lack of a clear resolution and closure—even though we destroyed an entire planet—as a blatant attempt to trick them into buying the sequel.

Guess what? Throughout the entire development time we were perfectly aware how crucial the beginning and ending of a game were, and still we got both of them wrong.

Be triple careful and assign an overkill amount of time and resources to making sure that the first and last page of your book are perfect.

### 3 wrong choice for the online mode.

/// I'm going to tell you a secret. We had a working, playable player-versus-player mode in BULLETSTORM. It was only a basic deathmatch, and it was a mere prototype, but it was playable—and it was tons of fun.

But we felt that the PvP space was too crowded. Ignoring the fact that our PvP was unlike anything else out there (thanks to the Skillshot system you could win with fewer kills than the opposite team), we felt we needed something different. That's how the Anarchy co-op mode was born.

That, in itself, was not a bad choice. The problem was that we decided on Anarchy too late in the development process. We managed to make the mode really fun for the hardcore, advanced BULLETSTORM players, but everyone else struggled. Anarchy requires tight cooperation, and is not bulletproof. If you don't work together, the mode is just not fun. That's quite unlike PvP, where most of the time both teams and individuals can enjoy the carnage.

You only have one chance to make the first impression. It's better to release a fun and polished bare minimum effort than an unfinished experiment, no matter how unique.

### 4 lack of context in the demo.

/// In the game industry we often debate whether releasing a demo makes sense. It's a controversial topic. But you know what's not controversial? Bad demos.

I wouldn't say our demo was textbook "bad," but it just wasn't the right demo. Demos should be about emotions, about the atmosphere, about the vibe of the game. Remember the BATMAN: ARKHAM ASYLUM demo? Perfection.

With the BULLETSTORM demo, we wanted to show people how fun the Skillshot system was. We wanted the players to fight for the best score, and thus replay the demo over and over again. So we took a very short, storyless fragment of one of the levels, stripped it of any story-related dialogue, and focused the gameplay flow on core mechanics. Also, because our mechanics were new in the FPS world, we added a tutorial movie.

EEEEK. Wrong.

In the full game, we take it slowly. We still have elements of the tutorial two hours into the adventure. Why did we expect that people would learn all that from a three-minute movie? Why did we expect people to watch these movies anyway?

Gamers were confused. So is this a time-attack, arcade-style kind of game? There's no story, right? How can such a shamelessly short demo be so boring? All I had to do to finish it was kick, kick, kick!

We should have chosen the opposite: a story-driven "blockbuster" fragment of the game, focused on mystery and visual appeal. The BULLETSTORM demo did get two million downloads across Xbox 360 and PS3 in two weeks' time, which is a good success metric, but we needed to make sure that the demo really communicated what we wanted it to.

### 5 echo mode: too little, too late.

/// At PCF, we keep saying that Echo mode—a sort of trials and challenges spin on the Skillshot system that ranks players based on creativity and clearing stages—was created by gamers and journalists.

Our E3 demo was focused on the gameplay and core mechanics: we knew there was no way anyone would be able to appreciate the story with a thousand other games around fighting for attention with speakers larger than a fridge.

The one silly thing we did in the E3 demo—displaying the amount of earned skill points—turned out to be a big deal. People stayed in the demo room longer just to see how others performed. They were taking photos of their score. Even the crew held an internal competition between waves of visitors. Then, when we introduced Echoes via the demo, it spawned thousands of homemade YouTube videos.

We had never planned on making an arcade experience for BULLETSTORM, but what happened at E3 could not be ignored. We decided to embrace the score hunt and added a special gameplay mode with unlocks, stars, and leaderboards—the whole shebang.

It was very risky, adding a new, big gameplay mode six months before going gold. But we did it. We didn't get it quite right, however. There were too many Echoes (maps) to make competition meaningful, and those maps didn't offer new content because they were simply locations from the campaign.

Players could win vanity items in Echo mode, but not tangible gameplay rewards, so the persistence layer was lacking. We also didn't have time to implement leaderboards such as Best of the Week or Best in the Area. The list goes on.

The lesson here is that the time needed to put a feature into the game does not equal the time of implementation. It's possible to strike gold and have your first iteration be the last, but most of the time that's not the case.

We addressed Echoes-related issues in the DLC packs by adding completely new maps made from scratch, but it was too little, too late. We're grateful that the mode took off in popularity so quickly, and the fact that it helped round out BULLETSTORM's gameplay, but we were unable to invest enough time to flesh it out.

references

Statistics on how many people finish games:  
[http://edition.cnn.com/2011/TECH/gaming.gadgets/08/17/finishing.videogames.snow/index.html?hpt=te\\_t1](http://edition.cnn.com/2011/TECH/gaming.gadgets/08/17/finishing.videogames.snow/index.html?hpt=te_t1)

### I BELIEVE!

/// BULLETSTORM is bittersweet for us. As a new IP, it sold well over a million units, which is amazing when you consider the craziness of 2011, but disappointing if you remember that everyone expected more. It suffers from a catch-22: it would have been very profitable if it had taken less than three and a half years to create, but it would not have been the same game if we did not use all that time to make it shine.

But, as Mike Capps, the boss of Epic, said, "It was worth it." Through hard work and passion we have become a ninja team ready for any zombie apocalypse. We helped improve the Unreal Engine, adding features and improving its multiplatform support. We made a game that got praise from the most prestigious, toughest magazines in the world, and that has become a lasting memory for many gamers.

Thank you, EA, and thank you, Epic. You kick ass, PCF. We're ready for the next round. @

ADRIAN CHMIELARZ is creative director of People Can Fly.



## THE 2<sup>ND</sup> ANNUAL



The Game Developers Choice Online Awards is the premier award ceremony for peer-recognition in the connected games industry. Taking place annually during GDC Online, the Choice Online Awards recognize and celebrate the creativity, ingenuity and innovation of the finest online developers and games created in the last year.

### AWARDS ARE PRESENTED IN THE FOLLOWING CATEGORIES:

#### 2011 AWARD CATEGORIES

- Audience Award
- Best Audio for an Online Game
- Best Community Relations
- Best Live Game
- Best New Online Game
- Best Online Game Design
- Best Online Visual Arts
- Best Online Technology
- Best Social Network Game
- Online Innovation Award

#### SPECIAL AWARDS CATEGORIES

- Online Game Legend
- Hall of Fame

Award finalists will be announced in August.  
Stay updated at [www.gdconlineawards.com](http://www.gdconlineawards.com)

PLATINUM SPONSOR  
**edgecast**

PRESENTED BY  
**GDC Online**

PRODUCED AND HOSTED BY  
**GAMASUTRA**  
The Art & Science of Making Games

**gd** GAME DEVELOPER  
MAGAZINE





REVIEW BY DAVID HELLMAN

COREL CORP.

# Painter 12

**Art is great.** But cleaning up is boring. I used to paint with oils, and I ruined a lot of brushes by not rinsing them thoroughly enough in the turpentine. Cleaning solvents are corrosive and smell terrible. Thankfully, there is a world encased in glass, aluminum, and plastic, where no one ever has to clean up. In this world, mistakes can be undone with a keystroke, “save as” dissolves the anxiety of ruining a long-nurtured masterpiece, and a wealth of editing tools speed up experimentation.

Yes, painting in a computer is much cleaner, faster, and more convenient than painting in reality. But you trade away the texture and dynamism of the pigment. You trade away expressiveness. Real-life media can be assertive collaborators. Watercolor is challenging, but in capable hands, its subtlety and richness enable poetry.

Enter Corel Painter. In the sterile world of digital painting, Painter enables an unusual amount of poetry. Faithful, intricate simulation of real-world media is Painter’s purpose, which it attacks deeply and broadly. In Painter, oil has depth; it builds up on the canvas. There’s a light source reflecting off it, bringing the paths of individual bristles into relief. Marker strokes have the variable fuzziness, bleed,



and saturation jitter that tell you they’re real. There’s an entire category of tools for blending and smudging your work, ranging from a pointed stump to something called “just add water.” You can even simulate wind blowing across the canvas.

It’s still no match for physical media, and the various effects are not as convincing. In Painter

however, you will hear the pigment speaking back to you more than in any other app. It’s spunky. It’s invigorating. And it’s fairly robust.

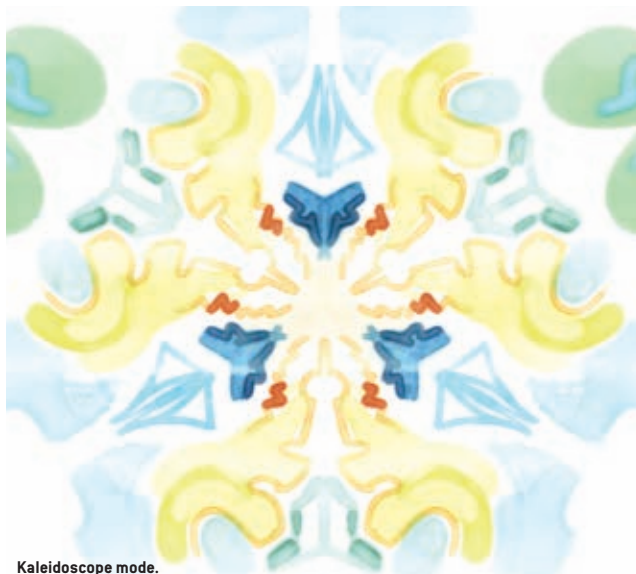
Available tools include, but are not limited to: airbrushes, chalk, charcoal, gouache, acrylic, palette knives, pens, pencil, and sumi-e. Each of these are categories with a dozen or more items within them. Painter 12 expands the toolset with new alternative oil and watercolor effects, called Real Wet Oil and Real Watercolor. These are really wet in the sense that they bleed, ooze, and otherwise transform after application.

Corel has added some cute stuff, too. Kaleidoscope Mode duplicates your painting across three axes in real time, producing an effect deserving of its name. It’s immediately hypnotic.

With so many tools and features, it’s a good idea to experiment for a while. Every brush comes with a giant control panel for adjusting every variable of its performance. Patient tinkerers are sure to find just the right thing.

When you launch Painter, you set foot into a labyrinthine digital art laboratory. Every corridor reveals another dripping, splattering creation—all blessedly raster-based, no poncho needed. Dials and knobs are everywhere and you’re invited to push and twist whatever you like: You are thrilled.

After visiting for a while, though, you admit to yourself that certain amenities have been neglected. With all the inventing and the late nights, nobody has tidied up—the interface is cluttered. All the customization you’ve been



Kaleidoscope mode.

**COREL CORP.**  
**Painter 12**

1600 Carling Avenue  
Ottawa, Ontario  
K1Z 8R7  
Canada  
www.corel.com

**PRICE**

- > Full \$429
- > Upgrade \$229

**SYSTEM REQUIREMENTS**

- > Microsoft Windows 7, Windows Vista (32-bit or 64-bit editions), or Windows XP (32-bit edition), with the latest Service Packs installed, 1 GHz processor or higher, 1 GB RAM, 600 MB hard disk space for application files, 1280 x 800 screen resolution, Mouse or tablet, DVD drive, Internet Explorer 7 or higher,
- > Mac OS X 10.5 or 10.6 (with latest revision), Intel Core Duo, 1 GB RAM, 300 MB of HD space for application files, 1280 x 800 screen resolution, Mouse or tablet

**PROS**

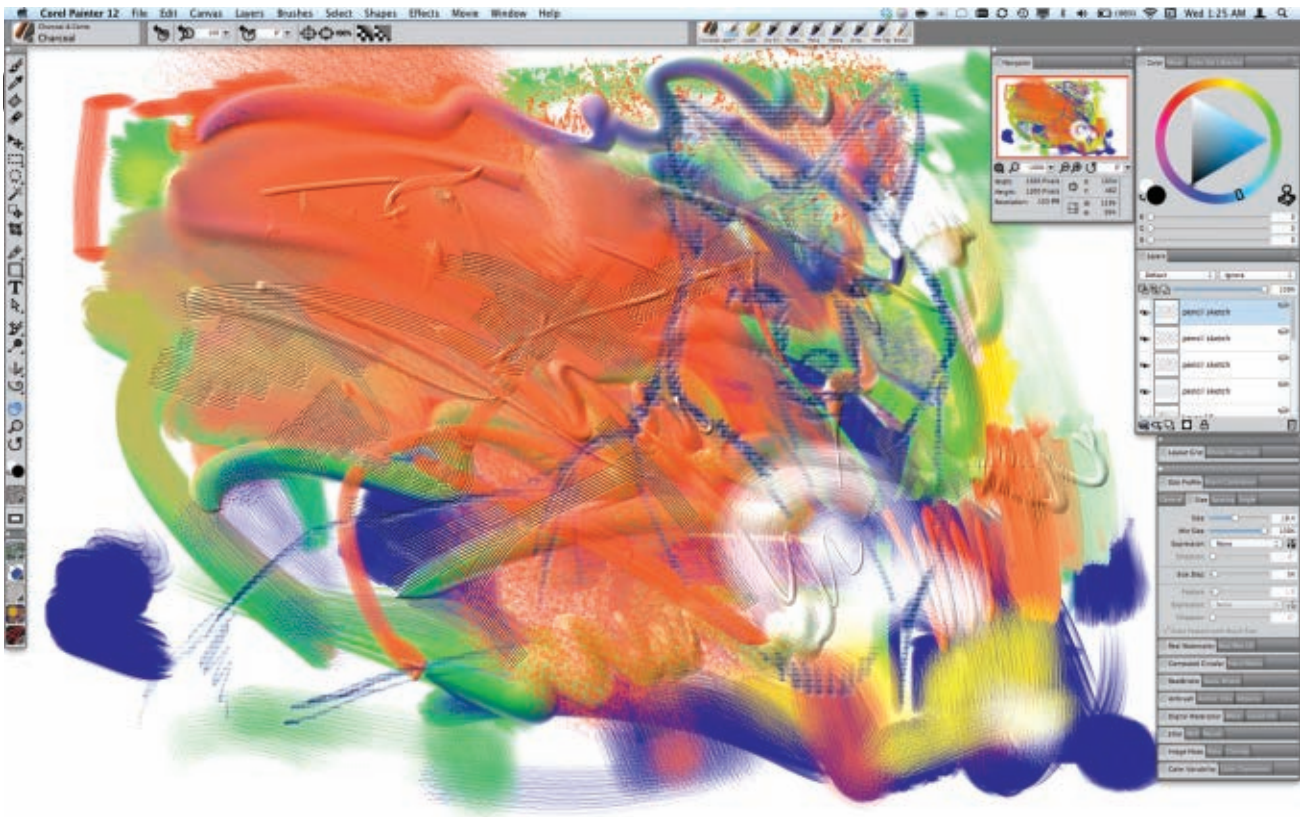
- 1 Dynamic real-media simulations
- 2 Huge range of tools
- 3 Customizable brush behavior

**CONS**

- 1 Inconsistent performance
- 2 Fiddly UI
- 3 Too much clutter

invited to do doesn’t quite excuse the condition of things.

Many brushes work great, but some perform so slowly that I found them unusable. It’s said that Cézanne would sometimes contemplate a canvas for hours between strokes. He would have had no problem, then, with the Real Wet Bristle brush (of the Real Watercolor set). During its post-stroke dynamic drying action, the app slows down so badly that



even the cursor stutters around the screen. God help you if you are painting dappled light on the surface of a lake. The rhythm imposed by this brush, and many others, is stroke—wait—stroke—wait. On the other hand, the Light Fringe brush performed quickly for me. It's up to you to figure out which brushes work acceptably on your system, and under what conditions. But do expect lag.

