

halo  
evolved  
anniversary  
gdm

combat  
evolved  
anniversary

GAME DEVELOPER MAGAZINE



smoke out  
the cheaters  
with the  
cloud

halo combat  
evolved  
anniversary  
postmortem

# Intel® GPA<sup>†</sup> helps Zombie Studios deliver *Blacklight\*: Retribution*

## ZOMBIE STUDIOS OPTIMIZES WITH INTEL® GRAPHICS PERFORMANCE ANALYZERS

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

*"Ultimately, the results of optimizations we did with Intel® GPA tools made the Blacklight\*: Retribution experience better and better for our players."*

— CHANCE LYON, LEAD DEVELOPER,  
ZOMBIE STUDIOS

### GPA SYSTEM ANALYZER

- Quickly analyze game performance and identify potential bottlenecks
- A heads up display (HUD) for real-time performance analysis
- Create DirectX\* state overrides and conduct real-time experiments
- Triage system-level performance with CPU and GPU metrics
- Game pause, step, resume

### GPA FRAME ANALYZER

- Optimize graphics performance through deep frame analysis of elements at the draw-call level
- Support for Microsoft DirectX\* 9/10/11, Microsoft Windows\* XP, Microsoft Vista\* 7
- Make changes and see visual and performance effects immediately
- Capture and share frames

### GPA PLATFORM ANALYZER

- HW thread view of game showing thread context
- Visualize performance of multi-threaded CPU and GPU tasks
- Intel® Media Analyzer for real-time and trace analysis
- Intel® OpenCL and browser profiling abilities

**DOWNLOAD INTEL® GPA FOR FREE »**  
[www.intel.com/software/gpa](http://www.intel.com/software/gpa)



# WHY DON'T YOU PLAY?



With all the tools and resources you need in one place, developing games for the BlackBerry® PlayBook™ tablet has never been easier — or more fun. Create rich, high-performance C/C++ apps and games with the BlackBerry® Native SDK for Tablet OS.

© 2012 Research In Motion Limited. All rights reserved. BlackBerry®, RIM®, Research In Motion® and related trademarks, names and logos are the property of Research In Motion Limited and are registered and/or used in the U.S. and countries around the world. All other trademarks are the property of their respective owners.

**START PLAYING**

[blackberry.com/gamedeveloper](http://blackberry.com/gamedeveloper)



# gd

h a c o m b  
e v e d  
a n n i v e r s a

THE LEADING GAME INDUSTRY MAGAZINE  
THE LEADING GAME INDUSTRY MAGAZINE VOL 19 NO 3  
MARCH 2012 INSIDE: DESTROY YOUR  
MARCH 2012 INSIDE: DESTROY YOUR VEHICLES!

DEVELOPER MAGAZINE

## CONTENTS.0312

VOLUME 19 NUMBER 3

### DEPARTMENTS

- 2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]  
A Confident Fall
- 4 **HEADS UP DISPLAY** [NEWS]  
Austin developers discuss the future of games, and the origins of Kyoto's 1H1D!! show.
- 30 **TOOL BOX** *By Mike de la Flor* [REVIEW]  
Wacom Inkling
- 33 **THE INNER PRODUCT** *By Remi Quenin* [PROGRAMMING]  
The SPUs are Hungry!
- 38 **PIXEL PUSHER** *By Steve Theodore* [ART]  
The Happy Cog
- 40 **THE BUSINESS** *By Kim Pallister* [BUSINESS]  
Copyright
- 42 **DESIGN OF THE TIMES** *By Damian Schubert* [DESIGN]  
Random is Random
- 45 **AURAL FIXATION** *By Damian Kastbauer* [SOUND]  
It's About People
- 47 **GOOD JOB** *By Brandon Sheffield* [CAREER]  
Q&A with Darren Hayward, who went where, and new studios
- 49 **GDC NEWS** *By Staff* [NEWS]  
GDC special awards revealed
- 50 **EDUCATED PLAY** *By Tom Curtis* [EDUCATION]  
DEITY
- 56 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]  
How To Annoy Your Testers

### POSTMORTEM

- 20 **HALO: COMBAT EVOLVED ANNIVERSARY**  
HALO: COMBAT EVOLVED represented a watershed moment for the Xbox. The design, multiplayer, and high-end graphics helped the game become a cultural phenomenon. And so, to celebrate the original title's tenth anniversary, Saber Interactive, 343 Industries, and Certain Affinity all worked together on a remake. Saber took on the brunt of the work, but even though the design and much of the code was fixed in place, it was no walk in the park. From collision issues to artistic choices, this postmortem is a primer on how to do a proper HD remake of an already-HD game. *By Andrey Iones*

### FEATURES

- 7 **CHEATING BEHIND THE CLOUD**  
Cheating is a problem in most online games, especially those with synchronous play. Microsoft's Ferdinand Schober discusses how you can implement a custom cheat-resistant cloud server to mitigate cheating without paying for anything but programmer time. *By Ferdinand Schober*
- 13 **GET SMASHED!**  
The challenges of collision in open-world games are legion. When players can go anywhere and do anything, you have to be ready for anything. But in SAINTS ROW: THE THIRD, Volition wanted drivable tanks to be able to crush objects, including other vehicles. Volition programmer Victor Cepeda IV outlines the visual deformation techniques from SAINTS ROW 1-3 to demonstrate how the team conquered this problem. *By Victor Cepeda IV*



# A CONFIDENT FALL

KNOWLEDGE-SHARING, THE RISE OF WESTERN GAME DESIGN, AND THE THREAT OF RUINING IT ALL

Western games have taken over the console game space, there is no denying it. It was tough to imagine this back in the late '80s and early '90s, when the common feeling was that all the best console games came from Japan, while Western companies churned out poor licensed titles and jittery platform games.

While Western companies may have struggled to match the design and vision of Japanese games on the console in those early days, they were simultaneously innovating on the PC for as long as the format existed. *QUAKE* might not have been a huge leap over the graphics of its PlayStation contemporaries, but by the end of the '90s, the PC was far and away the most graphically advanced platform available. And it was Western companies that were pushing this graphical revolution. Perhaps the variety wasn't there in the design and direction yet, but the tech was becoming peerless. And as tech becomes more of a solved problem, the focus turns to design. When this happened in the West, developers started to really push forward in this arena as well. Games like *HALO* proved that Western game design and tech had married to create a cultural phenomenon.

By the time *GEARS OF WAR 3* came out, nothing in Japan could match it visually, and that country's industry has not, to date, caught up. In 1990, would you have predicted that *UNCHARTED* would be made in Los Angeles, instead of Tokyo?

## HELPING HAND

» Even back then, there might have been reason to believe in the Western game industry. Starting in 1988, game developers began to meet to discuss the art of game creation at what would later become the Game Developers Conference. Developers shared their technical hurdles, and design quandaries. In 1994, *Game Developer* magazine was born, sharing technical knowledge in an even wider [and more frequent]

sense. In 1995, the organization now known as the IGDA came to be. Then in 1996, there was *Gamasutra*. Western game developers have been sharing knowledge with each other in some official forum or another for 24 years—and Eastern developers have yet to catch up.

Ray Nakazato (former president of *LOST ODYSSEY* developer FeelPlus, now at THQ in Japan) agreed with this sentiment in an interview we conducted a couple years back. "I'm worried about technology, Japan is already quite behind, and will be more behind," he said. "A lot of good technology is coming from the States. You have [events like GDC]. I was a founding member of CEDEC [Japanese development conference put on by CESA, which is akin to the ESA in the U.S.], and I wanted to make it the GDC of Japan, but CESA saw it differently. So it became CEDEC. It's modeled after GDC, but it's very different."

"Japanese people are not good at speaking," he adds. "They are afraid of disclosing things, so all the sessions are very vague and generic, and all are sponsored sessions, or academic. I think Japanese industry people want to hear a lot of stuff from people who are actually making games, but they don't speak much, so all they hear is the academia or sponsored messages."

CEDEC has gotten better, and Japanese developers do talk in a more casual sense about their problems, typically over drinks and the like. Still, official sharing of information across companies, and sometimes across teams within companies, is often frowned upon. This has led to Japanese companies solving problems individually, instead of collectively, and falling behind as a result. Western companies rocketed forward, propelled by their openness with each other.

## SPRING FORWARD, FALL BACK

» Why bring all this up now? Because, for whatever reason, as

indies and new media developers get more open, the big publishers are getting more tight-lipped. Though the developers of *CALL OF DUTY*, *FALLOUT NEW VEGAS*, and even *DRAGON AGE LEGENDS* on Facebook would have loved to share postmortems and technical pieces in this magazine, their publishers put the kibosh on their efforts.

Some publishers have a "no public postmortems" policy, across the board, while others are simply wary of sharing tech secrets. Others are willing to allow third-party publishing games to have postmortems, but not first-party. I would caution any publisher or developer that is reluctant to share (or in fact is flat-out against the concept) to learn from the above history.

Indie, social, and mobile games are making great strides, as the tools to support them get cheaper and more robust. And they actually make up a huge chunk of the market. Indie game bundles are selling in the millions of dollars, and app developers on iTunes have collectively made \$4 billion [though some of those are assuredly not games]. Zynga just went live with its huge IPO.

Publishers may be concerned that these burgeoning markets will encroach on their sales, as they try to buy their way into these industries. Makers of games in similar genres may be concerned that revealing the "secret sauce" will allow their competitors to get an edge. And there may be some truth to both of these. But transparency is good for the industry, and walled gardens only isolate and compartmentalize.

If big publishers (and big developers) want to succeed in this rapidly-changing market, they will have to learn what the Japanese game industry is still trying to catch up to; the more we help each other, the better we all become.

—Brandon Sheffield  
twitter: @necrosotoy



UBM LLC.  
303 Second Street, Suite 900, South Tower  
San Francisco, CA 94107  
t: 415.947.6000 f: 415.947.6090

## SUBSCRIPTION SERVICES

FOR INFORMATION, ORDER QUESTIONS, AND ADDRESS CHANGES  
t: 800.250.2429 f: 847.763.9606  
e: [gamedeveloper@halldata.com](mailto:gamedeveloper@halldata.com)

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

## EDITORIAL

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

**DESIGNER**  
Cliff Scorso e: [cliff.scorso@ubm.com](mailto:cliff.scorso@ubm.com)

## CONTRIBUTING WRITERS

Tom Curtis  
Mike de la Flor  
Remi Quenin  
Kim Pallister  
Damian Kastbauer  
Damion Schubert  
Victor Cepeda IV  
Matthew Wasteland  
**ADVISORY BOARD**  
Mick West Independent  
Brad Bulkley Microsoft  
Clinton Keith Independent  
Brenda Brathwaite Loot Drop  
Bijan Forutanpour Sony Online Entertainment  
Mark DeLaoura THQ  
Carey Chico Globex Studios  
Mike Acton Insomniac

## ADVERTISING SALES

**GLOBAL SALES DIRECTOR**  
Aaron Murawski e: [amurawski@ubm.com](mailto:amurawski@ubm.com)  
t: 415.947.6227

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

**GLOBAL ACCOUNT MANAGER, RECRUITMENT**  
Gina Gross e: [ggross@ubm.com](mailto:ggross@ubm.com)  
t: 415.947.6241

**GLOBAL ACCOUNT MANAGER, EDUCATION**  
Rafael Vallin e: [rvallin@ubm.com](mailto:rvallin@ubm.com)  
t: 415.947.6223

## ADVERTISING PRODUCTION

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

## REPRINTS

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

## AUDIENCE DEVELOPMENT

### AUDIENCE DEVELOPMENT MANAGER

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

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



UBM

WWW.UBM.COM

# FIREFALL



[firefallthegame.com](http://firefallthegame.com)



## the austin view

a panel of experts discuss the pending console transition

\\ The University of Texas at Austin Department of Computer Sciences hosted an event called Infinite Resolution Zero Latency on January 25, wherein local developers and other industry professionals took turns prognosticating the changes developers will see going into the next generation. Most present seemed to agree the mobile space will greatly influence how games will be made, played, and distributed moving forward, though they didn't agree on how we'll get there.

"I actually really believe the future is going to be handheld, but handhelds won't really be handhelds anymore," predicted Greg Zeschuk, VP of EA and general manager at BioWare Austin. He detailed a scenario where a consumer might place a hypothetical "iPhone 17" on a table, which would start a television. A player would then use a virtual controller that doesn't need to be physically touched. Zeschuk said a point would come where handhelds could significantly lessen the gap of what can be done with a high-end PC, citing a finite level of fidelity to which games can actually be built.