The UI suffers from the same unevenness. Painter 12 does attempt some improvements over

previous versions, but obvious problems remain. Brushes have been regrouped to make them easier to find, and yet the three types of watercolor categories are scattered around. Why? Because their names are Watercolor, Real Watercolor, and Digital Watercolor, and the menu is arranged alphabetically—even though it's viewed via icons!

Another example: There's now a toolbar of recently used brushes, so you can revert quickly, but many of the thumbnails look exactly the

same, and their names, which are displayed in a very tiny font, are truncated for space. Your best bet is to mouse over and wait for the overlay to appear. In short, the toolbar that's supposed to make brush selection faster is itself slow to use.

Many users will take these awkward points in stride. If you're a long-time user of Painter, you know what to expect and will be pleased with upgrades like the new Navigator panel, which displays a thumbnail of the entire canvas.

But I've been using pro apps for more than 15 years, and those years have made me a less—not more—tolerant person. I want the power and flexibility of Painter, but I want a clean interface, too. I want the most-used controls to be big, and the less-used things to go away until I need them, especially because this is an art app. All those buttons and sliders are like barnacles taking over my canvas.

Look at SketchBook Pro. Yes, it's a rubber ducky facing the Corel Painter battleship, but look at the approach to UI, at the brush

resize puck. It's 88 x 88 pixels big! It's easy to grab, which is good because you use it a lot. As you click-and-drag, an outline continuously scales so you can see how big your brush will be, relative to the zoom level of your painting. Back to Painter: The brush resize control is an 11x16 pixel rectangle, though "rectangle" makes it sound nicer than it is. There's also no visual feedback. (I don't count the slider itself as visual feedback because the value range is mapped unevenly.)

At the end of the day, artists will choose Painter 12 because it has the best simulation of real-world media. Digital watercolors that bleed and blend on your screen are a fantastic trick, and a boon to artists pursuing naturalistic effects on a tight schedule. For now, there's no app that does what Painter does, and certainly not as elegantly. 🐼

**DAVID HELLMAN** is a computer-based artist living in San Francisco. He produced the art for the game *BRAID*, the web cartoon "Jeff and Casey Time," and the comic "A Lesson Is Learned But the Damage Is Irreversible."



## product news

### Idea Fabrik Announces HeroCloud Middleware Bundle

/// HeroEngine developer Idea Fabrik recently announced the HeroCloud bundle, a proprietary development platform that includes middleware from five additional tool providers.

The HeroCloud platform includes Idea Fabrik's own HeroEngine (originally created by Simutronics), as well as modeling and animation technology from RAD Game Tools, the FMOD audio suite, SpeedTree, Singular Inversions' FaceGen, and the performance optimization tool Umbra.

Idea Fabrik says the HeroCloud license is available to developers for no up-front cost, though it uses a revenue sharing model in which Idea Fabrik receives a 30 percent cut of net revenue for providing the bundle.

"Our goal has always been to help game developers find ways to make better looking, better sounding, better playing games, and with the addition of these critical technologies, we continue to advance our mission—to remove the barriers that stand in the way of creative game development," says Idea Fabrik COO Neil Harris.

Earlier this year, Idea Fabrik established its own game development studio in northern Virginia, and introduced Idea System, a development and business package for 3D social games.

—TOM CURTIS

### OpenFeint SDK Adds New Cross-Promotional Social Newsfeed

/// Mobile social game network OpenFeint has announced a new social game newsfeed for cross-promotional updates. It's called

GameFeed, and it is available now in the OpenFeint SDK.

GameFeed's newsfeed provides players with real-time updates across OpenFeint's network of games, providing social information that includes updated player profiles and information from developers.

According to the company, 24 beta participants saw sessions-per-user increase by an average of 25 percent, with some showing growth as high as 60 percent.

"Sessions per user is a key engagement component of lifetime value and GameFeed significantly increases engagement at no additional cost to developers," said OpenFeint founder and CEO Jason Citron in a statement.

—FRANK CIFALDI

### iSwifter Adds Google+ Support For Flash Games On iPad

/// Cloud-based content streaming platform iSwifter has added Google+ game support to its iPad platform.

iSwifter 4.1, available now, allows players to access Flash-based Google+ games, in addition to its previous Facebook games implementation.

iSwifter launched just over a year ago as a cloud-based streaming service for iPad. Similar to OnLive or Gaikai, iSwifter streams the games over the cloud, bypassing the iPad's lack of Flash support.

—FRANK CIFALDI

### Sony 3D Headset Coming To Japan This November

/// Sony has announced that it will launch a head-mounted visor later this year in Japan, which will be capable of HD and 3D visuals through a pair of OLED screens.

The "HMZ-T1" device will feature two 720p displays in the headset, with both 2D and 3D high-definition visuals available, and 5.1 surround sound.

The headset will also be able to output its video and audio signals to a television via HDMI output, allowing users



to view the headset content on another display.

The available view given by the two OLED displays will cover a 45 degree area, and is designed to block out the surrounding areas and keep the viewer fully immersed.

Sony says that the headset will be compatible with the PlayStation 3, allowing users to connect it up with their console and view games through the visor.

The headset will be released on November 11, and will cost around ¥60,000 (\$783).

—MIKE ROSE

### Sony Reveals More Details On First PlayStation-Certified Tablets

/// Sony has confirmed the price and release details for its Sony Tablet S, which will hit stores this fall starting at \$499. It will be the first tablet able to legally download and play select games from the original PlayStation library.

First revealed back in April, the Android 3.0-powered tablet sports a 9.4-inch touch display and an NVIDIA Tegra 2 graphics chip, as well as Wi-Fi and 3G connectivity options. The \$499 model comes with 16 GB of built-in memory, while a \$599 model comes packed with 32 GB.

Sony didn't provide pricing details for the previously-revealed Sony Tablet P, a foldable tablet with two 5.5-inch touchscreens and 4G network

capability set for a November release. Each screen can be controlled independently by Android apps, with one screen providing a PlayStation-style controller interface and the other showing gameplay, for example.

The tablets will be Sony's second and third PlayStation-certified mobile devices, after the Spring launch of Sony Ericsson's button-sporting Xperia Play smartphone.

—KYLE ORLAND

### Crytek Releases Free CryEngine 3 SDK, Explains Commercial Revshare Model

/// CRYSIS series developer Crytek has released a free downloadable SDK for its proprietary CryEngine 3 engine, and has further outlined its revenue share model for commercial applications. Since its release last August, as of press time, the free CryEngine 3 SDK has been downloaded more than 300,000 times.

The engine is free to use for non-commercial purposes, but those wishing to use the engine for commercial products will have to enter into a royalty-only licensing agreement that sees Crytek receiving 20 percent of the game's revenues.

"This SDK contains more toys than we've ever released before—it empowers people to create whole new games from scratch, not just mod Crytek's own games, so we encourage all aspiring and indie developers to try it out," said Crytek's Carl Jones in a statement.

According to prior statements by CEO Cevat Yerli, this free SDK contains all of the features of its commercial version, saying that it is "the same engine we use internally, the same engine we give to our licensees, the same engine that powers CRYSIS 2." The SDK is available to download now at Crydev.net.

—FRANK CIFALDI



# THE IRON FIS

AN INTERVIEW  
WITH THE  
DIRECTOR  
KATSU  
HARADA

////////// TEKKEN is a seminal entry in the pantheon of early 3D fighting games, and one of the only series still running, alongside VIRTUA FIGHTER and the somewhat newer SOUL CALIBUR. While other fighters opted for a six-button setup using light, medium, and fierce punches and kicks respectively (like STREET FIGHTER), or a four-button strength-based system (like THE KING OF FIGHTERS), TEKKEN used a limb-based system, where buttons corresponded to left punch, right punch, left kick, right kick. The early days of 3D in games are intriguing, to say the least, as companies scrimped and saved every bit and byte while trying to deliver the maximum visual bang for their buck. As Capcom and Namco prepare to release separate STREET FIGHTER/TEKKEN mashups, we spoke with TEKKEN director Katsuhiro Harada, who has worked on their series since its inception, about the origins of TEKKEN's systems, its unique aesthetic, and its nearly nonsensical story.

**Brandon Sheffield:** *In the original TEKKEN, what was the thinking behind a "four-limb"-based button system as opposed to a strength-and-intensity-based one?*

**Katsuhiro Harada:** Well, before TEKKEN, Namco had been conducting a lot of R&D into polygon-based graphics, and the consensus we had was that true-to-life animation was going to become a huge aspect of game graphics going into the future. That turned out to be very true, of course. The limb-based control scheme sort of grew from that,



# THE DON ST

INTERVIEW  
TEKKEN  
DIRECTOR  
YUJIRO  
KAWADA



but the scheme also felt really good for a fighter—you could execute these one-two punch combos really quickly and intuitively.

**BS:** Right, the R&D was a big deal—when I talked with Yu Suzuki a few months ago, he mentioned that in those early days everyone was in this race to create the most realistic 3D graphics possible. Was the extra complexity a part of the decision, too—having four buttons instead of the three that VIRTUA FIGHTER had? [VIRTUA FIGHTER was a year old in 1994 when TEKKEN was released, and only had punch, kick, and block.]

**KH:** Certainly we were constantly aiming to improve ourselves, trying to avoid doing things that were already done before. One thing with VF was that the action your character took depended on which side he was facing—his moves were flipped depending on whether he was facing left or right. Including all the animation data to make that function is a ton of work, and required what, at the time, was a lot of data. The control system for the first TEKKEN was an attempt to avoid that requirement and just execute the same moves no matter which side the fighter's facing—that, in turn, freed up more

storage space for things like extra characters and moves.

**BS:** Most previous fighters were working along the lines of light/medium/fierce—both in movement and in thought process. What was the thinking behind moving away from that?

**KH:** I think the light/medium/fierce system is a pretty good way to represent fighting in a video game, but it's hard to define which moves go into which "level" of force. It gets even harder with certain fighting styles, because how can you define what a "weak" sumo or capoeira move is? That's why we concluded

# IRON FIST



Towards the end of the 20th century, a mysterious challenge went out to all who were deserving. No one knows from where it originated, but it's draw was irresistible. The promise of power and riches beyond those attainable by the common man all to go to the survivor of the Iron Fist Tournament. From the depths of the lower earth to the far reaches of the outlands the call escaped no one. Many perished during the mortal street fights that were only a preview of the challenges that were to come. Only 8 remain, those with the true killer instinct, all ready to die, to gain the title of

**"Tekken, Lord of the Rave War."**



that the light/medium/fierce system wasn't really appropriate for a game trying to encompass a really large variety of martial arts like what we were aiming for.

**BS:** *Where do you draw the line between fantasy and reality in this genre? To some degree you're going for realism, but flash is important too. Capoeira, for example, would not really be a powerful style to use in fights like this, but it looks super cool.*

**KH:** We certainly realize that "realism," as defined within the bounds of this game, can be a very different thing from what would happen in real life.

*Rocky*, for example, is a film that a lot of people liked—but that was a work of fiction, and real-life boxers would never use really flashy moves like that in an actual match. It's not "real" realism so much as "wouldn't it be nice if things were like this" realism. That's what we aim for here, this manga-or movie-like atmosphere that has impact upon the observer. We try to portray what people expect of the reality they see within games.

**BS:** *You have to build in those kind of dramatic moments—having the final hit replayed at the end of the match, and so on.*

**KH:** Right.

**BS:** *I feel TEKKEN has its own universe of physics, definitely different from reality, and from other games as well. You punch someone and they're immediately off their feet and horizontal, or flip around in the air and so on. How did you come up with this type of physicality?*

**KH:** That flipping around in the air thing is actually something that I devote a lot of close attention to. We want that manga-like atmosphere, to recreate the wide range of expression possible in manga. There's an older manga called *Ashita no Joe*, and that's one I draw a lot from.

**BS:** *Was it part of the plan to make getting hit as dramatic as hitting, so to speak?*

**KH:** Definitely. You want it to feel as dramatic and exciting as possible for the person attacking, to make him feel like, "Yes!" when he lands something.

**BS:** *Talking about animation—I talked with Yoshinori Ono from Capcom about this—in a fighting game, speed is one of the most important things. When a player hits a button, they need that impact immediately. Disney animation emphasizes the setup over the impact, but fighting games have to take the opposite approach. There's this delicate balance between finishing animations and making sure the next move comes on time. TEKKEN seems to animate a bit more and for longer than other series, so I wanted your take on the button-to-move gap.*

**KH:** The TEKKEN series treats both the feeling of speed and the feeling of exhilaration seriously, which is part of why you can throw people into the air so easily—it feels lighter than it should, and it's fun. Another thing we treat importantly, though, is the feeling of weight that the attacks give off. That goes back to what I said about taking damage—the animation's done to make that feel powerful, and emphasizing a certain fluidity to the animation contributes to that feeling of weight or power, like you're really winding up your kicks and other techniques. TEKKEN doesn't have much of a frame buffer between button presses and on-screen attacks, but we try

to use what we have to create that feeling as much as possible. It's a balance, like you say.

**BS:** *As a fighter progresses, additional complexity kind of gets layered in. New players coming to TEKKEN might be intimidated by this. How do you think about this? STREET FIGHTER IV did a good job with trials and bringing people back, and the ARCADE EDITION has a little achievement-style crawl like KING OF FIGHTERS 13. Even with that, though, it's very hard if you don't know the basic inputs.*

**KH:** That's an important topic for any fighting game. You can have a practice mode that really goes in depth and strives to help players improve, but what we've found is that a lot of players never even touch practice modes. It's worth noting that when the fighter genre first hit arcades, there weren't any real tutorials. Instead, the designers tweaked the difficulty level such that after a couple of credits, you had already gone from beginner to intermediate player—something you could then improve upon by learning more moves and practicing. That's the ideal that any fighter should go for, and funnily enough, although a lot of people complain that TEKKEN is too hard to pick up, a lot of other people say that it's too easy for beginners to enter the game and beat people by mashing buttons!

Personally, I don't see that as such a bad thing if it gets more people into the series, gets them curious about it. Another idea is to have an online mode where players can just beat on each other without any life gauges, chatting to each other while learning the moves. There are lots of things like that we'd like to try.

**BS:** *Here's a tough one. TEKKEN has a very complicated story. Can you summarize it within 30 seconds or less?*

**KH:** Ooh, that's hard! Basically, there are these three generations of father and son that don't get along, and two of them have this Devil gene, so their dad wants to know what the Devil gene is, and so they argue with each other about it for a bunch of years. [laughs] All the other characters just sort of get caught up in it. 🎮



### GDC CHINA 2011 FEATURES

- Online Track
- Global Track
- Social Games Track
- Mobile Summit
- Indie Games Summit
- Tutorials
- Sponsored Track



GDC China Expo

### SECURED SPEAKERS INCLUDE

- Mobile Games Summit**  
Keith Lee (Booyah)  
Phil Larsen (Halfbrick Studios)
- Independent Games Summit**  
Nathan Vella (Capybara Games)  
Ami Rao (Supergiant Games)  
Ye Feng (Coconut Island)  
Dongxu He (Gamegou)
- Social Games Track**  
Xiaojuan He (Ubisoft Chengdu)
- Global Games Track**  
Hitoshi Sakimoto (Basiscape International)  
Maxwell Peng (IGS)

### SAMPLE EXHIBITORS AND SPONSORS INCLUDE



# GDC 11 China



## Game Developers Conference™ China

November 12-14, 2011 | Shanghai Exhibition Center | Shanghai, China

Visit [www.gdcchina.com](http://www.gdcchina.com) for more information.