Red Fly Studios' director of product development Mike McShaffry is already working in the mobile space, and shared his thought that gaming will go the way of mobile without sounding the death knell of consoles as we know them. A recent report stated that the next Microsoft console would perhaps be six times more powerful than the current Xbox 360, and McShaffry predicted it could play part in failure of the system.

"I don't want to try to make a game that going to cost me \$1.7 billion to make," he explained, noting the ease and low cost with which developers can make a game for iOS and Android devices. He thinks that more quality games will make their way to the mobile space,



Twisted Pixel's THE GUNSTRINGER.

amid concerns for staggering development costs for future Microsoft, Sony, and Nintendo hardware.

Angst over these potential production costs hovered over smArtist principal Jon Jones's vision of how many will have to reconfigure their studios to survive. "As long as expensive plug in play blockbusters are made, dumb money will follow," Jones said. "Mistakes will be made, studios will crumble, and layoffs will abound."

Under these circumstances, Jones predicted what he called "mercenary studios" rising to prominence. These specialized studios will stay afloat doing project-to-project contract work. These boutique houses would be a safer way to hedge a developer's bets against closures and other calamities in the industry beyond their control, he claimed. The fall of such studios in the U.K. would suggest otherwise, but his opinion stands.

On the heels of Kinect title THE GUNSTRINGER, Twisted Pixel lead designer Dan Teasdale painted a markedly more positive future for next-generation game development, predicting a Golden Age. He cited his experience at Pandemic Studios as an example of the negative view of console transitions. He said the studio failed to produce the amount of content desired under the schedule necessary.

"Every console transition you learn a new set of skills," Teasdale said. "You basically relearn how to makes games again. You're learning how to make this content that fits exactly the console that you're authoring."

This time around, Teasdale claimed developers should be able to break the cycle by learning lessons from the past several years of development. "This generation we did something I don't think many people realized that we did," he continued, making a counterpoint to McShaffry's bloated budget

concerns. "We started authoring way above what the console can handle. We have stuff like Zbrush where we're authoring millions and millions of polys and breaking them down."

The significance of this for the next generation of consoles, according to Teasdale, is that developers already have acquired the skills to create those games. In a rather bold statement, he claimed the industry has had its *Citizen Kane* moment.

"I think we passed that point a few years ago without realizing it," he explained. "We have now defined our tools, our skill-sets. A design vocab was something people at GDC panels would try and make ten years ago. We have that now."

Others also provided examples of innovations and tools that are still being developed and learned. University of Texas computer science professor Donald Fussell described research students have put into efficiency-related software tools, such as a system checking process called model checking.

"What we want to do is have game [developers] be able to check the consistency of the virtual worlds that they are building while they build them," Fussell said. The idea is with a push of a button to allow developers to answer the questions concerning various states and the conditions for those states in a kind of concurrent bug checking.

No matter what view of the future speakers presented, there remained a constant strain of hope and opportunity for developers through the night. Whether the industry moves along with Teasdale's proposed "Golden Age of gaming," or continues further in the mobile and smartphone direction, developers were relatively bullish on the future of the industry.

—GERREN FISHER

# 1H1D!!!

the origins of kyoto's best chiptune party

\\ A cozy 8-bit themed bar, whose back corner substitutes for an elevated stage, has over the course of five years become among the world's most widely respected venues for live chiptune music.

Located in Kyoto, a stone's throw away from the historic geisha district of Gion, Café la Siesta has been host to Bit Shifter of New York, Swiss Atari musician STU, and Akira 8GB of Argentina.

Part of the allure of the location's 1H1D!!! live shows is the mellowness of the vibe. The clink of beer glasses and casual conversation mingle with the reverberating square waves. The party gets its name [short for "1 hour, 1 day!"] from game creator Takahashi Meijin's reminder to kids to limit their Famicom intake to one hour per day.

"These machines have had a profound impact on my life," says shopkeeper Master Kohta of the old-school Nintendo consoles. "I get a lot out of using them to create music."

The bar's frequent nighttime parties are set in motion by a DJ set by Kohta, as listeners in Japan and overseas join the Ustream feed. Broadcasting his shows has paved the way for surprising collaborations. For example, the freeware developer Locomalito, maker of the side-scrolling shooter HYDORAH alongside composer Gryzor87, recently reached out to 1H1D!!! over the web from their studio in Andalucía, Spain.

This correspondence led to the PC game's Spanish-language vocal track "Path of Scylla" being arranged by Kohta and Siesta regular BIG BROTHER. The remix,

featuring Japanese lyrics sung by vocalist Asaka Tomoko, was performed at the most recent 1H1D!!! live show.

As of this writing, the most recent party witnessed Swedish artist Nordloef, donned in a ski mask, dreadlocks whipping madly through the air, wielding dual Game Boys. The performer is perhaps best known for reinterpreting the aesthetic of the classic console through the lens of punk rock.

It's easy to see how a gaming device's signature sound can capture an artist's imagination when that sound is pushed in

March's 1H1D!!! headliner, Soichi Terada, is second only to foam-suited chip-pop group YMCK in elaborate style—he wears a white mask reminiscent of Noh theater and a Shinto shaman's red hakama during his multimedia "Omodaka" performance.

Omodaka addresses the audience through a guttural, electronically distorted vocoder. After introducing his band members, which include a DS Lite and PSP, he performs these instruments next to a flat-screen monitor that projects the image of a rosy-cheeked enka singer. Game

label VORC, and veteran game composer Kimitaka Matsumae. Even the sign outside is designed by PIXELJUNKEDEN artist Baiyon. These marks of legitimacy are easy to appreciate among chip music organizers in the West who know what it takes to draw a loyal crowd.

Five annual 1H1D!!! compilations have been published to disc, due to support from local artists. You'll usually see KAZ a.k.a.HIGE, Ben, Longfi, USK of Portalen, NTDSK, or the ever-unpronounceable NNNNNNNNNN at every show. The participation of these Kansai



unintended directions, as with Nordloef's cover of Weezer's "Buddy Holly". Many chiptune musicians prefer not to have their work confused with game scores, but at Café la Siesta the historical contexts of 8-bit are impossible to ignore.

hardware is put to use in conjuring up a contemporary take on ancient Japanese tradition. [Terada has previously written music for the APE ESCAPE game series.]

Other performances at Café la Siesta have featured Hally, the founder of influential chiptune

-area musicians, among others, has been the key to Café la Siesta's success.

So if ever you find yourself in Kyoto, do yourself a favor and head to Café la Siesta, and let the chiptunes begin.

—JERIASKA



Introducing

# fmod® studio

## The game changer for audio professionals

- New DAW inspired multi-track music and event editing interface
- Fully featured mixing desk with pro effects for mastering
- Create, add, edit and mix audio content **live** in game
- Integrated profiler captures game events and audio output
- Hardware control surface support, mixer snapshots, VCAs, sends, nested events, source control/multi user collaboration and more!

See us at GDC 2012, Stall 1532

Effects supplied by



To get involved contact us at [sales@fmod.org](mailto:sales@fmod.org).

Firelight Technologies Pty Ltd.

[www.fmod.org](http://www.fmod.org)

# CHEATERS BEHIND THE CLOUD

DESIGNING CHEAT-  
RESISTANT PEER-TO-PEER  
NETWORK PROTOCOLS

F E R D I N A N D S C H O B E R

You're at home, and you've just logged into the latest, greatest multiplayer FPS to come out of BigAddictiveTimeSuck Studios. You've spent countless hours in multiplayer matches and maxed out your character. You dumped more cash than you did on your previous vacation to Hawaii to upgrade your character until he had more killing power than Arnold Schwarzenegger as the Terminator. Your win streak was flawless—one more and you'll be at the top of the leaderboard. You log into a server and are ready to kick some butt. Then you get whomped by the first level-1 character with a wooden club.

Welcome to cheating hell: You picked the wrong server. There is no salvation. Your statistics are going to suffer. Kiss that flawless win streak goodbye.

Multiplayer and MMO game functionality is now at a stage where game developers are no longer investigating how to do it, but rather how to do it more securely and efficiently. There are more online players, and eSports, and in-game economies than ever. And those that are backed by real money require more robust security than is typically implemented for multiplayer games.

The current prevalent security approach is to have game servers under full control of developers and publishers, either on a closed console platform or in a datacenter. This limits the attack vector of cheaters significantly, and it allows for much more secure play than leaving the game server in the hands of the players. At the same time, it causes a scalability bottleneck and an end-of-life support issue for games.

To address scalability, an increasing number of developers and publishers are turning to cloud environments to host servers for multiplayer games. Pushing servers into the cloud has multiple benefits. First and foremost, it virtually eliminates capacity planning;

with a new game, it's often very difficult to estimate a concurrent player count, which frequently results in unused server capacity. Secondly, unlike in single datacenters, appropriating new servers in the cloud is easy, and it is no challenge to absorb even the most significant player spikes over the lifetime of a game (see Figure 1). Finally, it also frees up constrained development resources that could be used more efficiently elsewhere. Overall, hosting servers in the cloud can significantly reduce running costs of multiplayer and MMO games, especially if strong player fluctuations are expected during the game's lifetime.

Because of these benefits, moving to the cloud seems like a great proposal for most development studios. But what if you are a small studio that has to reduce costs? What if any sizeable datacenter is a no-go, and even using a cloud service is outside your reach? What if you want to provide your players with the ability to host their games securely for end-of-life scenarios, when you have to stop ongoing support of a game? The answer you may often hear is that "without servers under authoritative control, it's impossible to guarantee cheat-resistant gameplay."

Here I will show you a different approach: If you are willing to spend engineering effort, it's possible to build a cheat-resistant peer-to-peer

# DESIGNING CHEAT-RESISTANT PEER-TO-PEER

multiplayer system that requires no datacenter resources. It is effectively your own private, cloud-like system.

## DESIGN BASICS

/// When designing a new system like this, you should start by taking a closer look at which security and anti-cheating guarantees you need. Independent of game type or mode, the following guarantees are essential for a cheat-resistant design:

### 1. Strong player identities.

Players should not be able to participate in multiplayer gameplay without presenting strong proof of identity. Such identities should be a scarce resource. Strong identities can be used to uniquely identify cheaters and punish them if needed.

### 2. Message integrity and authenticity.

Any type of message between players needs to be protected from modification and replay attacks. Furthermore, message authenticity should be verifiable. Without these guarantees, cheaters can assume other identities and manipulate messages. Depending on the type of game, message confidentiality may also be essential to cheat-resistant gameplay.

### 3. Gameplay verifiability.

Gameplay actions need to be verifiable. Given a set of starting conditions and actions, a game client should be able to determine if all actions were valid and which final outcome is expected. Verifiable gameplay allows the identification of current and past cheating attempts.

If a system provides these guarantees, it can provide cheat-resistant gameplay. Guarantees 1 and 2 can be implemented in a peer-to-peer system, and they map directly to cheat resistance. Guarantee 3 is not required in a client-server model because all game logic can be pushed into the trusted server, but in a peer-to-peer system, we need to be able to verify the actions of other peers, since they are untrusted. How to achieve this in a peer-to-peer system is not immediately obvious: By definition, a peer-to-peer network has no single arbiter. It may therefore seem futile to even attempt to create a cheat-resistant peer-to-peer system.

However, in virtually all games, the vast majority of players will fall into one of three categories: players who choose the game for its challenge and would not consider cheating; players who are participating in a friendly game with their friends; and players who exploit glitches if they are easy and risk free—like item-duplication glitches—but don't actively participate in cheating. In other words, when you look at a larger sample set of players, the honest players will be in the majority. We can use this to our advantage by using the power of the combined player base.

But how do we do this? Democracy! We'll build a voting system, which is something you may already have seen on a small scale in multiplayer sessions. Let's assume that any given player is a cheating player with a probability of  $1/c$ , which (as noted before) is rather small for most games. When we look at a group of  $n$  players, the chance of picking a group that consists of only cheaters is  $(1/c)^n$ . For large  $n$  and  $c$ , the probability of having a group of cheaters decreases quickly. A vote in a large group of players should therefore be dominated by honest players, as cheaters should be the minority. To be more precise, the

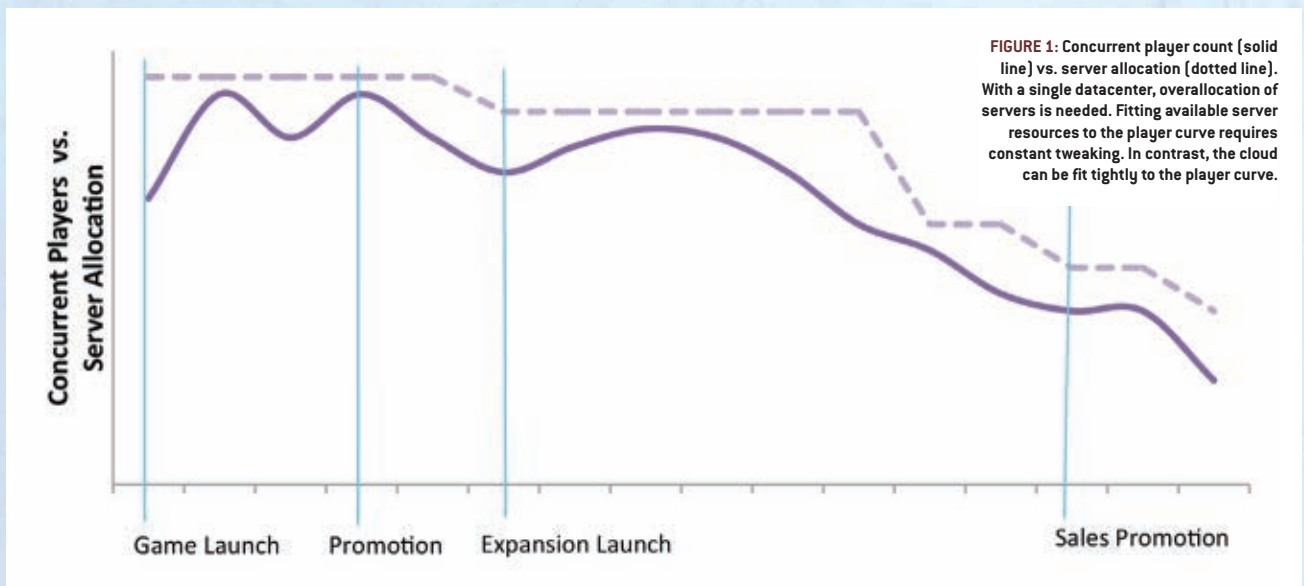
probability  $p$  that  $1/x$  of the players in a group of  $n$  are cheaters is  $p = (1/c)^{n/x}$ . This probability has to be low to ensure an honest outcome for most votes. Once we have identified  $1/c$  for a game, we can tune the number of players ( $n$ ) such that we have a satisfactory probability  $p$  for a value of  $x$ . This allows us to determine a minimum group size for gameplay verification. The value for  $x$  should always be greater than two to keep cheaters in the minority.

In practice, the exact group size may be hard to calculate because  $c$  has to be estimated. However, for any sizeable group of players (typically more than five), it is safe to assume that there are more honest players than cheaters. As a result, for any vote on game actions, a majority of the votes should come to the same honest (i.e., non-cheating) outcome. In this group, we will always be able to identify cheaters with one or more votes, as non-majority votes are highly likely to be cheating attempts.

A voting system like this is very powerful, and it will hold true as long as less than half of the player base is cheating. In practice, voting could even identify cheaters in environments where they are the majority, as long as not all cheaters are using a very similar cheating method. Simpler games are more likely to have identical cheating methods. If cheaters use different methods, they will disagree on the outcome of a vote and the largest matching vote would still be the vote of honest players.

## PEER-TO-PEER DESIGN

/// With the basics for gameplay verification in place, the first design challenge for our peer-to-peer system is authentication. As mentioned before, cheat-resistant systems need strong player identity. Without strong player identity we cannot leverage the power of the combined



player base—cheaters may just create zombie accounts to swing votes in their favor.

Depending on your platform, you may already have the tools to solve the authentication and identity problem. Current game consoles provide you with a basic player identity (or *profile*) and an accompanying authentication system. You can use this identity and any exposed authentication tokens for your own purposes. Similar functionality exists for most mobile platforms as well as desktop systems. If you're developing on a platform that does not provide this functionality or you choose not to use it, you'll need to create your own authentication system. The core issue to consider here is how to provide secure user authentication under *all* circumstances; multiple, well-known authentication protocols exist for such purposes (and it's unwise to design your own). It's important that user identities are also a constrained resource. For an example system, we will assume that user accounts are tied to game keys and that a secure login server is needed to start any multiplayer session. The login server will also provide a verifiable identity token for later use. We'll call this first peering step [Authenticate].

Once we have a set of authenticated players, we now need to construct a peer-to-peer mesh. The main security challenges here are integrity and efficiency. We want only authenticated players to connect, and any new players should find other players as quickly as possible. To join a peer-to-peer network, a new player needs to know at least one other active peer node. The distribution of such initial peers should be the final step during player authentication. Either the authentication system keeps a list of active peers and selects a suitable subset for the new player, or a handoff to a dedicated peer-to-peer tracker or dictionary occurs. The tracker is also responsible for verifying that peers are still connected to the network and providing updated peer selections when more connection points are needed. For security purposes, the peer selection shouldn't be predictable; otherwise, cheaters may be able to create cheating clusters in the network. In our example system, we assume we have a small set of trackers available. These will provide known, valid peers to any new peer with a valid authentication token. We'll call this step [GetInitialPeers].

A centralized tracker will still require some datacenter resources. However, unlike the client/server design, it's very lightweight. A tracker only handles client lookups and infrequent pinging, and it scales to very large numbers of concurrent players without any significant investment in hardware. Even better: at the end of life of a game it can be handed off to the fanbase, keeping the most vocal and hardcore fans happy well after the game's shelf life has expired. An alternative to a centralized tracker is a peer-to-peer tracking system that

keeps localized routing data on every peer in the network.

Next, we need to provide message authenticity and integrity guarantees. For this, all communication between peers has to be pairwise encrypted. Each new peer connection should start with a security handshake that establishes an encryption protocol and secret encryption key to ensure that all further communication is encrypted. Encryption provides additional confidentiality and digital signatures, and sequence numbers can provide further integrity and protection from replay and message injection attacks. For either option, each peer will have to keep one or two keys per active connection. To prevent unauthorized peers from joining, the handshake also needs to verify the authentication token. In our example system, we can satisfy these requirements by first using TLS to select a unique encryption key. We then use this key in symmetric AES encryption with CBC as a block cipher mode for pairwise communication. As a final step, we verify the identity of a peer through the authentication token that was provided by the login server. If it is invalid, the connection is terminated. Let's call this step [SecurePeerConnect].

This step has to be repeated for each peer node that joins the gameplay session. In addition, peers that drop from the network for any reason need to be replaced. (In our example system, a peer has to contact the tracker to retrieve replacement peers).

### GAMEPLAY VERIFICATION DESIGN

/// Now that we have a robust peer-to-peer system design, let's look at how to handle gameplay and voting.

The design of the gameplay propagation system will depend on the game type. In traditional multiplayer games, like FPS or strategy games, gameplay messages only propagate between a limited set of players, and the game state is transient. Gameplay data, except updates to the player's profile, does not persist longer than a short single gameplay session. In contrast, MMO games have to propagate game updates amongst a very large set of players and need a persistent game state.

### VERIFYING TRADITIONAL MULTIPLAYER GAMES

/// The two biggest problems here are finding peers for gameplay verification, and synchronizing gameplay effectively. While it's possible to perform synchronous updates and verification without any form of peer-to-peer structure, it's much simpler to use a two-tiered peer-architecture design. Here, we divide the peers into trusted and regular peers. A small set of trusted peers from the overall peer population

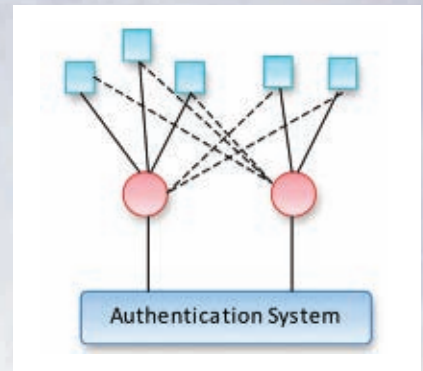


FIGURE 2: Simplified dataflow between an authentication system, and supernodes (red) and peers (blue). Gameplay and authentication flow is shown as a solid line, verification flow as a dashed line.

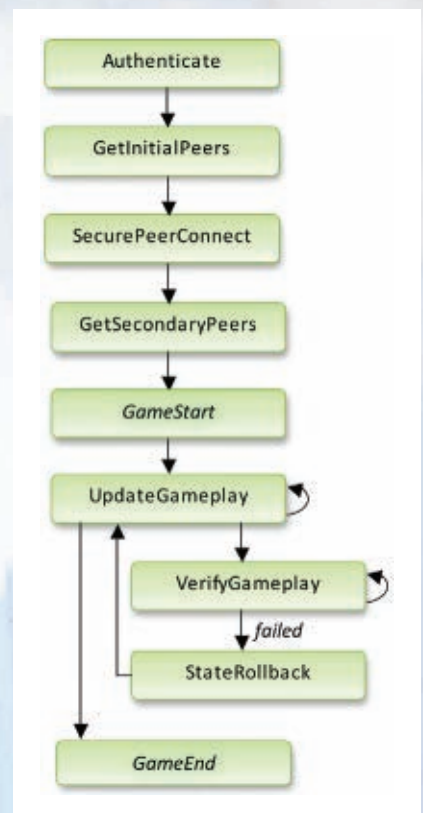


FIGURE 3: Event flow for a standard peer.

will serve as the arbiters and manage the other peers. Trusted peers—or supernodes—have proven over time that they are trustworthy and very unlikely to be cheaters, so their resources and trustworthiness can be used in the overall verification system. In our example system, we keep this trust information on the trackers.

Let's start by first picking and assigning to at least one supernode to each existing game mode (say, deathmatch vs. capture the flag). For redundancy, multiple supernodes are preferable,

# DESIGNING CHEAT-RESISTANT PEER-TO-PEER

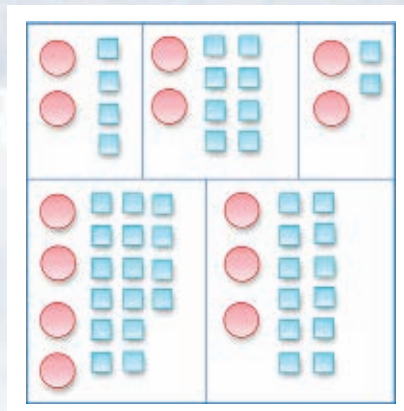
and the number of supernodes per game mode should also be scaled according to popularity. When new players join the peer-to-peer network, we now modify [GetInitialPeers] so they are first routed to the supernodes for their game type instead of to regular peers. Supernodes then divide new peers into gameplay groups and provide the peer set to the joining peer.

A gameplay group is the traditional set of players in a gameplay session. Additionally, new peers are joined to the supernode as a voting peer and may be called upon at a later point to vote on game actions. We call both join actions [GetSecondaryPeers]. All voting actions are handled through a dedicated connection, which is different from the gameplay connection. Each verifying peer is connected to at least one supernode, and it should never verify its own gameplay sessions.

Gameplay actions are now executed and verified as follows: When a peer performs a game action, its peers in the same session are updated directly in the same way traditional game clients would be. We call this action [UpdateGamePlay]. In addition, the peer sends its action and relevant game state to the supernode. The supernode forwards verification information to a set of verification peers and requests a vote. This set of peers can be across different game modes, but should never be in the game session they verify (see Figure 2). Each verifying peer executes the action and returns with one vote. The supernode can then confirm or reject the gameplay action to the gameplay peer. We call this action [VerifyGamePlay].

It may not be possible to verify all gameplay actions in all session types due to constrained network resources. A set of peers with very limited network bandwidth will not be able to duplicate all game actions without impact to general game flow. Fortunately, we can often rely on the game flow for such scenarios. In nearly all games, cheating attempts are not singular events, but have long-lasting impacts on the game state. Therefore, even verifying only high-risk game actions or a set fraction of game actions can effectively reveal cheating attempts.

Verification cannot be performed as quickly as a local gameplay action. In addition to verification time, it requires at least one round-trip each between the gameplay peer/supernode and supernode/voting peer. Peers can work around this delay by opportunistically executing gameplay actions before a vote is complete. All gameplay actions are temporarily accepted, but they are not committed until a vote is confirmed. For this, the peer keeps an action history to roll back any action that is rejected at a later point. It is not desirable to have rollbacks during a game session, but they are expected to be very rare. Specifically, rollbacks should occur only during a cheating attempt. At the point of a cheating attempt, players will prefer a rollback rather than



**FIGURE 4: Segmentation of MMO game world into chunks. Each chunk contains multiple supernodes (red) based on client population (blue). Chunks do not need to be of equivalent size in the game world.**

continuing with a cheated game state, even if it impacts continuity. We call this rollback action [StateRollback].