# SOUNDS GOOD!

## PROGRAMMING FMOD FOR ANDROID

**FMOD by Firelight Technologies** is the de facto industry standard middleware system for the production of console and PC game soundtracks. An Android version of FMOD has recently been released, providing powerful interactive audio capabilities for this expanding market. The FMOD Designer tool allows sound designers to import audio files, implement interactivity, and test sounds in context. Exported event and soundbank files are then delivered to the programmer for integration into the game code.

Two sets of APIs control the system: *FMOD Ex*, the low-level audio engine, and *FMOD Event*, the data-driven interactive music and sound effects component. Both APIs can be used as desired, depending on whether you want to directly manipulate audio files or trigger audio events created by the sound designer. Of course, many of the system's preeminent features, such as 3D positioning and reverb/filter processing, are either somewhat pointless or excessively CPU intensive, for use with mobile phones.

To demonstrate what can be done using the FMOD Event system on the Android OS, I added a soundtrack to an open-source (and originally silent) pinball game by Brian Nenner of Dozing Cat Software (see boxout pg. 38). The FMOD API can be downloaded from the web site ([www.fmod.org/index.php/download#FMODExProgrammersAPI](http://www.fmod.org/index.php/download#FMODExProgrammersAPI)), and contains recommendations for getting started. Note for Mac users: The download is a Windows .exe installer, but the necessary files can be extracted using the Unarchiver (or a similar utility), and then placed in your build structure as needed.

When writing and debugging Java code for Android apps, I use the Eclipse IDE with the Android plug-in, on a quad-core i7 iMac (I also have the machine networked to my Pro Tools rig, which facilitates audio production and implementation). The first thing you have to decide is what level of Android SDK you are writing for. This can get somewhat complicated, since there are many Android devices supporting a variety of operating systems and hardware features.

I tested this project on a Nexus phone running Gingerbread (version 2.3.4), so I set

### LISTING 1

```
void Java_com_dozingcatsoftware_bouncy_FMODaudio_cBegin(JNIEnv *env, jobject thiz, jstring
mediaPath)
{
    FMOD_RESULT result = FMOD_OK;
    srand (time(NULL));
    const char *_mediaPath = (*env)->GetStringUTFChars (env, mediaPath, 0);

    __android_log_print(ANDROID_LOG_ERROR, "fmod", "create event system");
    result = FMOD_EventSystem_Create(&gEventSystem);
    CHECK_RESULT(result);

    __android_log_print(ANDROID_LOG_ERROR, "fmod", "init event system");
    result = FMOD_EventSystem_Init(gEventSystem, 32, FMOD_INIT_NORMAL, 0, FMOD_EVENT_INIT_
NORMAL);
    CHECK_RESULT(result);

    __android_log_print(ANDROID_LOG_ERROR, "fmod", "set media path= %s", _mediaPath);
    //result = FMOD_EventSystem_SetMediaPath(gEventSystem, "sdcard/fmod/");
    result = FMOD_EventSystem_SetMediaPath(gEventSystem, _mediaPath);
    CHECK_RESULT(result);
    (*env)->ReleaseStringUTFChars (env, mediaPath, _mediaPath);

    __android_log_print(ANDROID_LOG_ERROR, "fmod", "load eventsystem");
    result = FMOD_EventSystem_Load(gEventSystem, "VPS2.fev", 0, 0);
    CHECK_RESULT(result);

    __android_log_print(ANDROID_LOG_ERROR, "fmod", "get initial events");
    //there is a noticeable pause when these events play for the first time
    //getting them during initialization seems to sidestep the issue
    result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/dings", FMOD_EVENT_DEFAULT,
&gEvent);
    CHECK_RESULT(result);
    result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/message", FMOD_EVENT_
DEFAULT, &gEvent);
    CHECK_RESULT(result);
    result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/rollOver", FMOD_EVENT_
DEFAULT, &gEvent);
    CHECK_RESULT(result);
}
```

up for SDK level 9. This means the app will run on Android devices that support higher level numbers (currently up to 13), like the Motorola Xoom tablet, but not on older phones running Froyo (version 2.2). Of course, you can set up your system for whatever SDK level and hardware you want, but this is one of the reasons

developers complain that the Android market is fragmented (as opposed to the "walled garden" and standard hardware of the iPhone).

There are some other tasks you'll need to do to complete the setup of your development environment, as discussed in the FMOD Getting Started guide. This involves digging around in

the Eclipse Preferences to set classpaths and build variables. You will also need to install the C/C++ Development Tools for Eclipse and, if you've previously only been working in Java, the Android NDK.

## I DREAM OF JNI

» FMOD provides libraries (.so files) for various Android SDK levels and configurations. The libraries are written and compiled in C, but since Android apps are written in Java, FMOD functions must be accessed using the Java Native Interface (JNI). This programming framework allows Android methods to make calls to FMOD C functions to initialize the system, trigger events, and perform other audio tasks.

I recommend using the example programs included in the API download to set up your JNI interface code in Eclipse, because the process is not exactly intuitive. Rather than put your source code in the "src" directory, you place the necessary Android and Application make files (.mk) along with your main.c code into a folder named "jni." You may need to edit the Android.mk file to point at the shared libraries, depending on where you've put them in your build tree. You will also want to add the header (.h) file exported from FMOD Designer, which contains useful names for audio events and music segments.

In main.c, the naming convention for functions takes the following form: "Java" underscore "ClassName" underscore "MethodName," and passes in a JNIEnv pointer, a jobject pointer, plus any additional Java arguments you might need. For example, the initialization function in my pinball game looks like this:

```
void Java_com_dozingcatsoftware_bouncy_FMODaudio_cBegin(JNIEnv *env, jobject thiz, jstring mediaPath)
```

## START ME UP

» Every time you make any call to the FMOD API, you must check to make sure the command completed successfully. This is standard practice for all FMOD implementations but doubly important when using JNI, because memory corruption or thread badness in the low-level C code has the potential to crash the entire system. If the returned result does not equal FMOD\_OK (zero), an error message is printed to the log, and the process is unceremoniously killed with an exit(-1).

Initializing the FMOD engine is performed using a fairly standard set of commands (see Listing 1). First, you create the EventSystem, which allocates memory for the audio engine and sets a pointer for use in subsequent calls. Then you initialize the system, passing in some setup variables. A series of init flags can be bitwise-ORed together to control various options, though most relate to game consoles and 3D

## LISTING 2

```
__android_log_print(ANDROID_LOG_ERROR, "fmod", "get musicsystem");
result = FMOD_EventSystem_GetMusicSystem(gEventSystem, &gMusicSystem);
CHECK_RESULT(result);

__android_log_print(ANDROID_LOG_ERROR, "fmod", "load samples");
result = FMOD_MusicSystem_LoadSoundData(gMusicSystem, FMOD_EVENT_RESOURCE_SAMPLES,
FMOD_EVENT_DEFAULT);
CHECK_RESULT(result);

__android_log_print(ANDROID_LOG_ERROR, "fmod", "prepare cues");
result = FMOD_MusicSystem_PrepareCue(gMusicSystem, MUSICCUE_VPS2_ANDROID, &gAndroid);
CHECK_RESULT(result);
result = FMOD_MusicSystem_PrepareCue(gMusicSystem, MUSICCUE_VPS2_BASS, &gBass);
CHECK_RESULT(result);
result = FMOD_MusicSystem_PrepareCue(gMusicSystem, MUSICCUE_VPS2_DRLOOP1, &gDrLoop1);
CHECK_RESULT(result);
result = FMOD_MusicSystem_PrepareCue(gMusicSystem, MUSICCUE_VPS2_DRLOOP2, &gDrLoop2);
CHECK_RESULT(result);
result = FMOD_MusicSystem_PrepareCue(gMusicSystem, MUSICCUE_VPS2_DRLOOP3, &gDrLoop3);
CHECK_RESULT(result);

result = FMOD_MusicSystem_SetCallback(gMusicSystem, segmentCallback, 0);
CHECK_RESULT(result);
```

## LISTING 3

```
void Java_com_dozingcatsoftware_bouncy_FMODaudio_cUpdate(JNIEnv *env, jobject thiz)
{
    FMOD_RESULT result = FMOD_OK;
    //called every 50ms by FMODaudio.java
    result = FMOD_EventSystem_Update(gEventSystem);
    CHECK_RESULT(result);
}

void Java_com_dozingcatsoftware_bouncy_FMODaudio_cStart(JNIEnv *env, jobject thiz)
{
    FMOD_RESULT result = FMOD_OK;
    result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/startup", FMOD_EVENT_
DEFAULT, &gEvent);
    CHECK_RESULT(result);
    result = FMOD_Event_Start(gEvent);
    CHECK_RESULT(result);
}

void Java_com_dozingcatsoftware_bouncy_FMODaudio_cPlayScore(JNIEnv *env, jobject thiz)
{
    FMOD_RESULT result = FMOD_OK;
    result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/dings", FMOD_EVENT_
DEFAULT, &gEvent);
    CHECK_RESULT(result);
    result = FMOD_Event_Start(gEvent);
    CHECK_RESULT(result);
}
```



## LISTING 4

```
void Java_com_dozingcatsoftware_bouncy_FMODaudio_cDoBassTrack(JNIEnv *env, jobject this)
{
    FMOD_RESULT result = FMOD_OK;
    FMOD_BOOL active;
    if (androidTrackPlayed) {
        FMOD_MusicPrompt_IsActive (gBass, &active);
        CHECK_RESULT(result);
        if (!active){
            result = FMOD_MusicPrompt_Begin(gBass);
            CHECK_RESULT(result);
        }
        result = FMOD_MusicSystem_SetParameterValue(gMusicSystem, MUSICPARAM_VPS2_
BASSEQUENCE, bassSeq);
        CHECK_RESULT(result);
        //cycle through the bass sequences
        bassSeq++;
        if (bassSeq > 2)
            bassSeq = 0;
    }
}
```

positioning. For Android, the default normal settings are probably best.

The number of virtual channels should be set depending on the maximum number of sounds you expect to be played at any one time. Big console games may have hundreds of sounds playing simultaneously, and mixing all of them together at the same time can tax even their beefy CPUs. But my little pinball demo will only play a maximum of twenty sounds at once [up to ten bumpers, five channels of music, plus additional sound effects], so the standard thirty-two channels is plenty.

### SETTING THE MEDIA PATH

» You must then specify the pathway to the folder where the media files are stored; that is, the event [.fev] and soundbank [.fsb] files exported from FMOD Designer. On many systems, these files are copied to a folder on a fast-access hard drive during the installation process. But for Android, the files are placed in the Assets folder of the Eclipse project, then compressed and encrypted as part of the application package [.apk, which is similar to a ZIP archive]. The package does not allow for access to a media directory.

This means you must first copy the files out of the archive into a folder named "fmod" on either internal memory or an external SD card. Then a Java string containing the `mediaPath` name is passed to the initialization routine. For

this game, the space used by the uncompressed audio files is only 828Kb, so using internal memory is acceptable. Larger game audio files might be best stored on the SD card, which works as well. Of course, this can cause problems if the SD card is removed.

Once the system knows where to look, you tell it to load the .fev file into memory. Larger games can support multiple event and soundbank files for various levels, which can be loaded and unloaded at will. However, there is a distinct pause when some of the events are triggered for the first time, which I assume is caused by memory being allocated. This issue is worked around by doing a `GetEvent` call on the problematic sounds during initialization. Better for the pause to happen in the start-up procedure than during gameplay.

### MUSIC TO MY EARS

» Another pause is produced when initializing the `MusicSystem` (see Listing 2), because the music segment samples must first be loaded into memory to prevent hiccups and loss of sync during playback. Again, this may not be a problem for fast console CPUs, but seems to be required in the mobile environment.

For this game, the combined `GetEvent` and `LoadSoundData` calls can produce as much as a 10-second delay when starting up the game, during which time the user is presented with a blank screen. This is okay for a demo project

like this one, but explains why many mobile games display a pretty splash screen or simple animation on start-up—to distract the user while the audio loads.

Calls to `PrepareCue` allocate memory and set the value of pointers needed to start and stop the music cues, which are comprised of one or more music *segments*. `SetCallback` designates the function to be called when a segment finishes playing, which in this case is used to start the "bass" track after the "android" segment completes its first playback.

Actually, there are a variety of callback types that can be tested for, including `Segment_Start`, `Sample_Release`, `Reset`, et al. In other projects, I've used the `Beat` callback to toggle button colors, so that the interface flashes in time with the music. Callbacks can also be used to time sprite animation to the music's tempo, change sound effect pitches to match the changing keys of a music cue, manage sophisticated transitions and hero moment fanfares, and more.

### KEEP ME UPDATED

» Most of the other functions in `main.c` are quite simple (see Listing 3), consisting of one or two calls to the FMOD API and checking the result each time. The most important of these functions is `EventSystem_Update`, which must be called in a continuous loop, every 50 milliseconds, the entire time the app is running. It is used to trigger callbacks, and to ensure that audio events execute correctly.

To play an event, you simply do a `GetEvent` call, specifying the event by name. This sets the pointer to the desired `FMOD_EVENT` using a generic `gEvent` variable. The subsequent `Start` call fires off the event, which could be a single sound (like "startup"), one of a number of sounds in a container (like "dings"), or a complex series of sounds in multiple layers controlled by parameter.

As the programmer, you don't really care too much about how the sound is produced—that's the beauty of data-driven sound design. The sound guy, who cares obsessively about the audio, has already done the interactive implementation work in FMOD Designer. So instead of telling you, "When the ball hits the bumper, you need to play one of these six sounds, at random, no repeats, at half volume, up to ten times simultaneously, dropping the oldest sound if you go over the limit," he only has to say, "When the ball hits the bumper, play the `dings` event." Easy!

A similarly simple command is used to play the bass music cue (see Listing 4), which consists of five different segments (three loops in two keys, with two transitions) controlled by

////////// As the programmer, you don't really care too much about how the sound is produced—that's the beauty of data-driven sound design. The sound guy, who cares obsessively about the audio, has already done the interactive implementation work in FMOD Designer.

parameter. Making sure all the segments stream, loop, and segue properly, at the correct times, while maintaining sample accurate sync, is not a trivial operation, but fortunately, you don't need to do any of that. All you have to do is `Begin` the bass line, and set the `bassSeq` parameter when appropriate. Easy!

### ALGORITHMIC VERSUS DATA DRIVEN

» Of course, in the resource-constrained environment of mobile phone applications, there are always special considerations. One is file size, and smaller is always better. So as an audio programmer, you can sometimes generate sounds algorithmically to save space.

The `cPlayRollover` function does exactly that (see Listing 5). The `rollover` event is just a single "plink" sound, but each time the function is triggered, it plays the event up to three times, repitching the sounds to a pentatonic scale. This creates a series of 41 possible notes and chords, using only a single sample.

A strictly data-driven way of doing the same thing would be to record all the combinations, put them in a container event, and trigger them to play randomly. The sounds produced during gameplay would be exactly the same as the algorithmic method, but they would require 40 times the amount of audio memory. Algorithmic techniques can significantly reduce the memory footprint required for certain kinds of sound effects by slightly increasing the CPU (and engineering resources) required.

The `DoDrumTrack` function performs a similar operation for the interactive music (see Listing 6). The three drum loops are designed to be played individually or in combinations. Rather than sample each combination or create intricate (and potentially confusing) music cue playback parameters, I simply start and stop the various loops programmatically, letting FMOD keep the tracks in tempo sync with the bass line.

The disadvantage of the algorithmic technique, of course, is that it breaks the data-driven model, taking the audio implementation task out of the hands of the sound designer, and making the programmer do the work. This may be considered "old school" (in the early days of game development, this was the *only* way to produce interactive audio), but it's sometimes worth the tradeoff for mobile audio. Besides, in this case, I'm the sound designer and the programmer, so it really becomes a "six of one, half a dozen of the other" situation.

### GIMME A CUP OF JAVA

» All functions in `main.c` are called from the `FMODaudio.java` class. While it's possible to use the native prototypes to call the C functions from any Java class directly, I prefer to keep all access to FMOD processing in one place in order to make modification, maintenance, and debugging easier to manage. The game code calls the `FMODaudio` methods whenever an audio task needs to be done.

#### LISTING 5

```
void Java_com_dozingcatsoftware_bouncy_FMODaudio_cPlayRollover(JNIEnv *env, jobject thiz)
{
    FMOD_RESULT result = FMOD_OK;
    FMOD_EVENT_PITCHUNITS units = FMOD_EVENT_PITCHUNITS_SEMITONES;

    //play up to three events, each randomly pitched to a different note in the
    pentatonic scale
    //the rollover ding is E, so in semitones, the other pitches are -4 (C), -2 (D), +3
    (G), +5 (A), +8 (C)
    float pitch[] = {-4, -2, 0, 3, 5, 8};
    int pitchDx[] = {0, 0, 0};
    int i;
    for (i = 0; i < 3; i++) {
        switch (i){
            case 0:
                result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/
                rollover", FMOD_EVENT_DEFAULT, &gEvent);
                CHECK_RESULT(result);
                pitchDx[i] = (rand() % 6);
                result = FMOD_Event_SetPitch(gEvent, pitch[pitchDx[i]], units);
                CHECK_RESULT(result);
                result = FMOD_Event_Start(gEvent);
                CHECK_RESULT(result);
                break;
            case 1:
                pitchDx[i] = (rand() % 6);
                if (pitchDx[i] != pitchDx[i-1]) {
                    result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/
                    rollover", FMOD_EVENT_DEFAULT, &gEvent);
                    CHECK_RESULT(result);
                    result = FMOD_Event_SetPitch(gEvent, pitch[pitchDx[i]], units);
                    CHECK_RESULT(result);
                    result = FMOD_Event_Start(gEvent);
                    CHECK_RESULT(result);
                }
                break;
            case 2:
                pitchDx[i] = (rand() % 6);
                if (pitchDx[i] != pitchDx[i-1] &&
                    pitchDx[i] != pitchDx[i-2]) {
                    result = FMOD_EventSystem_GetEvent(gEventSystem, "VPS2/VPS2/
                    rollover", FMOD_EVENT_DEFAULT, &gEvent);
                    CHECK_RESULT(result);
                    result = FMOD_Event_SetPitch(gEvent, pitch[pitchDx[i]], units);
                    CHECK_RESULT(result);
                    result = FMOD_Event_Start(gEvent);
                    CHECK_RESULT(result);
                }
                break;
        }
    }
}
```



For example, `FMODaudio.playScore` is called from `Field.java` whenever the ball hits a bumper, which calls `cPlayScore` via JNI to play the "dings" event. The music tracks are controlled based on a running total of bumpers hit. This way, the soundtrack will never be played exactly the same way twice, because the rate of change will depend on how quickly you score. It's a simple but effective means for controlling the variations of the interactive music (see Listing 7 on [www.gdmag.com/resources](http://www.gdmag.com/resources)).

`FMODaudio.java` also performs a few other essential tasks. It starts and calls the `cUpdate` loop every 50 milliseconds, using the Android `Handler` mechanism (as recommended by the FMOD example code). The main and FMOD libraries are loaded, and the native prototypes used by JNI are defined here as well.

The first time the game is played, the FMOD event and soundbank files are copied out of the archive into internal memory in `BouncyActivity.java`, the primary activity class that contains

The Android app and Eclipse project described in this article can be downloaded from [www.gdmag.com/resources](http://www.gdmag.com/resources) or on the author's Twittering Machine website: [www.twittering.com/FMODforAndroid](http://www.twittering.com/FMODforAndroid).

the application control overrides. Calls to `FMODaudio` methods to start, stop, and pause the soundtrack are also done here. The sound effects events are all triggered from `Field.java`, the main gameplay class.

### FMOD FOR ANDROID, AND BEYOND!

» Remember, this pinball app is intended to be a simple demo of what can be done using FMOD for Android. Much more complex, cross-platform implementations are possible for many handheld devices. The same FMOD Designer projects can export event and soundbank files for an Android phone, an iPad, a PSP, and a Nintendo 3DS. Most of the same API calls can be reused as well.

In fact, subsets of huge Xbox 360 and PS3 soundtracks can easily be ported to mobile devices, using the same audio assets as the big boys and much of the same code. As games become more social, more networked, more cloud based, more ubiquitous, and more platform agnostic, FMOD can provide sophisticated interactive audio soundtracks for mobile games everywhere. 🎮

**PETER "PDX" DRESCHER** is an interactive audio veteran, and published authority on game audio technology. He runs *Twittering Machine*, a Pro Tools project studio in Redmond, WA, providing music, sound effects, and programming services for multimedia software and mobile devices. Previously, he was an audio director at Microsoft, principal sound designer at Danger (maker of the *T-Mobile Sidekick*), and a road dog bluesman piano player.

### LISTING 6

```
//play the first three tracks in sequence, then in random combinations

ctr++;
if (ctr > 3)
    drumTrax = (rand() % 6)+1;
else
    drumTrax = ctr;

switch (drumTrax){
    case 1:
        result = FMOD_MusicPrompt_Begin(gDrLoop1);
        CHECK_RESULT(result);
        break;
    case 2:
        result = FMOD_MusicPrompt_Begin(gDrLoop2);
        CHECK_RESULT(result);
        break;
    case 3:
        result = FMOD_MusicPrompt_Begin(gDrLoop3);
        CHECK_RESULT(result);
        break;
    case 4:
        result = FMOD_MusicPrompt_Begin(gDrLoop1);
        CHECK_RESULT(result);
        result = FMOD_MusicPrompt_Begin(gDrLoop2);
        CHECK_RESULT(result);
        break;
    case 5:
        result = FMOD_MusicPrompt_Begin(gDrLoop2);
        CHECK_RESULT(result);
        result = FMOD_MusicPrompt_Begin(gDrLoop3);
        CHECK_RESULT(result);
        break;
    case 6:
        result = FMOD_MusicPrompt_Begin(gDrLoop1);
        CHECK_RESULT(result);
        result = FMOD_MusicPrompt_Begin(gDrLoop3);
        CHECK_RESULT(result);
        break;
}
```





take  
control  
of your  
future

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

the art and business of making games





# TAKING FEEDBACK

## WHY DESIGNERS MUST LEARN HUMILITY

**“** You have to design for success, plan for failure, and then know how to rebound from that failure. Our 1.0 version of *AI WAR* was successful but not wildly so. Little irritations added up and annoyed people enough that they stopped playing. When the public gets the game, they find problems that you never did, and you must devote time to fixing them. Once you do, the fans are happy, and the game becomes more successful than it would have been otherwise. You have to eat a lot of humble pie, but after a while you get really used to that, and the stuff that used to give me a hard time emotionally when I was first starting out is just par for the course now. I don’t even notice. It’s just feedback, and whether it’s viable determines if it goes in the ‘yes’ bucket or the ‘no’ bucket or the ‘maybe’ bucket.”

—Chris Park, designer of *AI WAR: FLEET COMMAND*, from the *Three Moves Ahead* podcast, Episode 37

/// To be a game designer is to be wrong. Ideas do not work out as planned. Certain mechanics prove tedious instead of fun. Players spend their time focusing on the “wrong” parts of the game. The original vision starts to slowly slip away as it comes into contact with the gamer.

Furthermore, designers are often bombarded with suggestions—from the rest of the team, vocal fans, well-meaning friends, and the dreaded executive flybys. Faced with such a deluge, a developer’s natural instinct is often to defend the design. These suggestions are simply trials and tribulations to be overcome so that you, the designer, can continue making the game “your” way.

The problem with that ideas is that processing feedback is a fundamental part of the game design process, as important as the original vision itself. Games are not static objects that can be observed or judged in a vacuum. Instead, they live in the minds of our players, and each of them might experience the game in a different way.

Thus, designers need to be great listeners more than great persuaders. If a designer ever finds herself needing to explain why a player should be having

fun when he is not, something has gone seriously wrong. Instead, designers must listen to players with a great sense of humility, with an understanding that this feedback is the only way to remove the fog that separates the game in the designer’s head from the one that is actually playable.

Ultimately, games must speak for themselves, so designers need to learn not to rely on the crutch of their own enthusiasm and communication skills to sell their ideas. Dropping one’s ego to learn from the criticism can be an emotional challenge for designers who identify too closely with their original vision. Instead, we need to place faith in the design process itself, not in that inevitably doomed first draft.

### LISTEN TO THE TEAM?

» Thus, gathering and assessing feedback is a crucial skill for any modern game designer. The first, and sometimes only source of feedback is the team itself. Any committed development team should be ready and eager to play its own game and provide the designer with crucial early feedback on what works and what doesn’t.

However, feedback from the team carries significant limitations.

To begin, the team lives with the game for months, probably even years—far beyond an average player’s time with the game. As veterans of development process, they can play on auto-pilot, blinding themselves to unintuitive mechanics or confusing UI. Developers quickly lose the ability to see the game objectively, often believing that the game is either in better or in worse shape than it really is.

Further, team members are often hired for their special skills—3D animation, sound design, network optimization—and not because of their passion for the product itself. They might forget some of the simple joys that new players will experience when starting the game. More dangerously, they might burn out on the game altogether and begin to resent the intense demands of their most vocal community members.



## TRUST THE COMMUNITY?

» The community, of course, can be an overflowing cauldron of ideas and suggestions. When developing *CIVILIZATION III*, our most active fansite presented us with “The List”—an exhaustive, 20,000-word tome detailing their expectations for the upcoming game. Wading into the forums can be an overwhelming experience for most designers, requiring a thick skin as posters rip apart their development choices.

However, no one understands a game better than players who are dedicated enough to join a community and make the game a part of their social life. These gamers might play for hundreds of hours, gaining knowledge of mechanics and systems that elude even the designers themselves.

The challenge with forums is that what players say and what they actually do are often two different things. In a talk at GDC 2011, Ben Cousins described just such a situation with the free-to-play online shooter *BATTLEFIELD HEROES*. The game had not been generating enough revenue, so the team reworked the monetization system to make it harder for nonpaying players to “rent” premium items and also to sell the items directly for cash.

The change caused an uproar in the forums; within a week, a 4,000-post thread developed decrying the changes, with many veterans pledging to quit the game. Exacerbating the debate was the fact that Cousins had publicly declared years earlier that “we have no plans to sell weapons.” The press picked up the controversy, leading to articles such as “*BATTLEFIELD HEROES* Is Practically Ruined” on Kotaku.

The metrics, however, told a very different story: revenue tripled with no discernible decrease in active users. It is hard to tell if the posters who pledged to quit were actually lying, but they were clearly not representative of the average player. Cousins dug deeper and found that only 20% of all players had ever visited the forums and that only 2% had ever actually posted a message.

Furthermore, compared with the silent majority, community members had a much higher conversion rate (27% vs. 2%) and ARPPU (Average Revenue Per Paying User) [\$110 vs. \$32]. Thus, the thoughts expressed on the forums were an inaccurate and misleading representation of the player base’s actual feelings. The posters were perhaps making threats in the hopes of changing the game in their favor (to save themselves money) instead of revealing their actual beliefs.

The *HEROES* experience highlights the importance of metrics as a secondary source of feedback. Watching what players actually do can be as important as listening to what they have to say. Still, metrics have their limitations, as no set of numbers is going to help the designer understand why people have stopped playing the game (though metrics can show exactly when players quit and don’t come back, which can be useful).

While measuring how often unit X is built instead of unit Y provides a valuable tool for balancing an RTS, it’s not necessarily clear that making the two units equally viable will actually make the game more fun. Metrics are great at answering specific objective questions that require real data—what difficulty level is most commonly picked first?—but to learn whether a game is actually fun, the designer’s only option is to find out what players are feeling by listening closely to what they are saying.

## FIND YOUR VOICES

» And so it is that designers are left with the conundrum that their best source of feedback—the vocal fan community—is not only an unreliable source of information, but one that might be actively trying to mislead the developers. Perhaps most frustrating is the possibility that the more forum posters are aware that the team is listening, the more likely they are to lie to the designers to get what they want. MMO developers are familiar with the player type who will always argue that his or her character’s class is woefully underpowered, against all objective evidence to the contrary.

This problem can be handled with a more proactive approach to gathering fan feedback. Not all fans are the same, as only a precious few are able to see the forest for the trees and provide accurate feedback that speaks to the

health of the game’s overall experience. While developing *CIVILIZATION IV*, we cultivated just such a group of enlightened fans to provide feedback we could trust.

These players had a history of being reliable sources of information during the post-release development of *CIV III*, and we provided them with a special private forum for direct communication with the team. This group became our primary source of feedback both before and after release, providing us with much greater certainty about which ideas were working and which ones were not. *CIV IV* would have been significantly different—and certainly worse—without their input.



These groups must be managed carefully, however, to prevent the members from developing a sense of entitlement or superiority over other players. For this reason, the group’s existence should be, if possible, a closely guarded secret. Further, the developers must try their best to find a representative group of players, perhaps looking outside the forums for new members.

## LISTEN EARLY, LISTEN OFTEN

» Another accurate way to gather feedback is with “Kleenex” testing, so named because players get access to the pre-release game once and are then thrown away. The valuable lessons here come from players’ initial reactions to the game, before they become accustomed to UI holes or gameplay quirks. Valve famously runs these tests regularly by gathering up random players from local game stores.

However, depth testing is also important, which can only be achieved by giving players continual access to the game before release, to explore and experiment with the game’s systems and mechanics. Big publishers often have trouble giving fans early access to their games, for fear of them leaking cracked versions to pirate sites or spreading confidential information to rival publishers.

Indie developers actually have a big advantage here because their greatest danger is not security, but obscurity. Thus, many recent indies (*SPELUNKY*, *DESKTOP DUNGEONS*, *THE WAGER*) have released early versions of their games, generating both marketing buzz and valuable feedback.

Some indie games, such as *FROZEN SYNAPSE*, *MINECRAFT*, and *SPYPARTY*, have even generated revenue by selling access to these alphas. This option gives teams the chance to bootstrap their way along while also learning how the game performs in the wild, a great option to help fight the long odds that most indies face.