Once a supernode rejects a gameplay update, it can also isolate the offending peers and react appropriately. The reaction can vary from removing the cheating peers from the gameplay session and marking the cheating attempt on their profile, to an immediate ban from the network.

It is important that supernodes are verified on a regular basis to detect any malicious behavior. When multiple supernodes are assigned to one game mode, they can arbitrarily verify each other's behavior and the voting results that are sent to the peers. If a cheating attempt is found, the offending supernode can be reported to the authentication service. Here, it can be removed from the list of trusted peers.

## MASSIVELY MULTIPLAYER GAMES

/// These can use the same voting design (see Figure 3), but gameplay updates need to be extended to take persistent game state storage into consideration. As a result, different segmentation of supernodes is needed. Instead of segmenting supernodes by game modes, the MMO game world is segmented into chunks, which represent game areas. The number of chunks will be static in most games. One supernode can handle multiple chunks, but each chunk should have at least two supernodes assigned to it (see Figure 4). When players move into a chunk in the gameworld, they transfer to one of the associated supernodes. A chunk to supernode mapping can either be provided by the tracker, or negotiated amongst supernodes.

The supernodes of a chunk and all connected peers keep its game state. Like in the previous design, game actions are executed and voted on. Similar to before, peers should not vote on actions in their own chunk to reduce the incentive for cheating. Players are less likely to cheat if it does

not affect them directly.

In a system where all data is stored on peers, a high peer churn can cause issues. Therefore, you should always ensure that game information is updated in a shared database in addition to the game state that is held on supernodes. Updates can be performed asynchronously from each supernode and merged into a time-stamped game state in the shared database. This shared database could itself be a peer-to-peer system, single system, or a cloud-based solution.

Multiple copies of the same game state and a large population can significantly reduce the need for frequent backups. By requiring at least two supernodes per chunk, we can ensure basic redundancy. Further redundancy is provided by connected peers, as they also keep parts of the game state. When peers leave, the supernode can provide the needed game state to new peers. When a supernode has to be replaced, the secondary supernode can be used for bootstrapping. If a conflict in states of multiple supernodes is detected, it is treated similarly to a mismatch in a game update vote.

Not relying on a shared database will work as long as sufficient supernodes and peers are present. As a result, during the early, high-population period of a game (as in Figure 1) the shared database will only provide an auxiliary service. As time progresses and player numbers drop, the shared database will increase in importance. It becomes essential when chunks only have one supernode. It then serves to bootstrap any chunks that have lost their last supernode or are devoid of peers and supernodes.

At this point, you should have a good understanding of how to design a cheat-resistant peer-to-peer system that requires minimal dedicated server resources. The outlined design should serve as a guide in implementing your own solution; however, some corner cases were left out due to space constraints and will require additional design effort. Still, you should already see that it is possible to create a cheat-resistant peer-to-peer network that is highly scalable and requires little to no server capacity. Once designed and implemented, you can use and benefit from such a system across all your games. Why bother with capacity planning when your players can take the load for you? 🎮

**FERDINAND SCHÖBER** is a software development engineer in Microsoft's Advanced Technology Group. He is an old-school RPG player and has been in the game industry for over five years, focusing on security and networking across different platforms. Ferdinand worked on a breath of projects ranging from the Windows 7 multiplayer platform to security testing on the *GEARS OF WAR* and *HALO* franchises. He also spent time working on academic research in cloud-based cellular malware detection and entertainment security analysis. You can find him on Xbox LIVE and most social media sites.

# with 180 million Arabs under the age of 25\*...



## ...the gaming industry is set to boom in the Arab world.

**twofour54° Abu Dhabi** – the tax-free gateway to a new world of gamers

The MENA region is one of the world's fastest growing media and entertainment markets with 19% growth in recent years. And with 80% of under-25s owning mobile phones\*, strong broadband take-up and new gaming innovations, it's a prime opportunity for gaming businesses. Over 100 leading media companies are already capitalising on the opportunity at **twofour54° Abu Dhabi**.

- 100% company ownership in a stable, tax-free environment
- Unique campus environment with facilitated business networking
- The region's only stereoscopic 3D Lab
- **twofour54°** gaming academy in partnership with Ubisoft®
- Easy licensing and business set-up services
- Guidance and liaison with UAE content regulatory bodies
- Dedicated fund for mobile apps development via Apps Arabia™
- Full on-site HD production and post-production facilities

Find out how we could help grow your business today.

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



**twofour54**  
Abu Dhabi  
media & entertainment hub

\*Sources: Arab Media Outlook 2010. Media on the Move 2009. A.T. Kearney. Introduction to Gaming. Michael Moore. Screen Digest. IDC.

IN 2012 RED 5 STUDIOS WILL LAUNCH WANT TO BE A PART OF

**FIREFALL** **SOMETHING BIG?**

DESTINED TO BE THE NEXT BIG HIT IN ONLINE GAMING AAA QUALITY, FACE MELTING  
OPEN WORLD SHOOTER

**RED 5 STUDIOS** IT WILL BE **COMPLETELY FREE2PLAY**

IS AT THE APEX OF  
REVOLUTIONIZING THE GAME INDUSTRY

USING A GAME MODEL THAT IS POISED TO  
SHIFT PARADIGM THE ENTERTAINMENT

AND THAT'S WE'RE REINNOVATING  
ONLY THE BEST A RADICAL

BEGINNING AT RED 5'S NEW ONLINE PLATFORM

WE BELIEVE THAT OUR INSPIRED TEAM OF GAMERS

WE ARE COMMITTED TO FINDING THE FUN IN EVERYTHING WE CREATE

RED 5 STUDIOS IS CURRENTLY RECRUITING

THE NEXT GENERATION OF INDUSTRY REVOLUTIONARIES

WE WANT TALENTED GAME DEVS

WITH A BURNING DESIRE TO SET THE NEW STANDARD

FOR ONLINE GAMES AND ENTERTAINMENT

GET IN ON THE GROUND THAT WE HAVE OUR OWN MAKE THE FUTURE

OUR COMPANY CULTURE IS UNPARALLELED IN AWESOMENESS

LOCATED IN WONDERFUL SOUTHERN CALIFORNIA

MINUTES FROM THE MOST BEAUTIFUL FLOOR OF TIKI BAR?

BEACHES IN THE GOLDEN STATE SOMETHING SPECIAL

**JOIN THE TRIBE!**



[red5studios.com/jobs](http://red5studios.com/jobs)

# smashed!

## Vehicle Deformation in SAINTS ROW: THE THIRD

One of Volition's goals for SAINTS ROW: THE THIRD was to exaggerate exciting and dramatic moments to create a wholly over-the-top experience. For vehicles, this meant one-button drifts, new aircraft, and above all, improved deformation. Players needed to see the fronts of their sports cars bashed several feet in after a 100mph collision, and be able to glance back from their tank at the trail of mangled cars in its wake. And while some of the techniques described here may only be necessary for games that feature a mixture of on-foot and vehicle gameplay, we hope that many of the challenges, shortcuts, and pitfalls will be relevant to those tackling similar deformation problems.

### VEHICLE COLLISION IN EARLIER SAINTS ROW TITLES

/// In order to move toward our deformation goals, it was first necessary to simplify our method of representing vehicle-collision geometry. In SAINTS ROW 1, vehicles used the somewhat naive approach of having collision geometry that very closely matched the visual geometry, including detailed interior and trunk geometry. An example of this

collision geometry can be seen in **Figure 1**. This high-quality model proved computationally expensive during collisions. Additionally, systems like our water simulation functioned better when dealing with simple, broad shapes.

In an effort to address both of these issues, SAINTS ROW 2 added the concept of "simple collision." As the name implies, simple collision was a vastly simplified version of a vehicle's collision geometry that was used in addition to the old collision model, then referred to as "complex collision." **Figure 2** shows an example of simplified collision. As you can see, this simplified collision is only a broad, mostly convex shape without any interior collision. The vehicle's simple and complex collision models overlapped each other, colliding with mutually exclusive subsets of object types. Larger-scale physics interactions, such as vehicle-vehicle or vehicle-world collisions, were handled with the simple collision model, whereas smaller scale collisions, such as vehicle-bullet or vehicle-ragdoll, were handled with the complex collision layer. While this double-layered system provided performance and interaction benefits, it would be unfeasible for our planned vehicle deformation. CONTINUED ON PAGE 15

Volition's SAINTS ROW: THE THIRD



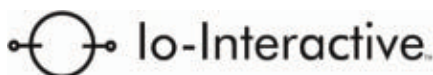


**GDC**  
BOOTH #NH1638

**umbra**  
SOFTWARE



Powering projects from



and many others...

## UMBRA 3 RENDERING OPTIMIZATION

Render more complex and visually striking worlds by cutting down CPU and GPU processing times by optimizing critical parts of a game such as rendering, audio and game logic. Umbra middleware automatically generates portals for game scenes in pre-process taking away tedious manual work from artists. The generated data is used at runtime to perform fast and efficient visibility queries.

By effectively determining what's visible and what's not, Umbra 3 helps you to render visually stunning worlds running at smooth framerates. Integrating Umbra 3 into any type of game engines is easy.

Umbra 3 middleware is available for all major console and mobile platforms.

*Umbra 3 is available for free evaluations from [umbra3.com](http://umbra3.com). Full technical support is given during your evaluation.*

**UMBRA3.COM**

CONTINUED FROM PAGE 13

## UNIFYING THE COLLISION MODEL

/// It was important for us to change a vehicle's physics geometry so that it accurately reflected any damage it received. This deformation would need to be applied both to the simple and complex collision layers to keep the two aligned, lest we cause issues with phantom object-specific collision lingering after deformation events. Since the methods we had been discussing and prototyping would likely require subdivision of a vehicle's collision model, using our old two-layered system would have required subdividing two different layers of vehicle collision and linking their corresponding pieces to make sure any deformation was synchronized. Neither vehicle artists nor programmers wanted this level of complexity in the collision geometry, so we began to investigate the feasibility of unifying our collision model.

We were already discussing the removal of complex collision at this point; it was time consuming for artists to create, required additional memory, and caused significant CPU performance hits in various types of collisions. The added headaches complex collision would have caused our proposed deformation system cemented our desire to get rid of it, but we needed to make sure we could do so without significant repercussions.

The two primary objects that collided with vehicle complex collision were ragdolls and bullets. Ragdolls were actually an easier case, because the main reason they needed to collide with complex collision was so they would behave properly when inside a car (interiors were modeled only in the complex collision).

In SAINTS ROW 2, while it was occasionally neat to see a corpse bounce around in the rear seat of a convertible you had just commandeered with lethal force, it was common for the ragdoll to jitter through unnatural positions as the physics system did its best to handle all the constrained bones being thrown around in a tight enclosure. In fact, animation had already taken umbrage with this behavior and was leaning toward an animation-driven solution for corpses in vehicles. This fit in perfectly with our plans: If shooting a passenger in a vehicle would no longer result in them ragdolling, there was no need for interior collision geometry.

## SHOOTING A BULLET THROUGH A SOLID CAR

/// Bullets proved to be a more complicated problem. Like ragdolls, they primarily collided with a vehicle's complex layer; in this case, a vehicle's interior collision geometry or, more specifically, the absence of interior collision geometry. The sheer number of vehicles and frequency of gunplay in SAINTS ROW makes it critical that weapons be able to fire through windows and open doors to hit enemies. However, the solid "simple collision" shapes didn't have any holes or open areas for the bullets to go through. Fortunately, bullets in SAINTS ROW are simulated with raycasts, and there was already a mechanism in place for rejecting early hits in case of breakable materials or penetrating bullets.

Exploiting this system, the "shooting through a solid car" problem was solved by calculating the number of windows or open doors the bullet ray would intersect upon collision with a vehicle, and then conditionally allowing the bullet ray to penetrate the vehicle. Since these doors or windows had no representation in our simple vehicle collision, this intersection test was done separately from our physics world. Depending on the number of windows or doors the raycast hit, we would do the following:

**0 hits:** The bullet hits the car where dictated by the car's simple collision.

**1 hit:** This case was the most complicated because sometimes the bullet would hit a human in the vehicle, and there were also a variety of issues related to firing through the window of a partially open door. Ultimately, we decided to both damage the vehicle and cancel the collision. However, we would cancel any ground-hit VFX to prevent dust explosions from clipping through the floor of a car when shooting at it through a window. If you carefully examine this case in the shipped version of SAINTS ROW: THE THIRD, you may notice minor visual oddities, but they're subtle enough that the vast majority of players won't pick up on them.