No matter what combination of methods you choose, the most important thing is that designers be humble and realize when they’re wrong as well as when they’re right. To err is human, but to be able to learn from, accept, and fix those errors can only make us better designers. ☺

---

**SOREN JOHNSON** is a designer/programmer at EA2D, working on web-based gaming with *strategystation.com* and *DRAGON AGE LEGENDS*. He was the lead designer of *CIVILIZATION IV* and the co-designer of *CIVILIZATION III*. Read more of his thoughts on game design at [www.designer-notes.com](http://www.designer-notes.com)



# GET THE MEMO

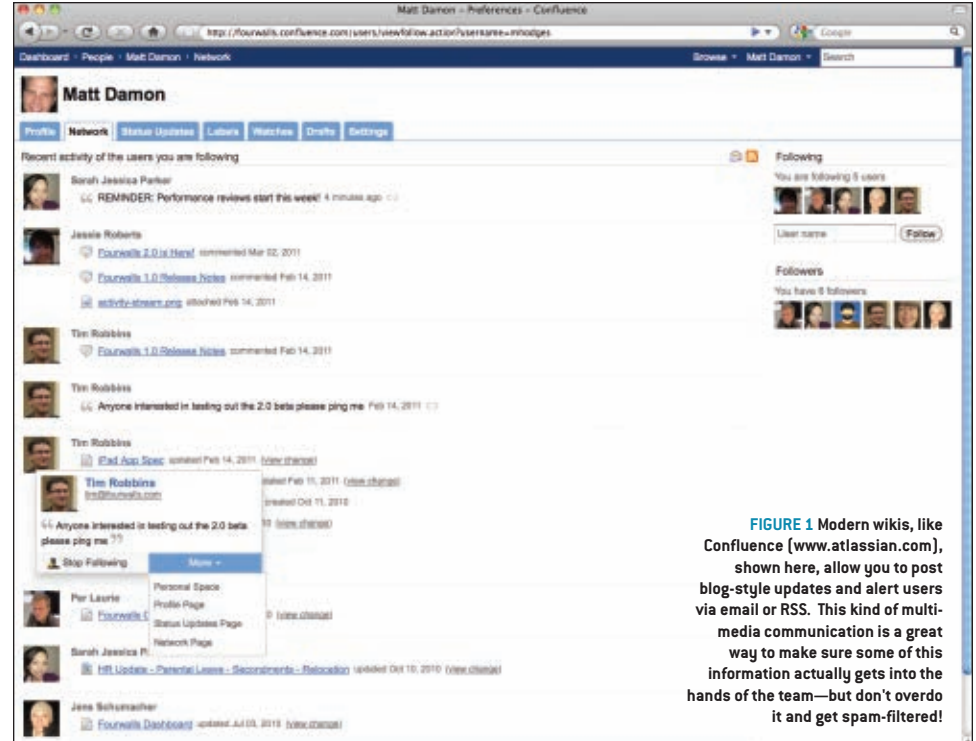
## COMMUNICATING WITH ARTISTS

**Art springs** from the eternal human impulse to describe, explain, and tell stories about the world. When we describe a great work of art, we say it speaks to us. So it's a bit peculiar, to say the least, that so many artists treat written communication the way most people treat dentistry: We put it off as long as possible, and when it becomes unavoidable, we try to float free of our bodies and let our minds go to their Happy Places until the ordeal is over.

You might think that's an unfair stereotype. Every art department sports a couple people who don't mind over-sharing (everything I need to know about EVERQUEST, I learned through the four inches of wood and drywall that separated me from a pair chatty role-players who knew that no epic quest is truly over until it's been sung—endlessly—by the bards). But it's hard to find an art director, a tool developer, or a producer who thinks that artists in general can be relied upon to read memos, keep up with news feeds, or, as the unprintable saying goes, "RTFM."

Twitter, RSS, Web pages and email are marvelous tools of modern communications when it's time to let people organize a pub crawl, zombie walk, or looting spree. These remarkable technologies put fear in the hearts of authoritarians everywhere—but the world's dictators can at least rest easy in the knowledge that the masses will apparently never use all those newfangled modern tools to correctly export levels, check in properly compressed DDS files, or change the length of a walk cycle.

If you are one of those unlucky souls tasked with getting an art team to follow directions, you've



**FIGURE 1** Modern wikis, like Confluence ([www.atlassian.com](http://www.atlassian.com)), shown here, allow you to post blog-style updates and alert users via email or RSS. This kind of multimedia communication is a great way to make sure some of this information actually gets into the hands of the team—but don't overdo it and get spam-filtered!

probably tried it all: email notices, wiki pages, video tutorials, cheat sheets, snarky Post-it notes, and humorously veiled threats delivered at team meetings. Most likely, you've also despaired. Getting the word out about anything is a hard, thankless slog. It's an unending task. Art direction, tools, and game features change all the time, leaving many artists uncertain—or worse, completely certain but just plain wrong—about how to do their jobs.

### RADIO SILENCE

» The most common antidote is simple: Leads and managers wait for things to go wrong, and then hover over somebody's shoulder until they fix the problem. Unfortunately, in the modern game business it's hard to survive entirely on word of mouth. Big modern

teams are too big for one person to effectively circulate knowledge. To make it worse, teams and outsourcers are scattered all over the globe, cut off from whatever local grapevine might exist in a particular studio to keep people at least marginally informed. And, to ice this dysfunctional cake, there's the constant churn of game technology. The only thing worse than having no information about how to get something into the game is documentation that's wrong or out-of-date. It's a perfect recipe for dysfunction and inefficiency.

You might just say, so what? If people don't know what to do, they'll ask or muddle along. People will figure it all out eventually—or get fired and become somebody else's problem. But it's often not the individuals who are the problem.

Individually, most artists do want to get their jobs done, and they'll take pains to make sure they know how things work. The real source of the trouble is the combination of constant change and poor or inadequate communications. Lack of reliable, accessible information leaves us in the position of isolated medieval villagers—we have to get by on a combination of rumors, superstitions, and tradition. Over time, the rumors and misinformation become almost impossible to root out. Every team acquires a little body of magic rituals that people believe keep things running ("Delete the history! Always triangulate transparent objects! Make sure the texture name is all uppercase!") even though these do nothing but waste time. These taboos can persist for

years as line artists pass them along by word of mouth while managers and tech are blithely confident that everyone has read the memos.

Good communication doesn't come easily. The most important tactic is, sadly, the most time-consuming and expensive: Do everything. You can't rely on emails, wiki pages, help files, video tutorials, or even brown-bag lunch demos—or, to be more precise, you can't rely on any of them alone. Each tactic will reach some fraction of the team—the only way to reach everyone is to try lots of different avenues at once. Of course, it's maddening to have to repeat the same information in so many different ways. To make it more irritating, keeping all these different versions of the same basic facts in sync is tricky and time-consuming. Despite these drawbacks, though, coordinating all these different kinds of documentation is the most important thing you can do to keep the team informed. The only offense that works is flooding the zone.

#### SPAMBOTS TO THE RESCUE!

» Fortunately, modern tools make this much less of a chore than it used to be, and they don't demand wizardly IT skills either. Most modern wikis or intranets include RSS, email, or even Twitter feeds, all of which can broaden the reach of your updates without too much extra work on your part. (See Figure 1.) The updates are an invaluable way to put critical info in front of your users. Unfortunately, (as anybody with an email account knows) too much information is the same thing as too little. If every artist's inbox has a dozen wiki update notices every day, you can bet that most of them are going to be unread. And not many readers will make it all the way through that 20-page monthly art department newsletter, either.

The best strategy is to use real-time communications to politely put useful information out there for the team, not to snow them under. Email and RSS are terrible for actually distributing detailed information, since they tend to scroll by at the edge of a user's attention.

However they are fabulous as teasers or reminders. Just think about how you use your own email inbox as a substitute for long-term memory: You skim the updates as they scroll by and mentally file them away for the future. That's how most artists treat the company mail: They'll only pay close attention when something has gone wrong and they need help—"Oh wait, wasn't there an email about that last week?" And then it's off to the search box.

The search box is your best friend—but only if you write docs that are search friendly. First and foremost, use consistent names for tools, menus, and buttons so that finding the relevant mails is easy. Second, remember that your bulletins will probably be seen weeks or months in the future, so leave the details out. If the art team finds detailed information moldering in their inboxes, they'll assume it's good—even though it's really as old and busted as a 2002 MySpace page. Plan for the day when they vaguely recall the email and dig it up out of their inboxes: Have good, search-friendly headlines, crisp summaries, and (most importantly) links to the wiki or document where the up-to-date mojo really resides.

#### LIVE FROM THE ART BIBLE

» The inbox time warp underlines another key point. Good documents are alive: If you have multiple copies of the same information floating around, you're inviting confusion. Don't expect a busy artist who's got looming deadlines to compare version numbers or file dates to find the right version of a doc. You can't even rely on source control unless you have a very rigid brand of sync discipline. Online documentation is the best way to make sure everybody is, literally, on the same page. Especially nowadays, when wikis and other web-based tools are so easy to find and use, there's no excuse for hiding important stuff in Word docs, readmes, or PDFs.

Interactive media like wikis are also superior to Word docs because they can be edited, commented on, and updated by lots of people. You do need to be realistic about what

you expect from your wiki, however. The early buzz on wikis was full of overblown, internet-boom-era optimism. When it turned out that very few users actively update or edit wikis, a lot of teams concluded that they were just another overhyped tech fad and lapsed back into their old email-and-stored-doc habits. It's true that only a small minority of users will post or edit wiki docs—but that's not a drawback. You don't need your docs to turn into Wikipedia. Good docs are not about democracy, they're about getting stuff done.

The genius of wikis isn't that they turn everybody into an editor, it's that anybody can post a quick comment that says "This button doesn't work like the docs say" or "It works a lot faster if you close your UV Edit window before you

 Nobody starts off a career as an artist because they want to make excellent wiki pages.

kick it off." Tips, tricks, bugs, and questions don't replace formal editing decisions, but they make the docs more accurate and keep them fresh. Just don't expect a wiki to magically free the people who really own the docs from having to maintain them.

The last great thing about online docs is that they are easy to integrate directly into your tools. Until pretty recently, adding decent help to in-house tools was a serious chore, requiring specialized programs to compile CHM files or other esoterica. Those tools have gotten better in the last few years. You can now use tools like Helpinator ([www.helpinator.com](http://www.helpinator.com)) rather than having to go through a complex compilation process, but the added time and trouble of having to run a program just to fix

a typo in your docs is a slow drain on a task which, let's be honest, is already one we tend to avoid. Oldschool CHM files do have a nice professional feel, but the slickness isn't worth the hassle. Updating a wiki, on the other hand, is almost frictionless—fixing a typo or updating a description when one of your users points out a problem doesn't demand much more than typing. Online help is also easy to integrate directly into your tools: A line or two of maxScript, MEL, or Python can open a Web browser to an appropriate help page that's always up-to-date. It's hard to have too many help links—every dialog box or tool window should have at least one; it's a key way for all those folks who never read their emails to learn about the studio toolset.

#### THE DOC IS IN!

» If you got into this business because you have a burning desire to create informative technical documentation, well... Actually, there's no point finishing that thought. Nobody starts off a career as an artist because they want to make excellent wiki pages. Unfortunately, the work we do is too complex and too fluid to get by without some attention to boring stuff like documentation. Many of us will have to deal with email bulletins, RSS feeds, obscure wiki formatting codes and all sorts of other thoroughly unartistic dross in order to focus on our real jobs. It's hardly the only aspect of the game art life that involves a bit of eat-your-vegetables stoicism. On the plus side, doing a good job may not be glamorous, but it will pay off in the long run—in the form of fewer bugs, less time spent handholding the noobs, and hopefully more time to do the job you signed up for in the first place. 🙏

---

**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*.



# EFFICIENCY...FOR WHOM?

## DISSECTING THE POPULAR FREE-TO-PLAY SALES CURVE

The game industry is currently head over heels in love with the free-to-play, microtransaction-supported business model. Time will tell whether this will turn out to be love everlasting or a passing fling. Personally, I'm always in favor of more choice for gamers, and that includes greater choice in terms of payment methods [aka "more paths to the cash register"].

While there's no doubt that free-to-play has a place (the list of successes is long and growing), I want to discuss two ideas common to the business model debate that seem to get misrepresented in many discussions: price elasticity (more specifically, price elasticity of demand) and market efficiency.

### PRICE ELASTICITY OF DEMAND

» I've seen the graph in Figure 1 referred to in a number of discussions. It's sometimes mislabeled as the supply/demand curve, but since in most cases supply is for our purposes unlimited (digital distribution) and/or price fixed at point of sale (as in console retail), it's really a "price elasticity of demand" curve. This curve states that lowering a price (say from \$60 to \$30 in the case of a big-budget retail game, or from \$1.99 to \$0.99 in a smartphone app), will cause demand to increase.

When offering a game at a single price, one attempts to hit the optimal price to maximize the area of the rectangle under any point on that curve. What you're doing is aiming for the apex of the curve seen in Figure 2. In this example, the high price at \$70 scared away too many customers, while the low price at \$30 garnered more customers, but not enough to make up for the lower revenue per customer.

While optimizing the area covered by the rectangle is good, it's easy to see that money is left on the table. There are customers paying \$60 that would be willing to pay \$70 for the game, and would-be customers willing to engage with the game at \$30, but without that option are not going to be customers at all. The industry has addressed this in the past through things like premium collector-edition releases (which have a higher-price option) downloadable content, and things like basic versions or time-delayed price waterfalls.



What free-to-play models try to do is to let customers engage at whatever level they'd like to, allowing an infinite number of points along this curve, thus maximizing the area under the curve that represents revenue.

This is a simple enough concept, but it becomes more complicated when one starts talking about having multiple price points for a product.

The way I've seen this discussed in several recent conferences is that companies making the leap should look at the graph in Figure 1 and say, "I'd like to garner all that revenue at P2, but also charge my most enthusiastic customers the price at P1, and offer the price at P3 for people

that might otherwise not be customers at all." Such an approach is represented in Figure 3a.

It looks like a great approach, but it makes three errors.

The first is relatively simple: the curve shouldn't extend infinitely. Raising the price of the product will eventually make the game hit a point where the number of customers willing to pay that price is zero. Similarly, dropping the price, even all the way to zero, does not guarantee volume will go up to, say, a number representing the Earth's population. Developers need to consider how the customers are going to perceive the game's potential value.

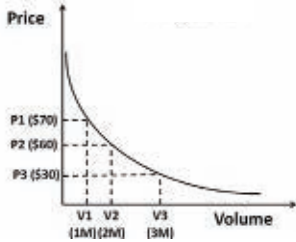


Figure 1 Price elasticity of demand.

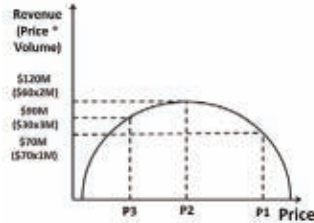


Figure 2 Price versus revenue.

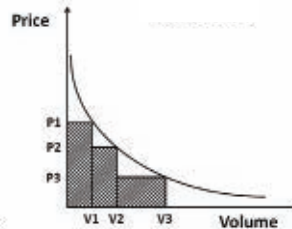


Figure 3a Price versus volume at a single price point.

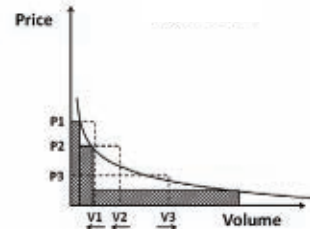


Figure 3b Price versus volume at multiple price points.

The second problem is that there's a difference between what volume would be achieved by offering the product at any one price, and the volume achieved at each price point when offering the product at multiple price points.

In the latter case, the curve is going to look more like Figure 3b.

### MARKET EFFICIENCY

» In Figure 3b, some number of customers are willing to pay price P1, but it's a lower number than it would be if the product were *only* offered at that price (represented by the dotted line). The same goes for price P2. In exchange, there are more customers at price P3. Some of these are previously would-be P1 or P2 customers, but hopefully many more are people that wouldn't have been customers at all.

The area under the curve represents potential revenue, and the shaded areas represent revenue earned versus any white space, which is potential revenue left on the table. This is a form of market efficiency, and is also where the third erroneous interpretation comes in.

One can extrapolate from Figure 3b that adding ever-more price points lets a developer better maximize the area under the curve from which they are extracting revenue—they would be more efficiently monetizing their audience.

However, another way to view it is that it lets the consumer better minimize the overall area under the curve. In this respect, the market efficiency is improving to let customers pay for only what they want to, and not a penny more.

Either way it is a discussion of efficiency, but it cuts both ways. There's the efficiency of the developer to monetize the customer, but also the efficiency of the customer base to extract value out of the game. I should be clear: Both of these are a good thing. Maximizing revenue is good for the industry. Maximizing value for dollar is good for the consumer.

This has played out in recent years in the music business. The default price point in years past was an album of 12 to 15 songs for around \$15. Now music fans can engage at any increment of \$0.99. Makers of many-hit albums may still sell entire albums, though fewer than before, but they will also reach many more customers at \$0.99. However, mediocre albums that

“Maximizing revenue is good for the industry. Maximizing value-for-dollar is good for the consumer.”

used to sell at \$12 as the only way to get that single hit song, have sales now supplanted by a single \$0.99 sale of that track. Musicians can reach more people than ever before, and consumers can buy the songs they like without having to buy a bunch of songs that they don't. I think this gives us an indication of the implications for games.

Free-to-play gives developers another model to target, one with many benefits, including a less daunting on-board ramp (a huge plus for free-to-play

that we didn't even touch on here), and more efficient monetization of the market. However, those developers coming from other parts of the market and from other models, need to ask themselves which form the curve will take, and at what level they believe customers will want to engage with their game. Free-to-play is already working for low-cost online games, and I'm certain it will work for the best big-budget franchises. But those developers transitioning from selling a limited or lackluster experience at retail for \$60 may find the free-to-play world even crueler to them than the old model. Such cases will find that the free-to-play romance ends in tears. ☹

**KIM PALLISTER** works at Intel doing game industry forecasting and requirements planning. When not prepping the world for super-cool hardware, he blogs at [www.kimpallister.com](http://www.kimpallister.com). His views in this column are his and do not reflect those of his employer.

# NORCO COLLEGE

Norco, CA

## We're hiring.

### Multimedia Instructor

a full-time, 10-month, tenure track position

**You will be responsible for**  
 lecture and/or laboratory instruction in Digital Game Art, Game Design, Motion Graphics/Interactive Media, Mobile Applications, and/or Digital Animation in support of two federal Gaming & Multimedia grants totaling \$6.6 Million.

**Education:** Bachelor's degree in Media Arts, Digital Animation, Simulation & Game Development, Interactive Media, or related area is required. Master's degree in a Digital Media or a Multimedia field related to Game Development, Motion Graphics, or Digital Animation is preferred.

for more details on the position and the application process, visit:  
<https://jobs.rcc.edu/applicants/Central?quickFind=53482>

an equal opportunity employer

# Recruitment at GDC Online

## A PRIMER FOR THE JOBS AREA OF THE EXPO FLOOR

∞ The Game Developer's Conference held each year in San Francisco has become legendary for its recruitment opportunities, with many game companies from across the world recruiting heavily. But it would be foolish for job-seekers to wait until March when opportunities like GDC Online, held between October 10–13 in Austin, Texas are on the horizon. GDC Online is intended for professionals developing online and connected games (such as cloud-based games, MMOs, virtual worlds, and games on social networks). For developers looking to move into—or change jobs within—this space, the show offers an ideal opportunity to make a targeted connection with recruiters in a more intimate setting. We talked to a few recruiters—from CCP Games, Kabam, and Nexon—to hear their opinions of recruiting at the more focused GDC Online.

"Austin has a large, deep pool of gaming and technology talent, and GDC Online, with its focus, is an obvious place to plant a flag for recruiting," said Scott Thomas, SVP of global talent at Kabam (KINGDOMS OF CAMELOT, DRAGONS OF ATLANTIS). Paula Fellbaum, executive director, HR and admin operations at Nexon (MAPLESTORY, COMBAT ARMS) said that GDC Online's location and focus weren't the only attractions.

"[GDC Online] offers an opportunity to be more one-on-one with everyone, and isn't as crazy as the huge shows," she said. "It feels like a more intimate environment where you have the opportunity to actually get to know a little bit about the people you are meeting and not just qualify and move on to the next person."

It's clear that this chance to be known as more than just your resume is important: With the continual evolution of online games, the recruiters surveyed all expressed interest in finding candidates with subtleties.

Kelley Barnes, senior recruiter at CCP Games (EVE ONLINE, DUST 514) explained, "Transmedia is a relatively new buzzword in employment circles, but people who can imagine the game experience and how it can impact consumers moving from the computer into the real world and back again are going to be highly desirable. Immersive virtual worlds are the next iteration of gaming and we look for creative talent in art, marketing, content, and production who understand that and embrace the change."

in you. Just do your research and (in the words of Thomas) "don't get left behind."

"There's a real market for hardcore games on these platforms, and with the ever-increasing quality and fidelity, it's becoming obvious that going forward if you want a long-term career in gaming, the time to learn the ins and outs of the social arena is now."

Added Fellbaum, "If you have what it takes to stand shoulder to shoulder with us then we will consider you just like anyone else—just bring yourself up to speed."

Thomas in turn recommended that students do their best to get as stuck-in as deeply as possible. "Get yourself behind the scenes to see how the sausage is made—intern at a social gaming start-up, understand what makes a social game successful, build your own games. Enthusiasm and demonstrated proactivity will often make up for a lack of experience when trying to nail that first job interview."

You can't, however, get discouraged if that first job interview doesn't end in success.



PHOTO COURTESY OF GAME DEVELOPERS CONFERENCE

Thomas added that Kabam was looking for candidates with a "unique blend of skills," such as "traditional gaming veterans with exposure to social gaming best practices" he added. "We also look for candidates with experience building and managing products with a 'live services' component and short development cycles. In our world, a feature can go from concept to development to launched and played by millions of players in weeks."


Of course, if you've been a developer in traditional video games previously and feel left out by that kind of description, it's not to say that the recruiters at GDC Online might not still be interested

Similar tips were offered to recent and upcoming graduates scouting GDC Online for their new career, with Barnes noting that (for example) CCP was at GDC Online to "be in the minds of new graduates and entry-level people" for future recruitment drives in customer support, quality assurance, and programming.

"Stay on top of trends," recommended Fellbaum. "Go to as many mixers and industry events and meet as many people as possible; keep your options as open as you can. All you need is that one foot in the door, that one helping hand, and then it's up to your skills and work ethic to see you the rest of the way."

Thomas admitted, "Job seeking is tough... Don't let rejection get you down; keep at it."

Fellbaum agreed, with the caveat that "the industry shifts all the time and jobs open up every day. You just need to be persistent and showcase yourself professionally."

Thomas concluded, "Don't be afraid to ask questions. Soak up as much as you can. You never know which interaction will lead to your dream job, so put yourself out there." Why not begin by putting yourself out there at GDC Online? 

**MATHEW KUMAR** is a Toronto-based freelance journalist, who also runs the exp. Magazine ([www.expdot.com](http://www.expdot.com)).





# SEPARATION ANXIETY

## INTERACTIVE GAME SCORES, ORCHESTRAS, AND YOU

The specter of the next generation looms on the horizon, and as work on strictly current-gen titles begins to set, the biggest achievement with regard to music in this waning console cycle has been the standardization of interactive scoring. Partially through iteration of proprietary tool sets and partially through the proliferation of middleware like FMOD and Wwise, interactive game scores have become the rule rather than the exception. Whether they're layered puzzle-solving sign posts as in *PORTAL 2* or beat-synchronized instrument lines in something like *LITTLEBIGPLANET 2*, composers continue to find new ways to make scores react to creative gameplay systems and player input.

Arguably the most complicated production process is interactive scoring centered around orchestral music. Rock, pop, and electronic scoring in 2011 all exist within the digital box of easily isolated Pro Tools stems. Orchestras, however, are by their very nature large groups of live musicians in a room preforming to produce one homogenous sound. Miked primarily by a central three-microphone Decca Tree with supplemental spot mics, a room full of live microphones is fundamentally contrary to the needs of isolated interactive stems.

There are approaches to orchestral stem recording, though, that can create all the flexibility needed while working in concert with the nature of orchestral scoring.

### DIVISI

» The first option is to divide your orchestra, and even this has two separate approaches with which composers and engineers can experiment. There is a tendency for interactive scores to frequently be composed of

beat-synched pieces all within the same key, the same tempo, or both. In fact, existing interactive music tools, such as those included as a part of Wwise, are built to easily facilitate this style of interactive design.

If your score is being recorded to a click and is predictably going to be the same tempo each time the cue is played, the first option for interactive stem creation is to multitrack your orchestra in sessions centered around individual instrument families. Just as you would multitrack vocals or guitars over a

rock combo's rhythm section, so can you multitrack different sections of the orchestra. Start with the section that will be playing the most notes, and will therefore be providing the most reference material for subsequent

sessions. This is most likely going to be your strings, with brass second in line for the stage. You'll probably want to include sampled reference stems in your Pro Tools sessions to cover the parts of the orchestra that haven't yet been performed.

On the plus side, you'll have perfectly isolated sections of each family of instruments. These clean stems can then be mixed either interactively or in post to create any number of alternate mixes. This approach has been used to great effect in the *GOD OF WAR* scores.

There are drawbacks to this strategy, though. An orchestra playing along to a click or sampled mock-up is not an exact science. Orchestras have a tendency to drag, and when playing in separate sessions, sections of the orchestra might drag at different rates, necessitating editing to tighten up their performances. Additionally, setup and teardown for different sessions means having to carefully plan which sessions occur on which days to avoid delays related to issues like mic setup or unwanted headphone bleed.

The other option for isolated recording is to have your orchestra perform at the same time, but in a space separated via a combination of baffles and isolation booths. Abbey Road, AIR, and Skywalker Sound's studios all have isolation booths connected to their main halls, though some are larger than others. Strings would have to be in the main space due to the sheer number of players, but smaller sections like winds and percussion could be isolated cleanly. Depending on the stage and the score, strings, brass, and woodwinds could all be isolated while percussion is tackled in post via samples.

This approach works well if the score isn't tied to a particular tempo, and there's a freedom of interpretation available with each take. The biggest hurdle with this approach is that, by removing sections from the larger live hall and moving them into tiny isolation boxes, the mix engineer will need to fake their homogeneity through digital reverbs.

### TUTTI

» The other option is to have everyone in the same space and simply record separate stems in turn. Recording cues together lets the players learn the piece, and it gets you a full version of the cue that you can use for in-game, reference, or soundtrack mixes. Then record the cue in sections—strings, low strings, high strings, brass, etc.—while those not playing sit silently in the room.

This approach is great for experimentation, as any performance changes or instrument combinations can be called out to the orchestra together as they are all in attendance. Some mix engineers prefer this approach because the players are all within the same physical space and digital fakery is kept to a minimum.

Having players sitting idly on the stage can be costly, though, if the session isn't strategized well. Idle players are also at a constant risk of accidental stage noise, which can ruin takes and waste both time and money. Additionally, be careful of anything that falls into the category of multitracked overdubs, meaning when a player plays over the top of something they've already recorded. This can incur an additional cost, depending on the musician's contract. 🎧

JESSE HARLIN has been composing music for games since 1999. He is currently the staff composer for LucasArts. You can email him at [jharlin@gdmag.com](mailto:jharlin@gdmag.com).



ILLUSTRATION BY KELSEY KRAUS

## 2011 GAME DEVELOPERS CONFERENCE EUROPE HOSTS MORE THAN 2,100 ATTENDEES



Richard Garriot.

PHOTO COURTESY OF GAME DEVELOPERS CONFERENCE

∞ The 2011 edition of GDC Europe concluded on Wednesday, August 17, with a successful showing and a final attendee count of more than 2,100 game professionals representing 57 countries. The show featured some 170 European and international speakers, a total of 46 exhibitors, and 35 sponsors, and more than 300 media representatives covering game development on PC, social networking sites, consoles, and mobile platforms.

Now in its third year, GDC Europe continued the tradition of bringing high-quality content to the European game development community. The introduction of the Social Games, Independent Games, Smartphone and Tablet Games, and Community Management summits demonstrated the increasing importance of these developing areas to European game development. The Social Games Summit also proved to be a forum that showcased content across both European

and Western independent developers including speakers Kellee Santiago from thatgamecompany, Frictional Games' Thomas Grip, and Spaces of Play's Marek Plichta.

Highlights of Day One at GDC Europe included wooga CEO Jens Begemann giving a keynote discussing today's social games and how they should not be designed to turn all core gamers into FARMVILLE farmers. In his talk, Begemann surprised attendees with the live, real-time launch of MAGIC LAND, a title that contains a mix of social gaming elements with aspects from the dungeon-crawling genre. Also featured was a keynote talk by Enric Alvarez, game director and co-author of CASTLEVANIA: LORDS OF SHADOW, who discussed how his Spain-based studio Mercury Steam met the challenge of taking on a beloved franchise from a Japanese developer.

Day Two of GDC Europe was headlined by Dr. Michael Capps, president of Epic Games, who discussed

lessons learned from the studio's development of games from Unreal to the GEARS OF WAR franchise to INFINITY BLADE, and also revealed that the studio is currently at work on five new unannounced titles. Cevat Yerli, president and CEO of Crytek, also gave a keynote at the conference on Tuesday. Yerli identified the challenges that Crytek faced when working on the development of the AAA online first-person shooter, WARFACE.

Day Three of GDC Europe was led by a keynote given by Richard Garriott, legendary creator of the genre-defining ULTIMA series and founder of social media games developer and publisher Portalarium, in which he focused on what he considers the three eras of gaming—single player, massively multiplayer, and casual and social—detailing the lessons learned from each.

GDC Europe also featured talks from video game veteran Mark Cerny, who commented on the shifting industry landscape; BRAID creator Jonathan Blow, who touched on philosophical aspects of game design and development; Fishlabs Entertainment's Marc Hehmyer, discussing taking an iOS-based game to Android; and Quantic Dream's Guillaume de Fondaumiere, illustrating the need for further evolution in game rating systems.

GDC Europe will return to Cologne, Germany on August 13–15, 2012.

## GDC China Honors Winners of Shanghai, Beijing Game Jams

∞ GDC China's organizers are proud to highlight the winners of the IGF China-sponsored Shanghai and Beijing Game Jams, both of which invited attendees to test their game design skills in a two-day marathon of collaboration and creative experimentation.

These two China-based events invited professionals, indie developers, and even hobbyists to form teams and create a working game prototype from scratch, using only their own tools and tech. To add a twist to the traditional Game Jam formula, each event also had its own theme around which entrants had to build their games: the Shanghai Game Jam's topic was "Resurrection," while the Beijing event's theme was "Utopia."

In addition, the events encouraged entrants to meet other developers and work with new team members, and GDC China—part of UBM TechWeb, as is this magazine—awarded Tutorials and Summits passes to the outstanding teams.

The following are the winning entries from both the Shanghai and Beijing Game Jams.