**2+ hits:** the bullet goes through the vehicle, shattering any windows it intersects.



# smashed!

FIGURE 1

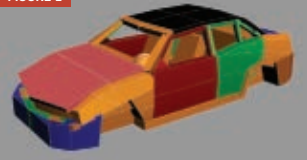


FIGURE 2

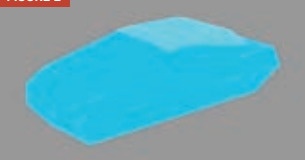


FIGURE 3

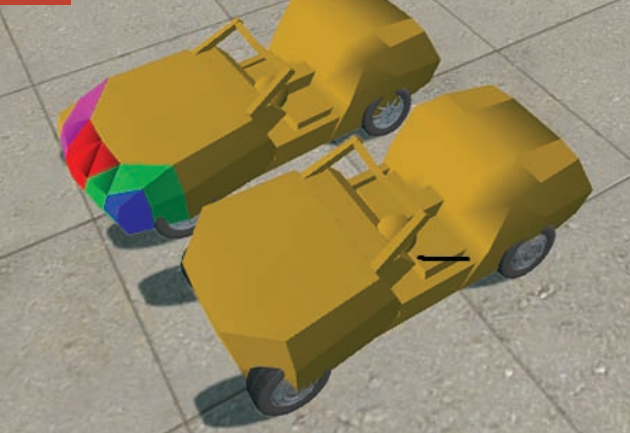


FIGURE 4

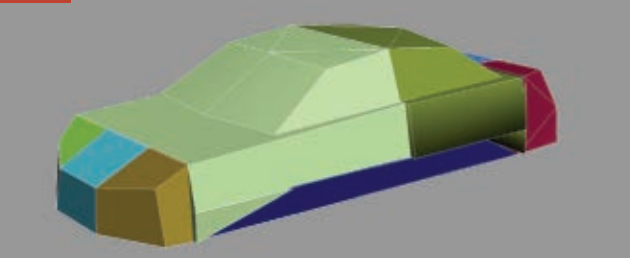


FIGURE 5



This approach allowed us to simulate bullets going through our otherwise solid simple collision, and it removed the final blockade toward a system where all vehicle collision was represented by a single simple model.

## DEFORMABLE COLLISION PIECES

/// With this new model in place, we were ready to start prototyping different methods of deformation, and though we had a variety of ideas for potential approaches, time and complexity concerns drove us to prototype the simplest solution first, which ended up working so well we eventually shipped with it.

Our solution for the physics side of deformation was to subdivide the vehicle's collision to include a variety of deformable collision pieces (DCPs). These pieces corresponded to areas of the vehicle we wanted to be able to bash in, and their collision could be turned off when a sufficiently damaging event occurred. Since we could already build vehicles with dislodgeable components, prototyping this approach required no new pipeline support (though we did modify the pipeline once we settled on this approach).

Figure 3 shows one of our early prototypes. The front of the car is colorfully subdivided into different pieces, all of which could be knocked off to

change a vehicle's physics. Iterating with various patterns, we found that the removal of smaller pieces had almost unnoticeable effects on the character's interaction with the vehicle, so we moved toward coarser subdivisions with fewer pieces, eventually settling on the pattern seen in Figure 4. This pattern features only three pieces on the front and back sides of a vehicle, one for each corner and a central piece. While this reduced number of pieces made things simpler from a pipeline and simulation standpoint, we were still concerned about visual fidelity.

## LINKING VISUAL AND PHYSICS DEFORMATION

/// While we had no system for physics deformation in previous SAINTS ROW games, our artists had still been pushing the limits of what could be done with visual deformation without diverging so far from the physics geometry that the disparity was obvious. Figure 5 gives an example of the difference between a SAINTS ROW 2 vehicle and its fully deformed state.

To accomplish these deformations, the artists built a deformed copy of the visual geometry of any parts of the vehicle they wanted to deform (any part not duplicated into the deformed copy wouldn't deform in-game). When a vehicle was first deformed during play, we would allocate it a set of morph weights that matched the number of vertices. Each weight was only a byte long, but this was not a trivial amount of memory given our 10–20K verts on a car, so we tried to be careful about what cars deformed in an effort to keep memory usage under control. When a collision or other damage event occurred on the vehicle, we processed a "deformation sphere" on it, which deformed morph weights of vertices within a radius of the sphere's center with declining severity for points further from the center. The vertex shader associated with vehicles would then take these weights as a parameter and interpolate between the deformed and undeformed vertex positions for the two models the artists had built.

This deform target system for handling visual deformation had no hard limits in terms of the amount of visual deformation allowed, so we decided to keep it mostly intact, only extending the amount of deformation built into the models and adjusting our morph weight system to tie in the DCPs described above. For the first adaptation, the vertices on the front and back of deformable vehicles often differed by 1–2.5 feet from their undeformed position, whereas the vertices of a deformed model previously differed by only a few inches from their undeformed counterparts.

We didn't realize the second adaptation—linking visual deformation with the removal of DCPs—was necessary until we tried and failed to use the vehicle damage events to drive both visual and physics deformation. Our first mistake was making DCPs use a hitpoint system, where their hitpoints were gradually worn down through multiple collisions, while visual deformation caused by the deformation weights was not cumulative. Instead, only the most extreme deformation value of all relevant deformation events was used. However, even after revising DCPs to use a damage threshold (where a certain amount of damage had to be done in one event to remove the DCP), we ran into trouble; while visual vehicle deformation happens as a gradient process, the removal of a DCP is one discrete event.

It makes sense to see the front of your car gradually get more bashed in as you experience more and more extreme collisions, but if you set your DCP removal threshold at enough damage to do 50% visual deformation, the visual difference between a front end that's 52% deformed and one that's 48% deformed is minor, though their physics differences are as far apart as possible in this system. The above example is exacerbated if applied to two adjacent DCP sections because, though they are deformed similarly, one has physics geometry and one doesn't. Additionally, because deformation is applied in spheres, the visual geometry overlapping a DCP could be deformed substantially by a collision, even if that collision wasn't with the DCP in question.

Considering the above issues, plus the possibility of other undiscovered ways for the DCPs and visual collision to get out of sync, we decided to explicitly link the two in the following fashion: Each DCP has both a maximum deformation value any of the verts within it can reach before being removed, and a minimum value for every vertex associated with it after it's been

removed. These values are not the same, and the gap between them helps with the 48/52% problem above.

Conceptually, you can think of it as being impossible to deform a vert associated with a DCP anywhere between 35 and 60%. (The actual lower value varies based on positioning; I'll go into that in greater depth later.) Once a vertex is deformed more than 35%, all vertices in that DCP jump to >60% deformed (though we try to preserve some of the previous deformation proportions to prevent the vertices from looking too uniform) and the DCP is removed.

From a player's standpoint, this dead zone isn't noticeable, because any impact significant enough to cause >35% deformation is severe enough that causing 60% deformation isn't unreasonable. From a visual standpoint, there is still the problem that 34% deformed regions would have the exact same physical geometry as undeformed regions. Fortunately, players are trained to expect a certain amount of hovering as characters in video games can rarely press themselves directly against visual geometry (such as a wall or other barrier), so being unable to force the character to touch the vehicle's visual collision isn't out of the ordinary. Also, the difference between two inches of hovering and twelve inches of hovering, while noticeable if you're looking for it, was unnoticed by virtually anyone outside the vehicle team. It's worth noting that clipping looks much worse than hovering, so while there is minor clipping in fringe cases, like a character's hand going through a part of the hood when running into a 60% deformed vehicle at a diagonal, we mostly opted to go with increased hovering instead.

I previously mentioned the variable threshold for when visual deformation causes a DCP to be removed. This is necessary to avoid discontinuous jumps in the amount of visual deformation caused on the front of a vehicle. If the deformation limits described above were strictly enforced, then every vertex corresponding to a part of the car missing a DCP would have a minimum deformation of 60%, whereas every vertex in a neighboring part of the car with a DCP present would have a maximum deformation of 35%, meaning there would be a significant visual disparity on the boundary between these two regions. Our solution was to have each vertex compute its maximum deformation threshold by taking into account the distance from its two nearest DCPs. **Figure 6** gives an example of how the deformation threshold would vary across vertices with the center DCP missing. As you can see, the vertices are 100% deformed near the center of middle DCP, and they become less deformed as they near the side DCPs until they reach the threshold of 35%.

Our first pass at this visual deformation system saw CPU spikes of 20ms, so we had to heavily optimize all the per-vertex code. We rearranged our vertex buffer so that reading through it was cache friendly, used SIMD commands to process 4 vertices at the same time, and made a deformation job system to process different parts of the car on different threads. Ultimately, we were able to get the processing time down to something that barely registered on performance readings.

Having resolved these issues, we were able to create a deformation system that allowed for a wide range of visual deformation and link it to a physics system that was simple, yet provided sufficient support so as to not undermine the visuals. Now it was time to turn our attention to tanks.

#### PREVIOUS ATTEMPTS AT CAR-CRUSHING VEHICLES

/// We had wanted vehicles that could drive over other vehicles in earlier SAINTS ROW games (especially the monster trucks in SAINTS ROW 2), but our efforts were frustrated by our vehicle wheel model. The SAINTS ROW series uses the Havok physics engine and a heavily modified version of its vehicle kit for our vehicles. Havok simulates wheels by casting a wheel shape at the ground and placing the wheels wherever the cast hits, solving the eventual position of the wheel and the resulting suspension forces from there. There are lots of good reasons for this approach, ranging from improved CPU performance to improved contact with the ground. One important consequence, though, is that the wheels themselves don't interact with other objects in the simulation except to push up off of them.

This is rarely an issue with most vehicles, as the wheels are encased in wheel wells, so nothing would directly collide with a wheel anyway. However, when we began making vehicles with wheels large enough to potentially drive over cars, the lack of proper wheel collision became problematic. When working on monster trucks for SAINTS ROW 2, we realized that, while large wheels would allow us to drive over large obstacles like cars, they would also allow stationary monster trucks to be lifted up by smaller cars that drove at them. The root problem was that the wheels were purely reactive to the geometry near them because they were simulated by linear casts. Also, while wheels could exert forces on cars beneath them, the linear cast would have already moved the monster truck's wheels to be above the smaller car by that point. Additionally, while the wheels of the monster truck would exert some force on the car underneath, it wasn't enough to keep the smaller car from driving completely under the monster truck. Adding vehicle-collision geometry that overlapped with the wheels "solved" this problem, but at the cost of almost completely removing the monster truck's ability to drive over cars.

After this disappointing result, we decided to try simulating vehicle wheels with rigid bodies as opposed to linear casts. This approach was fraught with a variety of handling issues related to excess friction, and we eventually reverted to our non-ideal-linear-cast-with-overlapping-collision approach to ship SAINTS ROW 2.



#### "THE THIRD" TIME'S THE CHARM

/// SAINTS ROW: THE THIRD offered us another go at this vehicles-crushing-vehicles problem (although this time the story called for tanks instead of monster trucks), and we now had both more development time and a more refined deformable collision system.

As we approached the problem of tanks crushing cars, we drew inspiration from two sources. The first was Radical Entertainment's PROTOTYPE, which created the feel of a tank rolling through crowded city streets, even if the resulting visual and physics deformation weren't up to the standards we wanted; however, that shouldn't be taken as a slight to PROTOTYPE, as I think its trade-offs were quite reasonable given the role of vehicles in that game. The second source of inspiration was various YouTube videos of tanks running over vehicles. In both PROTOTYPE and videos we watched, we noticed that tanks seemed to go through vehicles much more than they went over them. This approach presented a way to circumvent the various issues we encountered when implementing monster trucks.

Our most important goal with this approach was to make a tank interact with a vehicle as if it were a speed bump. PROTOTYPE gave us the idea of

# smashed!

knocking all the wheels off a vehicle when it was being crushed by a tank, which helped, but cars lying on the road without wheels were still tall enough to cause problems for tanks. This is where our DCP system comes in, as it made sense that a tank would be able to crush them because DCPs represent areas of a car that can be deformed. However, we only had six pieces in the front and back of a vehicle up to this point, so we added additional center pieces (that could only be removed through crushing) as well.

Our eventual DCP breakup for a standard vehicle is highlighted in light blue in **Figure 7**. As you can see, the vast majority of a vehicle's collision consists of DCPs, leaving only a "chassis" piece at the bottom which looks not unlike an elongated speed bump. We then removed DCPs whenever they would collide with a tank's geometry, which resulted in our tank driving through them as if they weren't even there and interacting only with the lower "speed bump" section of the vehicle. This successfully gave the feel we were hoping for. However, as was the case with DCPs when we used them for collisions, their binary nature meant we only ever had coarse control of the physics geometry. And, as with removing the DCPs from collision, we paired this coarse geometry with a high-quality visual model to convince players of a car's deformation.