### SHANGHAI GAME JAM WINNERS

Topic: "Resurrection"

∞ **1ST PRIZE:** THE END IS NIGH

As a frustrated soul left behind in the Rapture, players must try to prevent their neighbors from ascending into Heaven, either by dragging them down or simply shooting them with a gun.

∞ **2ND PRIZE:** BUD

In this title, players control seeds that sprout and grow, bringing new life into a landscape torn apart by war, industry, and death.

### BEIJING GAME JAM WINNERS

Topic: "Utopia"

∞ **1ST PRIZE:** PAC-RESISTANCE

This PAC-MAN-inspired title has players navigating a dystopian city as they avoid police forces and take down buildings run by the oppressive government.

∞ **2ND PRIZE:** UTOPIA

Players must navigate the game's side-scrolling environments using clues from "Utopia," a poem by Polish Nobel prize winner Wisława Szymborska.

In addition to the above titles, many other games produced at both of these events are available for download via the official Shanghai Game Jam site and Beijing Game Jam site.

Registration for the show is now open, at the event's official website. This year, the Independent Games Festival China will return for its third year, and will complement the show's numerous tracks and summits.

GDC China will take place November 12–14, 2011, at the Shanghai Exhibition Center. @



# Cutting the Tether

## KEN TAYA JUMPS FROM BUNGIE TO 5TH CELL

*Ken Taya is an accomplished illustrator, known in the Seattle area for his posters and comics. In games, he worked as an environment artist at Bungie before moving to 5th Cell to expand his sphere of experience. We spoke with Taya about his influences, and the boons of working at a smaller company.*

**Brandon Sheffield:** *What made you decide to leave Bungie and then join 5th Cell?*

**Ken Taya:** I wanted to go to a company that reflected the wide range of styles that I also embody in my personal work. I also wanted to simply try something different.

**BS:** *At 5th Cell, do you have more autonomy and influence on the art style? Or now that HYBRID is coming, is it not much different?*

**KT:** At a smaller company, you undoubtedly have a relatively larger impact than at a big company, so yes. HYBRID is different from my previous work at Bungie because I got to be a part of exploring and defining HYBRID's new art style alongside each of my teammates. At Bungie the style had already been established, and we just worked to get it done. On HYBRID there was more of an exploratory phase I got to experience for the first time in my video game career. The exploratory stimulus you get from trying and then scrapping idea after idea is where the fun is.

**BS:** *5th Cell as a company has excelled at 2D art. Have there been challenges getting the team up to speed on newer techniques in the polygonal world?*

**KT:** HYBRID is 5th Cell's first 3D game, so it is challenging on many levels, but throwing yourself at challenges is a great way to grow. The Hybrid team is a bunch of relatively new hires that each have their own background in 3D games from previous companies, so our team has experience.

**BS:** *Your personal artwork and illustration is very different in style from what you've done in games. How do you view your personal art? Is it something to unwind with after working? Is it your true passion? An exercise to develop your skill?*

**KT:** My personal work is my own creative outlet to do exactly what I want. I love the autonomy of doing what I want without any pressure to earn money from it. I make stuff for me, assuming there are other people like me, rather than making stuff I think other people will like. This assures that I stay my most authentic self.

My online webcomic "i fart rainbow" ([www.ifartrainbow.com](http://www.ifartrainbow.com)) allows me to poke



fun at culture using a cute and colorful delivery.

I sell posters of my art at conventions such as Emerald City Comicon & Anime Expo, and also through a series of galleries like Kobo and CakeSpy in Seattle.

The satisfaction I get from working on games and having millions of people view my work, selling my personal artwork and having hundreds of patrons, and then simply drawing for my daughter and having only her truly enjoy it are essentially the same.

**BS:** *To look at your illustration work, it seems you've been influenced by Moebius and Masamune Shirou. Who would you claim your greatest influences are?*

**KT:** I have a lot of influences; Toriyama Akira, Tekkon Kinkreet, and Kozyndan to name a few. My work gets described as busy, colorful, and meaningful. I grew up reading manga and playing video games, so they both heavily influence my work, and allow me to make games as a profession.

**BS:** *Do you think there's more room for an illustrative style in games now, compared to the past?*

**KT:** There is always room for colorful illustrative works in games. You are seeing a bunch of these types of games being explored in many other XBLA titles.

## new studios

Gameforge (STAR TREK - INFINITE SPACE, GATES OF ANDARON) founder and former CEO Klaas Kersting has formed a new mobile games studio, flaregames, to develop games with real-world interaction.

Tokyo-based social and mobile gaming giant DeNA has announced plans to establish a new studio in Singapore, as a means of expanding its operations in Asia, with an initial capital of 500k Singapore dollars (\$410k).

Publishing veterans from Atari, Super-Ego, and Stormfront have formed a new downloadable game publisher called FreePlay Labs, which recently launched its first game based on the COWBOYS & ALIENS property.

Industry vet Aaron Cammarata, with credits on TONY HAWK'S PRO SKATER, PITFALL 3D, and multiple handheld titles, has formed AJC Games to merge mobile and social gaming into what he hopes will be a rainbow-filled, fungasmic paste.

## who went where

Free-to-play MMO company Outspark (DIVINE SOULS, 7 DRAGONS) has announced that its chief operating officer, Philip Yun, will take the reins of the company as CEO to replace departing founder Susan Choe.

MINECRAFT developer Mojang has added a new member to its team, announcing that Henrik Pettersson has joined the staff as a graphic artist.

Zipline Games, the studio behind the Lua-based Moai mobile game development platform, has appointed Shane Kim, formerly Microsoft Game Studios' vice president and overseer of franchises such as HALO and GEARS OF WAR, to its board of directors.

RIFT studio Trion Worlds has hired Xbox veteran David Luehmann as VP of third-party development, where he will work with external developers and publishers who can take advantage of Trion's proprietary online platform and technologies.



# A Flipping Good Time

EARLIER THIS YEAR, A STUDENT TEAM AT THE DIGIPEN INSTITUTE OF TECHNOLOGY DEBUTED A FLIPPING GOOD TIME, A CHALLENGING 2D PLATFORMER THAT LETS PLAYERS MANIPULATE GRAVITY TO NAVIGATE A SERIES OF COMPLEX UNDERGROUND CAVERNS. WE RECENTLY SPOKE WITH THE TEAM BEHIND THIS PAX 10 SHOWCASE WINNER TO LEARN MORE ABOUT HOW THE PROJECT CAME TO BE.



**TOM CURTIS:** *How did you arrive at the overall design concept for the game?*

**RICHARD WESCHLER** (game concept, game designer, level designer):

Being big fans of classic platformers such as YOSHI'S ISLAND and DONKEY KONG COUNTRY, we really wanted to make a game that captured the traditional feel and fun of those games but with some sort of twist. The original design had a weight-changing mechanic that allowed the player to change the weight of the main character depending on different-sized ore he picked up. The player's current weight could be tracked with a weight bar, but after prototyping this concept, we quickly realized this was just too much for the player. The mechanic was streamlined to a simple flipped or non-

flipped state which made the player lighter when he picked up a cape.

**TC:** *What tools did you use?*

**RYAN DAVIDSON** (physics programmer, art, animator): Our development environment was Visual Studio. The game itself was written in C++, and we used DirectX 9 for graphics and input. External tools included Photoshop, a school-hosted SVN, and of course, the support of our fellow students and DigiPen.

**TC:** *What were some of the technical hurdles you had to overcome?*

**MARK MCKENNA** (lead programmer): There were plenty of technical issues that had to be resolved throughout the development process. Our level editor, for example, was a bit of a challenge,

and never did work quite right. There were also quite a few little things like camera movement, player movement, and the death animation that took several iterations to get to their current state.

One area we put a great deal of time and effort into was the drawing of the environment. Originally, we just needed to get something drawn so we could start creating levels, playtesting, and developing new features. In our early builds, the levels had a really ugly checkerboard sort of look. From there, without abandoning our simple tile-based approach to level representation, we developed some techniques for mapping larger textures over the environment, and trimming the borders between different tile

types to soften hard edges. The result was a reasonable balance between visual appeal and content creation costs.

**TC:** *What sort of design challenges did you encounter during development?*

**STEPHEN FOGG** (graphics designer, level designer): One of the biggest challenges we faced revolved around the difficulty of the game. Are we targeting the casual platformer? Do we go after the hardcore SUPER MEAT BOY fans? How soon and at what points do we ramp up the craziness? We decided that we wanted a game that someone entirely new to video games could sit down and have a good time with, but we also wanted to appeal to those individuals who

crave an insane amount of challenge from their platformers. Our solution was to create two tracks of levels: a normal track that would introduce all the various gravity-flipping mechanics while being easy enough for a beginner to get through, and then a challenge track of levels that would only unlock with the collection of gems hidden within the normal track. The more the players proved themselves capable in the normal track levels, the more comfortable we were in allowing them to experience the higher difficulty of the bonus content.

The other major design challenge was sticking to a core concept and avoiding feature creep. Fairly early on in development we knew from our prototype that we needed to have gravity balls, pads, and beams. Beyond those, we had a gigantic list of ideas that we thought might work well in the game: springboards, wind, water, break-away platforms, moving platforms, timed gravity, enemies, bumpers, and so on. Ultimately, 90 percent of these ideas had to be scrapped because they either clashed with the already established feel of the game, or we simply didn't have the time during the school year to implement and experiment with them. We had to shoot for the best

(or perhaps the simplest) ideas, and experiment with those select few. The mine cart proved to be both. Not only was it fairly easy to implement (we just force the player in a certain direction while in the cart), but also provided some of the most exciting gameplay moments.

**TC:** *What process did you use to tune the difficulty of your game, and how long did it take? Some of those levels are really tough!*

**RW:** It was a very long process. Coming right out of the gate with a prototype, difficulty was something we were keeping a close eye on. What we found is that players really enjoyed the free-flowing movement of the game and had a lot of fun simply flipping through our environments, but at the same time, we also knew we wanted to make the game fairly challenging. So we came up with the idea that we would have both normal levels and challenge levels. That really helped in that it allowed us to create an enjoyable experience for a casual player, but then gave us the chance to ramp it up for the more hardcore player. Around the second half of the development process we were playtesting almost every week, and almost every playtest revolved around the difficulty and fun factor of specific levels. Levels went through many iterations, and by the end, we had a solid grasp on what was fun yet still fell into the difficulty range we wanted.

**SF:** It was mostly playtesting and iteration. As soon as we reached our first playable milestone, we immediately put some beginning levels together

and put them to the test. Every Friday we ran an official playtesting session at DigiPen, asking fellow students and teachers to have a go at our game. We didn't stop there though. Every member of the team took the game outside of school to our friends and family so that we might gather feedback from a wider range of player types and skill levels. We asked for everyone's feedback and tried to actually watch each tester's playthrough. We took note of where players often died, which puzzles were too difficult, where the game pacing lagged a bit, whether the mechanics were clear, what players would like to see added to the game, and things like that. Using this feedback, Richard and I would then go back into the level editor and revise what we had. For every level you see in the finished game, there may be twice as many that were scrapped, modified, or cherry-picked from. Each level received multiple passes to get the overall path, add obstacles, add items, place fireflies, and then fine-tune problem areas found during playtesting. The process was never-ending, and continually cycled throughout the spring semester and into the summer.

**TC:** *Your game certainly seems to draw a lot of inspiration from indie platformers like WVVVV and SUPER MEAT BOY. Can you talk a bit about your influences and how they affected the game?*

**RW:** In all honesty, no one on the team played WVVVV when we started making the game. It wasn't until

about halfway through the development process when enough people brought up WVVVV that I sat down and played it. I really enjoyed the game a lot and was blown away! The game has such simple mechanics, but they are used in such creative ways. It taught me to look at our mechanics and to try and stretch those out as far as we can. SUPER MEAT BOY was something I brought up to the team a number of times. I wanted to have levels that were both challenging but fair. SUPER MEAT BOY gets away with having such a high difficulty due to the fact that its levels are so short, and in A FLIPPING GOOD TIME we made sure to kind of do the same thing using checkpoints. There is always a checkpoint before and after every difficult part. Granted, our game doesn't get nearly as hard.

have had an editor and art pipeline up and running much sooner. Eventually we made up for it, but it wasn't until the jump from the Alpha to Beta milestones that we really started flowing.

**TC:** *How does it feel to have your game showcased in the PAX 10? Do any of you have plans to expand upon the title now that it has gotten such exposure?*

**KL:** Being showcased at PAX definitely reassured us that we were on the right track. The future doesn't include A FLIPPING GOOD TIME as of now. However, if we made a second version it would be in 3D and titled "A Flipping Good Time 2 3D" in hopes that people would read it as 23D and wonder how we managed so many dimensions.

**SF:** It feels incredible to be recognized by PAX. Being a part of this team

of work, and watch them enjoy playing it. And now we are lucky enough to show our game to so many passionate gamers who might otherwise have never have seen it. PAX has given us the chance to showcase our game beyond the confines of a typical student project, and we are incredibly grateful for that opportunity.

As for whether we will expand on this in the future—who knows? The team currently has our sights set on starting fresh on our junior year game. However, we like to think of A FLIPPING GOOD TIME as a resounding proof of concept. We know we tapped into some amazing gameplay with this project, and perhaps in the future we may have the opportunity to further explore the possibilities.

**RW:** It feels amazing. Being a big fan of Penny Arcade and PAX; this is a dream



**TC:** *Now that you've finished the game, is there anything you would have done differently during development?*

**KENNETH LOMBARDI (producer, graphics programmer):** Considering our game is very dependent on the content and visuals, we should

have been an amazing learning experience for everyone involved and we couldn't be more proud of what we were able to create in this game. It is so absolutely rewarding to have someone pick up something you've helped fashion and create through hours and hours

come true for me. If DigiPen wasn't such an intense school I would love to expand on A FLIPPING GOOD TIME, but we will most likely be very busy working on next year's game.

**RD:** I came to this school to be part of something great, and that is exactly what happened. 🍄

# **Ai** The Art Institutes®

## CREATE TOMORROW

### **A SERIOUS LEVEL OF OPPORTUNITY.**

Create your tomorrow with the Game Art & Design programs at The Art Institutes.

**Go from game player to game developer.** Imagine having a choice of just one video game. In one dimension. With one landscape, one set of characters, dull graphics, and clunky mechanics. Well, if it wasn't for people with a love of gaming, a head full of ideas, and a way with technology, that might be the all-too-real world of gaming.

Of course, the reality is that every day seems to bring new consoles, new mobile gaming devices, and new technologies. And if you want to join the creative minds who keep pushing the industry forward, your first move is a focused education at an Art Institutes school. We understand how creative people think. And we know how to help you develop your talents so you can make the leap from game player to gaming professional.

We're your connection to this dynamic and creative industry.

**Bring us your creativity.** We'll help you explore the possibilities. In our Game Art & Design programs, you'll find yourself in a creative community, collaborating with other students who share your passion. And instructors who work in the same fields they teach. Which means they know where the industry is and where it's heading – and the demands of the career you're preparing for.

You'll start with the fundamental courses, then move on to building your skills in drawing, color, design, computer applications, and creating lifelike characters. Then comes image manipulation, cinematography, creative storytelling, storyboarding, and 2D and 3D modeling techniques, featuring professional-grade technology – including Apple and HP workstations; Microsoft XNA, Unity 3D, and Unreal SDK game engines; and software including Adobe Master Collection and Autodesk Entertainment Creation Suite.