## VISUAL DEFORMATION FOR TANK CRUSHING

/// As was explained in the section about visual deformation for collisions, the SAINTS ROW vehicle-deformation system uses a set of morph weights to interpolate between deformed and undeformed versions of a collision model.



Because the deformation caused when getting crushed by a tank is very different from the deformation caused by getting T-boned, we knew we would need to extend this model somehow. However, we wanted to do this without requiring vehicle art to build another damaged version of the vehicle. We'll first discuss the case of 100% crushed vertices because, once we arrive at that, we could just linearly interpolate between it and the uncrushed position for intermediate values.

Our initial approach was to crush vertices all the way down to a "crush height," which artists would set per vehicle. And while this flattened the visual geometry as intended, it did so by taking all the vertices above the crush height and placing them in the same plane, creating an unrealistically smooth and jumbled mess. Our next attempt fared better, as we introduced a "crush factor"


to determine how close to the crush-height vertices on the vehicle it would be crushed to.

As an example, say a vertex is at height 5ft with a crush height of 1ft and a crush factor of .75. The vertex would be translated down 4ft to match the crush height, but since the crush factor is .75, it's only translated down  $4 * 0.75 = 3ft$ . Using this system ensured that proportions of different areas above the crush height were maintained, and it kept the crushed vehicle from degenerating into a perfectly flat mess. However, our results were still too smooth, so we decided to enforce a minimum level of collision deformation to accompany any crushing deformations. This allowed visual crushing to benefit from the geometric noise and damaged normal maps used for collision deformation, completing our 100% crushed visual.

Now that we knew what we wanted our crushed model to look like, the next question was how to arrive at the appropriate per-vertex crush percentages. We wanted every vertex underneath a tank to be 100% crushed and then to have a continuous falloff from there, so we projected the tank's bounding box onto the vehicle being crushed and made the strength of crushing morph weights inversely proportional to the distance from this rectangle. (Note that technically, it wasn't exactly a projection because a projection of a box wouldn't always guarantee us the rectangle we wanted for deformation reasons, but it was conceptually almost the same.)

Naturally, we were concerned about the cases where the high-fidelity visual deformation would look out of sync with the lower-fidelity DCPs, especially the central DCPs, which were particularly large and spanned the width of the car (as was seen in Figure 4). So, we built our pipeline to be able to support an arbitrary number and arrangement of central DCPs, and we only went with our arrangement of two for prototyping. However, when we began demoing deformation to the development team, no one noticed any of the problem cases. As it turns out, people rarely jump out of their tank to run around the vehicles they just crushed, and the remaining DCPs usually approximated the visual geometry close enough that any collisions between pursuing cars and crushed cars didn't look awkward.

In essence, our strategy for both tank and collision deformation is similar: create a simple framework of DCPs to support a high-quality visual-deformation

system, and link them where needed. Hopefully, demystifying the various parts of our approach and explaining the rationale that engendered them will help you jump-start your own deformation systems. 

**VICTOR CEPEDA IV** has been a programmer at Volition for nearly four years, working primarily on the physics, handling, and audio for Saints Row vehicles. He can be reached at [Victor.Cepeda@volition-inc.com](mailto:Victor.Cepeda@volition-inc.com).

The author would like to thank Justin Miller for his work on the art side of iterating deformation, Jeremiah Zanin and Shawn Lindberg for their physics advice, and Scott Kircher and Jason Lowe for their help figuring out the rendering side of things.



## BORDERLANDS 2 PUSHES NEW BOUNDARIES WITH UNREAL ENGINE 3 TECHNOLOGY

When Gearbox Studios' 2009 game *Borderlands* landed, it took everyone by surprise. The critically acclaimed title looked unlike any previous Gearbox release, and immediately distinguished itself as a groundbreaking Unreal Engine 3 powered game with its unique concept art visuals and innovative gameplay. Now the development studio is pushing Unreal Engine technology even further with an ambitious sequel.

"As a studio, we've been using Unreal technology for about eight years now," said Steve Jones, technical director on *Borderlands 2* and Gearbox Software. "So we've benefitted from a growing institutional knowledge of Unreal Engine 3 and how to best leverage its features and updates to support our goals for our games. The new Unreal Content Browser has really made a positive difference for us in the development in *Borderlands 2* – so much so that it would be difficult to imagine being where we are today without it. The official integration of Scaleform was a very welcome addition, as well."

Jones said many of the improvements the studio has made to its rendering pipeline aren't "visible" – they're behind-the-scenes optimizations that reduce the cost of rendering things in their concept art style. This allowed the team to spend extra memory and performance budget in other areas, which ultimately resulted in gameplay enhancements, including more enemies on screen, better weapon visuals, and more "badassery" across the board.

"The Unreal Engine's material system enabled our artists to add a significant amount richness and depth to the materials used on the guns," explained Jones. "Unreal Matinee is extremely valuable not just for in-game cinematics, but also as a critical part of our pre-visualization process. Very early on in development, we were able to identify new features we wanted to bring to the game, and Matinee was used to rapidly develop in-game

proof of concepts that enabled the team to quickly evaluate ideas with minimal development cost."

Gearbox employed Unreal Kismet extensively throughout the game, and Jones said that it's the first stop for designers seeking to add interactivity and life to the game world in a fast, flexible, and powerful manner. Jones' team was also able to make use of UE3's networking support to create co-op gameplay in the huge open world environments of *Borderlands 2* without cutting back the amount of enemies and loot available.

"In *Borderlands 2*, we pushed the presentation and gameplay of our enemies, action skills, guns, and missions (to name a few features) far beyond the first game," said Jones. "This is made possible by really leveraging what Unreal Engine 3 offers for a development environment, as well as the extensibility it offers licensees to tailor it to our particular needs."

Throughout the development process, Gearbox utilized Epic's Unreal Developer Network (UDN) extensively. Jones said UDN is an irreplaceable resource, a gathering of experts on UE3 technology assisting developers across a wide spectrum of issues encountered with game development.

"UDN has been helpful in a variety of ways, including initial framework planning, collaboratively working through an issue, proactively sharing solutions, and even documenting problems other people haven't encountered yet," said Jones. "It's an environment that fosters continuously sharing knowledge and we all benefit from that. All of the other licensees and Epic personnel contributing to UDN and its mailing lists are a fantastic resource for getting questions answered."

According to Matt Charles, producer of *Borderlands 2* at Gearbox, the team had lofty goals heading into this second installment. Charles said one of the biggest challenges facing the team was meeting such high expectations – including their own – since the success of *Borderlands*. The team took time to reflect on the original game before writing its own recipe for fun in the sequel. For the team, a desire to keep things fresh and avoid stagnation was important, but they also realized that *Borderlands* presents

a large enough world that they could explore in many possible ways.

During pre-production, Gearbox spent time looking at fan and press feedback on the original game, which resulted in some very real and positive changes to the sequel such as the new mini-map on the HUD. In addition to expanding on character growth paths, the team also added more richness to the game's missions.

Aside from its signature art style, open world environments, and those interesting characters, *Borderlands* is known for its massive amount of weapons. This time around, Gearbox has focused on bringing more personality to each gun, more distinction between manufacturers, and creating cooler visuals.

Gearbox has managed to showcase just how far UE3 can be pushed with the right art direction and creativity at the helm. PC, Xbox 360 and PlayStation 3 gamers will be in for a wild ride with *Borderlands 2*.

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



Canadian-born Mark Rein is vice president and co-founder of Epic Games based in Cary, NC. Epic's Unreal Engine 3 has won Game Developer magazine's Best Engine Front Line Award eight times, including entry into the Hall of Fame. UE3 has won four consecutive Develop Industry Excellence Awards. Epic is the creator of the mega-hit "Unreal" series of games and the blockbuster "Gears of War" franchise. Follow @MarkRein and @UnrealEngine on Twitter.

### UPCOMING EPIC ATTENDED EVENTS

**GDC 2012**  
San Francisco, CA  
March 5-9, 2012

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



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

# halo

combat evolved anniversary

## GAME DATA

XBOX 360



### PUBLISHER

Microsoft Studios

### DEVELOPER

343 Industries/Saber  
Interactive Inc./Certain  
Affinity Inc.

### RELEASE DATE

November 15, 2011

### PLATFORM

Xbox 360

## O

## ersary

/// When Microsoft/343 Industries approached Saber with a proposal to work on the remake of the original HALO: COMBAT EVOLVED, it was an opportunity that we couldn't miss. Prior to this project Saber had never worked on a major franchise, so we were looking forward to a new experience. We also have a lot of team members who are hardcore fans of the game (well, I guess most game studios can say this) who immediately got super excited. We didn't know much about the scope or the intended goals of the project, but we immediately assigned a team of artists to put together some initial concept pieces which demonstrated some of the potential visual improvement ideas, and within a week we were on a plane to Seattle to discuss the opportunity.

It turned out that 343 was looking for a complete remake of the original game, which needed to come out on the 10th anniversary of the original HALO's release. We would have just over a year to work on the project.

We spent the next few days brainstorming ideas and fleshing out the pillars of the project. What Microsoft Studios and 343 Industries wanted to do was a "true remake"—keep the original gameplay intact, but upgrade all the visual aspects of the game, making it look and feel like a true next-gen AAA title, and add a few new features that would make for a "better package" overall. This included things like co-op over Xbox LIVE, Kinect support, Skulls and Terminals, and the "toggle feature," which would allow switching between the old [Classic] and new [Remastered] visuals in real time. This feature proved to be very useful since it allowed comparing, in real time, the differences between the old and the new. People have great memories of the original game and think it looked great—and it did, 10 years ago. Toggling between the views helps players appreciate the improvements made to the game.

Every visual element was going to be remastered, including level geometry, textures, lighting, characters, weapons, vehicles, special effects, cinematics, and UI. The music audio would also be redone (only voiceovers remained unchanged). In terms of content, doing this in a year sounded like a crazy idea, but Saber and 343 believed that with proper planning it could be pulled off.

Obviously we were not making an original game from scratch, so we could rely on a lot of design decisions that made HALO: CE so successful. Very early on in the production cycle we established the list of assets we needed to build, and created a schedule to track the progress. In that sense the project was very predictable, since the scope of experimentation was relatively limited—we did not have to go back and forth on level flow, game mechanics and design issues, try different types of AI, balance weapons, or work on the story. All those decisions were already made. However, the



fact that we had to keep the gameplay intact imposed some important limitations and constraints. For example, it was decided not to change any of the in-game sandbox animations. Even though they look very dated and low-quality by modern-day standards, replacing the old handmade keyframe animations with new mocapped ones was too risky, because it would have affected AI, and generated too many bugs which could not be addressed given the short production timeline.

## WHAT WENT RIGHT

### 1 / overall engine solution.

The first production issue we had to resolve was the overall technical approach that would satisfy two goals: retaining the original gameplay, and allowing our artists and designers to achieve AAA visuals. We considered various solutions.

The first idea was to use the Saber engine and just reimplement the game, including AI behaviors, animations, gameplay mechanics, weapons, vehicles, and all the other components. This way we would be able to give the artists the tools they were familiar with, and we could rely on the AAA visuals and technologies our engine supported. We quickly discarded this option since we would never be able to recreate the gameplay exactly—that was simply impossible, even if we had unlimited time.

The second idea was to use the original game engine for gameplay, and somehow marry it with the Saber graphics engine and toolset. Imagine you are tasked with putting the mighty 8-cylinder, 7-liter engine of a 1963 Cadillac DeVille into the body of a modern stylish Audi A7, and making the engine take the inputs from all those state-of-the-art sophisticated electronics. Both pieces are amazing and feature great engineering—but they are also deeply incompatible. This is how Saber initially felt about marrying the original HALO: CE game code with Saber's own rendering engine and toolset—however, this seemed like the only viable solution.

Having thought more about it, we realized that this approach is similar to how our physics engine interfaces



HALO: COMBAT EVOLVED  
ANNIVERSARY's remastered  
visuals (top) compared to the  
original's (bottom).

with Havok—the physics world is represented within the Havok engine, and then the important pieces of information, such as object positions, velocities, and collision information, are transferred over from Havok into our game world through an interface layer. Following the same principles, we decided that we could create a proxy object for every game entity in the HALO world, and transfer the “vitals” from a HALO simulation into the Saber engine. These vitals included camera position, animation data, various game events (such as start/stop events for special effects), audio events, and more.