**Get ready to share your ideas with the world.** As you prepare to enter the job market, we'll help you assemble a digital portfolio to showcase your talents – both on your own and at portfolio shows, where you can share your work with companies familiar with our program and our graduates. It's all about getting your foot in the door with an entry-level job such as game tester/analyst, game designer, level designer, texture artist, cinematic artist, 2D artist, or 3D artist in a software, game design, or education company.

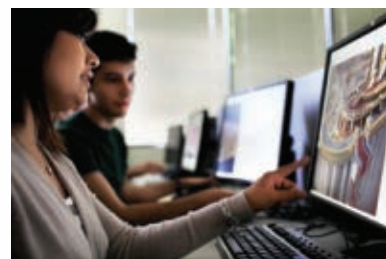
**Talk to us today. And prepare to create tomorrow.** Contact us and we'll help you get started toward a career in Game Art & Design.

The Art Institutes is a system of over 45 schools throughout North America. Financial aid is available to those who qualify.

Program offerings, technology, and credential levels vary by school.



*You've got ideas for game characters. Getting them from your head onto consoles starts with fundamentals like drawing, color, and design.*



*With your creativity and our technology, you can develop the talents you need to launch a future in game design.*



ART SCHOOL

#### → THE ART INSTITUTES

Administrative Office  
210 Sixth Avenue, 33rd floor  
Pittsburgh, PA 15222  
1.800.894.5793

**Gamer.ait.edu**

# TURN YOUR PASSION FOR GAMING INTO A CAREER

## Game Art

Bachelor's Degree Program  
Campus & Online

## Game Design

Master's Degree Program  
Campus

## Game Development

Bachelor's Degree Program  
Campus

## Game Design

Bachelor's Degree Program  
Online



### Campus Degrees

- Master's**
  - Entertainment Business
  - ▶ Game Design
- Bachelor's**
  - Computer Animation
  - Creative Writing for Entertainment
  - Digital Arts & Design
  - Entertainment Business
  - Film
  - ▶ Game Art
  - ▶ Game Development
  - Music Business
  - Recording Arts
  - Show Production
  - Sports Marketing & Media
  - Web Design & Development
- Associate's**
  - Graphic Design
  - Recording Engineering

### Online Degrees

- Master's**
  - Creative Writing
  - Education Media Design & Technology
  - Entertainment Business
  - Internet Marketing
  - Media Design
  - New Media Journalism
- Bachelor's**
  - Computer Animation
  - Creative Writing for Entertainment
  - Digital Cinematography
  - Entertainment Business
  - ▶ Game Art
  - ▶ Game Design
  - Graphic Design
  - Internet Marketing
  - Mobile Development
  - Music Business
  - Music Production
  - Sports Marketing & Media
  - Web Design & Development



**FULL SAIL**  
UNIVERSITY.

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

Winter Park, FL

800.226.7625 • 3300 University Boulevard

Financial aid available to those who qualify • Career development assistance • Accredited University, ACCSC  
To view detailed information regarding tuition, student outcomes, and related statistics, please visit [fullsail.edu/outcomes-and-statistics](http://fullsail.edu/outcomes-and-statistics).

Architectural rendering of the new Centre for Digital Media to open fall 2012.

## PREPARING THE NEW LEADERS OF THE DIGITAL MEDIA INDUSTRY

The Masters of Digital Media program (MDM) is Canada's first professional graduate degree program of its kind in digital media and entertainment technology.

Offered at Vancouver's Centre for Digital Media, this 16-month program and internship engages students in real world projects where they gain valuable leadership experience, hands-on training, and top industry connections.

Find out more.  
[mdm.gnwc.ca/leaders](http://mdm.gnwc.ca/leaders)

### CENTRE FOR DIGITAL MEDIA

MASTERS OF DIGITAL MEDIA PROGRAM | GREAT NORTHERN WAY CAMPUS

Our Academic Partners:

# 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 Aldo Martinez-Gonzalez

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

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

DAVID BOWRING, GAME DESIGN GRADUATE  
 GAMEPLAY DESIGNER, SAINTS ROW 2

## ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
Academy of Interactive Entertainment	15	Masters of Digital Media Program	53
Blizzard Entertainment	3	Rad Game Tools	C4
Epic Games	12	RCCD Norco College	45
F+W Media Inc.	15	Seapine Software Inc.	C2
Full Sail Real World Education	53	The Art Institutes	52
Havok	C3	Vancouver Film School	54

*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 United Business Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is \$49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: \$69.95; all other countries: \$99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *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.



## GET YOUR GAME ON!

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

- News and features for students and educators
- GETTING STARTED section – an invaluable how-to guide
- Message Boards



### DIGITAL COUNSELOR

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

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



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

### UNITED STATES POSTAL SERVICE Statement of Ownership, Management, and Circulation

1. Publication Title: Game Developer. 2. Publication No.: 13782. 3. Filing Date: August 30, 2011. 4. Issue Frequency: Monthly with a combined June/July issue. 5. Number of Issues Published Annually: 11. 6. Annual Subscription Price: \$49.95. 7. Complete Mailing Address of Known Office of Publication (Not Printer): UBM LLC, 303 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107. Contact Person: Roy Beagley. Telephone: 203-775-9465. 8. Complete Mailing Address of Headquarters or General Business Office of Publisher (Not Printer): UBM LLC, 303 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107. 9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor: Publisher: Simon Carless, UBM LLC, 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107; Editor: Brandon Sheffield, UBM LLC, 303 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107; Managing Editor: None. 10. Owner: UBM LLC, 600 Community Drive, Manhasset, NY 11030-3875, an indirect, wholly owned subsidiary of UBM LLC, Ludgate House, 245 Blackfriars Rd., London, SE1 9UY, U.K. 11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities: None. 12. Tax Status: Has Not Changed During Preceding 12 Months. 13. Publication Title: Game Developer. 14. Issue Date for Circulation Data Below: September 2011.

15. Extent and Nature of Circulation:	Average No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total No. Copies (Net Press Run)	24,934	26,148
b. Paid and/or Requested Circulation		
(1) Outside County Paid/Requested Mail Subscriptions Stated on Form 3541	16,203	18,067
(2) In-County Paid/Requested Mail Subscriptions Stated on PS Form 3541	0	0
(3) Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid or Requested Distribution Outside USPS	1,861	2,054
(4) Requested Copies Distributed by Other Mail Classes Through the USPS	0	0
c. Total Paid and/or Requested Circulation [Sum of 15b. (1), (2), (3), and (4)]:	18,064	20,121
d. Nonrequested Distribution [By Mail and Outside the Mail]		
(1) Outside County Nonrequested Copies Stated on PS Form 3541	4,053	3,949
(2) In-County Nonrequested Copies Stated on PS Form 3541	0	0
(3) Nonrequested Copies Distributed Through the USPS by Other Classes of Mail	0	0
(4) Nonrequested Copies Distributed Outside the Mail [Pickup Stands, Trade Shows, Showrooms, and Other Sources]	2,455	1,800
e. Total Nonrequested Distribution	6,508	5,749
f. Total Distribution (Sum of 15c and 15e)	24,572	25,870
g. Copies Not Distributed	262	278
h. Total (Sum of 15g and 15h)	24,934	26,148
i. Percent Paid and/or Requested Circulation [15c Divided by 15f Times 100]	73.51%	77.28%

16. Publication of Statement of Ownership: This Statement of Ownership will be printed in the October 2011 issue of this publication. 17. Signature and Title of Editor, Publisher, Business Manager, or Owner [signed]: Simon Carless, Date: August 30, 2011.



# SANDWICH PITCHES

## WILL YOUR LUNCH GET GREENLIT FOR PRODUCTION?

### ROUND 1: TURKEY, HAM, AND SWISS

*"The new Turkey, Ham, and Swiss sandwich will add the delicious flavor of turkey to the already popular Ham and Swiss product. Eaters across the globe will thrill at the addition of a new ingredient—turkey—to this already proven combination. Turkey, Ham, and Swiss will appeal to established fans of Ham and Swiss sandwiches, as well as anyone else looking for a good sandwich."*

Thanks for your pitch. If we're not mistaken, all you've done is taken a Ham and Swiss sandwich and added turkey to it. "More of the same" works sometimes, but there isn't really a lot for consumers to get excited about here. I'd encourage you to think about what really differentiates your sandwich from the other lunch options in the marketplace today. We're not just competing with other sandwiches anymore; in these unprecedented times, people are just as likely to eat hot dogs or even pizza.

We appreciate the time you took to put this together—come back to us once you have a better hook.

### ROUND 2: THE MINDBLOWER

*"Developed in secret for over 12 years by a small team of sandwich industry mavericks, The Mindblower will make consumers rethink everything they know about sandwiches! The Mindblower's unique combination of tahini, pickled okra, and Pop Rocks in a tomato basil wrap will punch past the jaded, cynical mind of the average sandwich consumer and hit them straight in the gut!"*

That's a very interesting approach. We definitely responded to the boldness of your design, and we like that you're willing to take on some risk in pursuit of innovation here. We can see the appeal of getting customers to break out of the "been there, done that" sandwich routine and experience something new. However, there do seem to be certain unproven elements in the design, which are a cause for concern. Particularly, we zeroed in on how the reddish color of the tomato basil wrap might put off consumers who aren't used to eating light red or pinkish foods.

Additionally, because this concept is so off-the-wall, you may want to think about allowing people the opportunity to customize certain elements of the experience to their own preferences. Maybe somebody wants the Mindblower sandwich except on rye bread instead of in a wrap, for example. User customizability is, of course, a big part of our strategy right now.

### ROUND 3: METASANDWICH

*"Metasandwich is much more than a sandwich—it's actually a powerful and easy-to-use meal-creation platform. Our dynamic sandwich-assembly system, The Sandwich Counter™, allows customers to arrange and rearrange their own custom sandwiches on the fly—in real time. Don't like avocado? Don't put avocado on there. Go crazy and add two slices of cheese, or an extra piece of lettuce. Then, share your personal sandwich creations with your friends using the internet! Metasandwich is lunch for the remix generation."*

Okay, we get this. We like the user-generated content strategy here, and the idea that every customer can have the experience they want. It sounds like something that could really take off and sustain itself once the community gathers steam.

At the same time, our design director brought up a good point: We would need to support all the millions of possible combinations that this system

would allow. What if someone didn't put any meat on their sandwich and then had a bad experience? I think we'd have to build in some controls—have some of the elements of the sandwich customizable, and others not—so that no matter what they do, users are always left with a good-tasting sandwich. We'll also want to put a disclaimer on there saying that modifications to the sandwiches are not necessarily endorsed by us and that we can't guarantee anything about how they taste. I've CCed our legal department here so they can offer some additional suggestions on how to move forward with this.

In the meantime, the community aspect of the pitch was intriguing. You mentioned sharing sandwich creations with friends—that reminded us a lot of Twitter and Facebook! It'd be great if you could push more of those social aspects into the concept.

### ROUND 4: SANDWICHTOWN

*"SandwichTown is a connected entertainment experience designed around the wide-ranging fun of making and eating sandwiches. In SandwichTown, customers are challenged to assemble their own sandwich by rounding-up and coordinating groups of friends to perform various tasks, such as gathering wheat, kneading dough, and baking loaves of bread. Advanced sandwich options such as peanut butter or diagonal cuts are gated through a series of microtransaction tiers."*

*Our pioneering 'sandwich as a service' model allows us to upsell a premium experience at every step of the*



*sandwich-creation process. The Pro Basic monthly subscription level gives lunch customers access to a condiment bar with unlimited mustard, exclusive sneak peeks at upcoming*

*ingredients, and special blue plastic trays instead of ordinary brown ones. The Pro Premium level offers an even more enhanced level of service (exact features TBD). Try SandwichTown today—all your friends are doing it!"*

Yes, we like where you've gone with this. It feels like sandwich-as-sticky-app (and we don't mean because of the grape jelly, ha ha). How soon do you think you could get together a proof-of-concept dynamic sandwich engine up and running? Have you thought about integrating some sort of cloud strategy?

Just one note before we start production: While we do think of this as the way forward, a concern was raised about the core audience and the perception that we might be dumbing down the sandwich experience. Of course we want a big, diverse audience, but we don't want to alienate hardcore sandwich-eaters. Just make sure you've got a value proposition for those customers, too. Maybe in addition to all this, you could also offer something like, I don't know, maybe a simple Ham and Swiss? 🍷

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

# Get your game performing like a scripted masterpiece.



## havok™ Script

*The smallest, fastest LUA compatible virtual machine designed specifically for game development*

Havok™ Script armed Relic Entertainment® with a robust toolset, optimized engine, and never-before-seen visibility into their script-related performance and memory changes for the development of Warhammer® 40,000®: Space Marine®.

With these weapons at their disposal, Havok Script was able to help Relic realize their gameplay vision, ensuring each Ork will meet his fate against the chainswords of the Space Marines as efficiently as the last.

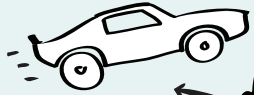
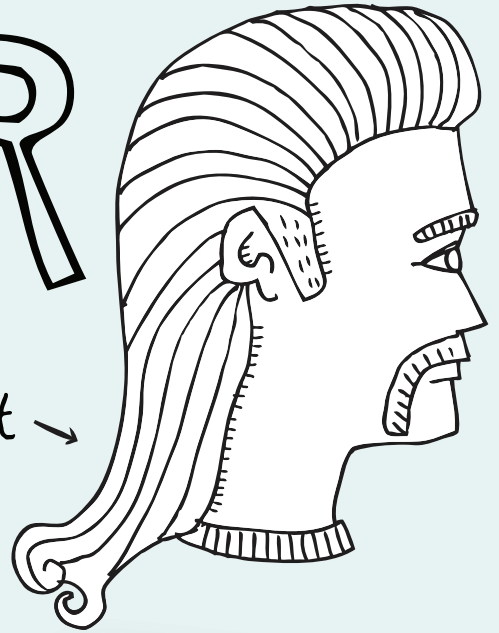
Learn more: [www.havok.com/havok-script](http://www.havok.com/havok-script)



Warhammer 40,000: Space Marine -- Copyright © Games Workshop Limited 2011. Space Marine, the Space Marine logo, GW, Games Workshop, the Games Workshop logo, 40K, Warhammer, Warhammer 40,000, Warhammer 40,000 Device, 40,000, the Double-headed Eagle device and all associated marks, logos, places, names, creatures, races and race insignia/devices/logos/symbols, vehicles, locations, weapons, units and unit insignia, characters, products, illustrations and images from the Space Marine game and the Warhammer 40,000 universe are either ®, TM and/or © Games Workshop Ltd 2000-2011, variably registered in the UK and other countries around the world, and used under license. All Rights Reserved. Uses RVD2 Library v2.0: © University of North Carolina at Chapel Hill. All rights reserved. Developed by Relic Entertainment. THQ, Relic Entertainment and their respective logos are trademarks and/or registered trademarks of THQ Inc. All rights reserved. All other trademarks, logos and copyrights are the property of their respective owners.

THERE ARE LOTS OF WAYS TO BE AN EVEN

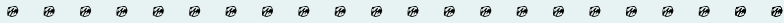
# COOLER GAME DEVELOPER



like grow a mullet →

← drive a sweet old Camaro

or you can use **BINK VIDEO**.



You get an amazing, super **FAST** video and audio codec - all in a simple, clean API. And Bink Video

runs on **EVERY PLATFORM!**



USING BINK VIDEO NOT ONLY MAKES YOU cooler, IT MAKES YOU rad!



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