The advantage of this approach was that we did not touch the underlying game logic at all—it was merrily running on a dedicated

hardware thread, and the “only” thing we had to do was provide proper visualization of the game logic within our engine.

One of our engine leads, Roman Lebedev, was tasked with the implementation of the basic “stitching” of the HALO and Saber engines, and within a couple of months he was able to build the game platform to where we could run around the levels, shoot the guns, and kill the Covenant. The artists could go ahead and build the game assets right away, relying on the entire toolset—the assets were exported directly into the game, and all the features existing in Saber engine were readily available for creative use.

**2 / online co-op solution.**

One of the additions to the featureset of the original game was online co-op. HALO: CE supported only split-screen cooperative play, and initially Saber was doubtful that an addition of such a major feature was possible. Split-screen co-op support in the original game meant that most game systems such as AI and game events were already “made aware” of two players existing in the game world, but synchronization of the game worlds across the network was a challenge. In our own games, Saber uses dead-reckoning-based synchronization schemes which rely on extrapolation of the game world states. Implementing such a system for HALO ANNIVERSARY was very risky since every single game entity would have to be synchronized—and that seemed like too much work for the available time frame.

Greg Hermann, the technical lead on the project on the 343 side, and someone with a deep understanding of the original Bungie technology, came up with an elegant approach. The original HALO: CE game engine was built using a deterministic solution. In other words, given a certain game state and a series of inputs, the game would behave exactly the same every time you run it. So theoretically after the initial synchronization of the game worlds on two consoles running the game in online co-op, only the inputs needed to be synchronized—and then the simulation would continue in sync.

This concept seems simple on paper, but considering the complexity of the game engine and all the things that can break this determinism, Saber did not believe this would work. We ran the initial tests and almost immediately ran into desyncing issues which seemed to prove that this approach was bound to fail. Greg was adamant that the solution was viable, however, and he analyzed the nature of some of these desync issues. It turned out that most of them were related to a few features from the PC port of HALO: CE which were introduced after the release of the original Bungie version. After those few features were switched off (Greg’s change list affected over 40 source files, so it wasn’t exactly a simple effort), we were able to run the game deterministically—until

we ran into other causes of non-determinism. However, these initial tests proved the concept to be viable, and we ran with it. Roman built a suite of tools that recorded the game states in runtime and compared it, frame by frame and bit by bit, on two consoles running the game in online co-op, and whenever the simulations diverged he had to find the cause of the issue and make the fixes. So even though it wasn’t exactly a smooth ride, the solution worked, and in hindsight it was the only approach that allowed us to support the feature.

**3 / visual targeting/art direction.**

Saber was very fortunate to have Microsoft’s Ben Cammarano as the art director on the publisher side. For such a high-profile franchise as HALO it was critical to stay consistent with the original creative vision while working on our numerous art improvements. Ben traveled multiple times to our St. Petersburg office and spent long hours with Saber’s lead artist Dmitry Kholodov, working out the art strategy for ANNIVERSARY. He also served as a central point of communication between Saber and the 343 franchise team, and provided the overall art direction for the project. During his very first visit to Saber he established the four visual pillars of the franchise (heroic vistas, iconic imagery and characters, clean and vibrant aesthetics, and visceral action) which helped guide our art team.

Ben suggested a simple yet effective tool to communicate the vision within the production team, as well as to the upper management. He suggested the use of visual targets, or VTARs, which were pieces of concept art that we created for game environments. Saber certainly used concept art to establish reference visuals on prior projects, but that was more for inspiration purposes than for direct guidance. For HALO ANNIVERSARY we developed a different approach. Since Saber was high-resing the original game, we took screenshots in every level from HALO: CE and did paintovers that showed how those areas could be improved. These visual targets established lighting, texturing, environmental effects such as rain or fog, and overall mood. We showed these pieces to Ben and the franchise team, and they allowed

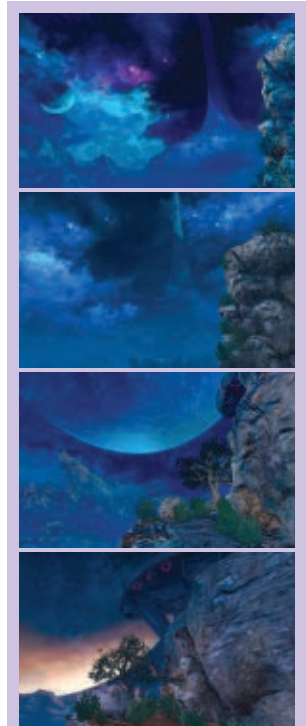
us to agree on the art direction well before the levels went into modeling.

This process may sound simple, but we were never able to put it to such a great use before HALO ANNIVERSARY. In the past when our concept artists created inspirational art pieces, their value was relatively limited since the actual level geometry was changing during the game design phase. For ANNIVERSARY, however, the level geometry was pre-established by the framework of the original game, so the concept artists could focus on ideas on how to make it look great. Having learned these lessons on ANNIVERSARY, we found an efficient way to utilize our sketch artists for level concepting for the original games we’re working on. Once the low-res level geometry is finalized, we take screenshots of a number of key areas and have the artists do pretty paintovers, which can be referred to by level and lighting artists as they go ahead with their high-resing work.

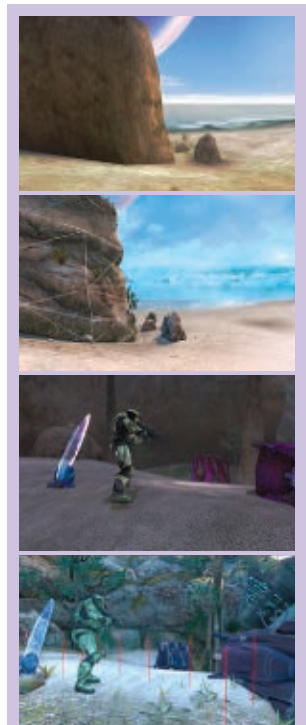
**4 / successful vertical slice.**

Saber went through a number of vertical slices in the past on other projects, but the one we had on HALO ANNIVERSARY was probably the most valuable. Even though we didn’t have to experiment with gameplay or game mechanics, there were plenty of things that needed to be proved out. Basic engine stitching technology, online and split-screen co-op, and art and cinematics production pipelines were all established during the first few critical months. In addition to that, we were able to find our stride with our overall approach to modernizing the sights and sounds of the game.

Visuals were certainly the focus for the entire HALO ANNIVERSARY project. We knew that the game would be judged based on how we performed on the art side. The initial feeling was that the art of the original game could be updated freely to maximize the visual impact. Prototyping the vertical slice level (Truth and Reconciliation) it was decided to improve the skies by repositioning the planets and the Halo ring to ensure their perfect framing for the most important viewpoints, changing the color of the planets to make them look more interesting, and changing the lighting. We spent a fair amount of time trying to achieve



Early planet placement and color vs. correct planet placement and color.



Differences in original geometry and Saber geometry. Saber artists needed to update collision meshes for new assets. Red beacons also marked geometry height differences.



those goals, and the final visuals were indeed great. However, after careful consideration and review with the franchise team it was decided that maintaining the universe of the original game was key, so we had to make a course correction and revert all those heavenly bodies to their original places. The same logic was applied later on to other art improvements—the things could be improved visually, but the universe had to remain the same.

Not only did this consistency to the original universe remain true to the creative vision of HALO: CE, but also it became a critical decision for the Toggle feature—imagine the planets and the ring shifting positions when the players activated the Toggle feature going back and forth between Classic and Remastered modes. This would have ruined the consistency of the game worlds.

### 5 / listening to the early community feedback.

// 343 first showed the game at E3 2011 behind closed doors, and then continued presenting it at various

industry events such as Comic-Con and Gamescom. When the first pieces of the game hit the internet, fans immediately commented on things. Obviously the game was not fully done at the time, and some trailers and screenshots contained some placeholder or incomplete assets—but it was great to listen to the early vibe of the community and focus efforts on improving the assets that were getting the most attention.

Because the game was developed against such a challenging schedule, sometimes placeholders stayed in builds longer than anticipated, or issues were given lower priorities by Saber and the franchise team—but the fans always picked up on them right away. One great example is the assault rifle. Saber originally used the assault rifle from HALO REACH as the rough-in for ANNIVERSARY's version. However, once the gun was already implemented into the game we realized that the design of the assault rifle changed from HALO: CE to HALO REACH. The changes were relatively small—but they would have been very noticeable to the hardcore fans.

So we decided that we had to use the version of the assault rifle from HALO 3 as the starting point. The gun was reimplemented into the game and was included into the official trailers that were released online. To my surprise, our lead engineer Roman (who also became a hardcore expert on the HALO universe through the course of development) showed me a post by a fan that said we had accidentally used the third-person version of the assault rifle for first-person view. These versions looked very similar—but they still were different. Amazingly enough, neither Saber nor 343 noticed this prior to the trailer—but thanks to the tip from one of the fans (and watchful Roman who noticed the post online), we caught the issue and fixed it.

There were other instances where early feedback from the fans helped improve the game. Saber came up with some great designs for the floor textures used in Forerunner structures—unfortunately, they were not fully consistent with the overarching philosophy of Forerunner culture, so we ended up fixing them.

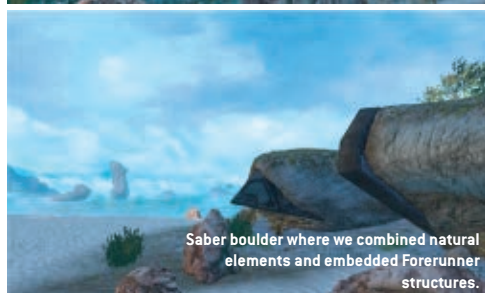
Ben and the franchise team were actually the ones who picked up on that first, but Saber was reluctant to make the change since it would require a significant amount of work and we just did not believe it was that important—but listening to the fans helped Ben win the argument.

#### WHAT WENT WRONG

### 1 / collision issues.

// Handling collision issues turned out to be a massive headache for the production team toward the end of the project. Some of these problems actually came up during the vertical slice phase, but because on a typical game project collision issues are very easy to fix we did not pay much attention to them. We were wrong. There were all kinds of collision issues, each of which had to be dealt with using a unique strategy.

Because the original game code remained intact, the entire game world simulation, including AI, shooting mechanics, object floor placement, vehicle driving, and basic player collisions were handled by the original game code, using the original HALO: CE geometry, which was obviously very low resolution. Imagine, for example, a beach section in a level such as Silent Cartographer, which was modeled with a handful of polygons, and was pretty flat. In HALO ANNIVERSARY, that area was made using a high-resolution mesh with a smooth surface profile that deviated a lot from the original. However, the enemies, the player, and the vehicles that moved on that beach handled their collisions using the original mesh, and therefore sometimes they were standing halfway into the ground, or floating in thin air. Weapons that enemies dropped sometimes also got stuck in the air, or fell through the ground and disappeared (but the HUD markers "Press X to pick up the weapon" remained on-screen). When players tried to move along the wall to stay in cover, sometimes they would bump into invisible collisions (if walls in the original game were in front of the walls in the remake), or sometimes players could walk through the walls and see through geometry (if the original walls were behind the ones in the remake). Sometimes the player was trying to snipe the enemy and



his bullets hit invisible objects, and sometimes a player was apparently shot through the walls.

At some point, our test team had generated over a thousand bugs on collisions alone—and we all realized that we couldn't delay tackling the issue. It was clear that we couldn't make the remastered geometry match the old one exactly in every inch of the game world—it would immediately make the graphics look low-res. We also knew that we could not afford to go through the entire game and adjust the geometry, because it was simply too much work for the time we had. Yet we could not ship the game with all those collision bugs.

Saber's initial suggestion was to build additional collision geometry that would "hug" the remastered geometry, and run all the collision detection using that—this makes collisions work in a typical game. 343, however, mentioned that this would change the stated goals of keeping the original gameplay intact. We also realized the impact this work would have on what we called "secret areas"—the areas players would not normally go into during a regular playthrough, but which can be accessed using some tricks. An area on the hilltops of Silent Cartographer, or the areas on the periphery of Guilty Spark are examples. There are many more of those areas in the game, and there is no real list that details them all. Apparently fans of the game spend years of their lives trying to break the boundaries of the original game, and every now and then there are discoveries of new secret areas and ways to access them. 343 and the franchise team were adamant that maintaining access to those areas was important because we just couldn't take them away from the fans. Unfortunately that meant that we couldn't add "Saber collisions" whenever we wanted to fix the bugs.

Saber's proposed solution was to create a list of different types of collision issues and address them using different techniques. The most benign type of collision was weapons falling from dead enemies—these were purely visual bugs, but even very slight variations in old and new terrain meshes made the weapons disappear underground. So we decided to use the "combined polygon soup" of old and remade geometry,

Up close, Master Chief is shoulder deep in geometry.



effectively landing the weapons on the higher of old and remade floors. This completely removed "weapons falling through the ground" bugs, but sometimes weapons continued to float in the air. It was decided that if the weapons floated above the ground by less than a certain threshold it was not really noticeable to the players, and in other places we had to bring the remastered terrain closer to the original. To make the process smoother, we built tools for the artists that helped them visualize the height differences between the original and remastered geometry—and then they went through the entire game and made the fixes.

This process also ensured that our remade terrain and floors matched the originals more closely, and now neither the player nor the enemies were floating, or stood waist-deep into the ground.

For "player seeing outside of the world through the walls" bugs we couldn't really adjust the wall geometry easily, because it would make the wall profiles very flat and low-res when viewed from the side, so in this case we had to rely on "Saber collision geometry." But then we could only do it in places where we were sure there were no secret areas around (and this caused heart palpitations at 343, because there were no guarantees there).

There were some special objects where we had to really think outside the box to come up with a solution. For example, giant boulders on the

beach of Silent Cartographer were originally modeled with a couple dozen polygons, and there was just no way we could high-res them while keeping the remastered geometry close to the original—either it looked like a smooth and round natural boulder, or it was a low-resolution 10-year-old object. Our level art lead Petr Kudryashov came up with an idea—what if we turned that boulder into a Forerunner-made object? That allowed us to end up with high-resolution visuals and still maintain the original shapes.

All in all, once we started to address the collision issues seriously we were able to find solutions that worked. Unfortunately many of those solutions required a lot of reworking of the assets and levels that were already done, and caused a fairly high number of bugs that needed to be fixed. Understanding the importance of the problem during vertical slice would have saved a lot of time.

## 2 / secret areas.

// At the beginning of the project nobody thought about these hard-to-access places from the original game. For a typical game, developers simply restrict access to those areas, and don't need to worry about high-resing additional art or work on performance issues related to seeing giant sections of the levels. Unfortunately for ANNIVERSARY we did not have that luxury—the franchise team established that those areas were an integral part of the game and they

had to be maintained in the remake (this was a very foreign concept to the dev team, and we had to reinforce it many times for the artists).

Similar to collisions, this was identified relatively late, so we had to deal with the consequences. The test and franchise teams were tasked with cataloging the known secret areas, and Saber began working on possible solutions. Because these secret areas were oftentimes on the boundaries of the normal playing sections, they served as backdrops, and the artists modeled them as such. For example, the hilltops in Silent Cartographer looked great when viewed from a distance, but if we allowed the player to go there, it looked like a mess; the player was standing shoulders-deep in the ground, he was walking through the geometry, and there were massive performance problems because visibility was not controlled and the player could see almost the entire level.

Saber artists quickly realized that they could not model those areas to work both as backdrops and playable areas because the modeling techniques are just so different. So we all had to make a tough decision to only address collision and performance issues, and leave the art mostly as-is—after all, those areas did not look that great in the original game either.

## 3 / streaming issues.

// Saber had a solid audio streaming solution as part of our TimeShift engine, and we were working on



# MAGIC PIXEL GAMES

We craft original games.

We love what we do and we want your help.

## WE'RE HIRING!



ENGINEERS, ARTISTS, DESIGNERS

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

©2012 MAGIC PIXEL GAMES, ALL RIGHTS RESERVED

texture streaming for quite some time to increase our texture budgets. This tech was ready by the time we started to work on the project, but we had never used it in a major game release. Texture streaming is one of those technologies that is very dependent on the overall state of the project. Until you have your levels, assets, and textures mostly done, memory requirements optimized, audio integrated, and engine stable enough to start testing with DVD emulation, it's very hard to evaluate the quality of texture streaming. Unfortunately, when all those components are ready, the game is close to being done and there is very little time for testing and tuning. That was when we started to pay close attention to texture streaming and realized there were quite a few issues.

Basic texture streaming worked like a charm, but many special cases presented problems. Whenever we had to load massive volumes of textures such as in levels or checkpoints, toggling between Classic and Remastered views, or changing camera position abruptly before, during, and after the cinematics, there were texture streaming issues. We realized that on average it takes about 5 to 7 seconds to load all the needed textures from DVD, so either we showed an image with low-res textures, or we had to block the game execution, preload the textures, and then continue rendering. While the game was loading textures we would show a black screen that faded into the game. This simple idea became a generator of various bugs, starting from compliance issues (can't show black screens for longer than 5 seconds) to synchronization issues in online co-op to keeping the picture and audio in sync for cinematics. Things got worse when we integrated in-game audio and started to test in emulation. Because we did not have audio and texture streaming balanced, sometimes it took up to 25 seconds to load all the high-resolution textures.

Unfortunately the only solution was to fix all these issues one by one, and do a lot of balancing, and this was one of the areas of the project that required quite a significant amount of crunch time.

#### 4 / lighting solution.

// For a long time Ben mentioned the need to improve our lighting

solution, especially for outdoor environments—but unfortunately there was never enough time to give it serious consideration. At the beginning of the project Saber was focused on engine stitching and building the game platform. Then came various high-priority integration and performance issues. We simply couldn't find the time to work on lighting solution improvements.

However, midway through the project, while working on our first public demo (The Silent Cartographer level which was shown at E3), our rendering leads decided that we just had to find the resources and make the push—HALO levels were massive, and combined both indoor and outdoor sections in a single scene, and our existing tech did not handle it that well. The root cause of the problem was traced to how we set up ambient lighting. Our artists simulated global illumination by placing multiple local lights in the scene, which was fairly time-consuming. And if that worked well for indoor environments, it wasn't very practical for large outdoor sections. So our tech group suggested a solution to simulate ambient lighting using a simple six-direction control which allowed us to set up lighting coming from different angles in the environment.

This simple tool worked surprisingly well, and it was quickly integrated into our engine and tools pipeline—but unfortunately this all happened late in the project, so to achieve a consistent look for all the levels the lighting artists had to go back and adjust levels that were already done and approved. Considering the lighting artists were already stretched very thin, this unexpected volume of work wasn't a welcome surprise. Still, the efforts really paid off and we managed to improve the quality of our lighting quite a bit.

#### 5 / cinematics.

// Saber and 343 decided to handle HALO ANNIVERSARY cinematics using a real-time in-game solution (as opposed to prerendered movie playback). Saber had a robust timeline-based technology to allow for quick manipulation of special events, mocap data, and other components, and we did not expect to run into production issues here.

Early on it was decided that we could explore a good degree of creative freedom to improve upon the experience by enhancing camera angles and adding more action and characters around the main events in the scenes. Saber was going to replace all character animations with mocapped data and add finger and facial animation.

Handling mocap for HALO cinematics was the first challenge we had to overcome. Because the game was using the original voiceovers, we had to record mocap animations and facial performance to perfectly match the timing of the original game. To achieve that, we played video clips of original cinematics shot on a large screen at a mocap studio, and the actors were acting out the scenes trying to match the timing. We knew it was the only solution, but it was difficult for the actors to match the voiceovers perfectly. We ended up manually tweaking a lot of facial animations to ensure proper lip-sync—but we were not too happy with the final results.

Saber has an automated tool that renders all game cinematics offline every night after the daily build is compiled. The tool is invaluable for detailed cinematics analysis; we can play the latest renders of cinematics in a video player and analyze them in detail. Because the project was done so quickly, however, we didn't really have time to sit back and do a careful evaluation of the cinematics until after we hit Alpha—and then we started seeing all sorts of issues which were exposed thanks to this tech. Streaming issues, audio syncing glitches, animation bugs, lip-sync—all these problems resulted in over 300 bugs, which were entered into the database at a critical point of the project. In addition to technical bugs, after a careful review with Ben and the rest of the franchise team we realized that sometimes we went too far with our attempts to improve upon the original experience. For example, by using a real-life female mocap actor for Cortana and having the actor play out Cortana's emotions we made the holographic model "too dramatic." Yes, the animations were perfectly smooth, but that exaggerated emotions being acted out affected the "artificial computerized feel" of Cortana's character in the game.

Because it was too late to go back to the mocap studio, our animators had to combine the original key-framed animations with the new ones to achieve the desired results.

Looking back I think most of cinematic issues were caused by a lack of proper preproduction and early analysis of the results of this particular area of the project. In the end, adding more animators and working closely with the Microsoft QA team helped solve most of the issues.

#### HAPPY ANNIVERSARY!

// Overall, HALO ANNIVERSARY was a very smooth ride. Even though the timelines were short, the fact that the scope of experimentation was limited primarily to engine tech and art helped reduce risks. The core decisions made at the beginning of the project helped focus the team on the important issues. The production team assigned by 343 made communications easy. Our producer Dennis Ries did a phenomenal job championing the project within Microsoft and to the press, and got a ton of frequent flier miles shuttling between Seattle and St. Petersburg.

The initial decision to make a complete overhaul of the game art that didn't stop short at just adding better materials, throwing in a couple of shadowmap lights, improving UIs, and adding the achievements really paid off. The addition of new exciting features such as online co-op, toggle for Classic/Remastered, Skulls and Terminals, and Kinect made the offering even better. HALO ANNIVERSARY went on to get multiple "Best of E3" nominations and awards (which is unusual for a remake), and many players and reviewers alike praised the game as "the best remake ever made." This was our original (though unstated) goal and Saber is very proud of having accomplished that. Working on a top franchise with a great production team taught us a lot of lessons which made the studio stronger in many respects. 🎮

**ANDREY IONES** is COO and founder of Saber Interactive, and has almost 20 years of experience in the game industry. He started by working on technology for pre-rendered graphics for major titles published by Activision and Electronic Arts in the '90s, and has a PhD in computational geometry.

# Auto Club Revolution\* and Intel

## Making your game faster, faster



*"Intel® Graphics Performance Analyzers proved key bottlenecks and other issues with Auto Club Revolution."*

– DR  
EUTECHNYX CHIEF

Intel® Graphics Performance Analyzers (Intel® GPA) are a suite of graphics analysis tools that help game developers make fast games run even faster. Intel GPA was developed with game developers in mind, using your feedback to ensure the tools work the way you do. And because Intel GPA is fast and easy to use, you'll improve efficiency and save time, getting your game to market even faster.

**DOWNLOAD INTEL® GRAPHICS PERFORMANCE ANALYZERS FOR FREE at [www.intel.com/software/gpa](http://www.intel.com/software/gpa)**

© 2012 Eutechnyx. All Rights Reserved. Image courtesy of Eutechnyx. Copyright © 2012 Intel Corporation. All rights reserved. Intel, the Intel logo, are trademarks of Intel Corporation in the U.S. and other countries. \*Other names a

# Intel® GPA<sup>†</sup> Master.



## Intel® GPA System Analyzer

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

- A heads-up display (HUD) for real-time performance analysis
- Create DirectX state overrides and conduct real-time experiments
- Triage system-level performance with CPU and GPU metrics
- Game pause, step, resume

## Intel® GPA Frame Analyzer

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

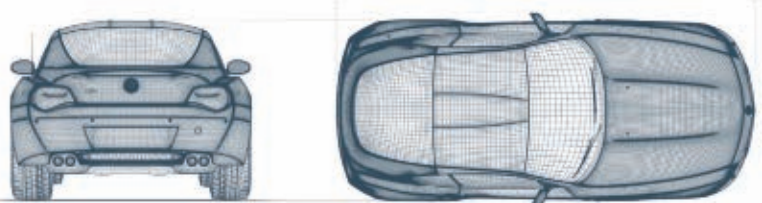
- Support for Microsoft DirectX\* 9/10/11, Microsoft Windows\* XP, Microsoft\* Vista, 7
- Deep analysis of individual draw calls
- Make changes and see visual and performance effects immediately
- Capture and share frames

## Intel® GPA Platform Analyzer

Visualize performance of your application's tasks across the CPU and GPU to ensure the software takes full advantage.

- HW thread view of game showing thread context
- Visualize performance of multi-threaded CPU/GPU tasks
- Media Analyzer for real-time and trace analysis
- Intel® OpenCL and browser profiling abilities

To learn more about Auto Club Revolution, visit:  
[autoclubrevolution.com](http://autoclubrevolution.com)







# WACOM Inkling

REVIEW BY MIKE DE LA FLOR



**FIGURE 1:** The Wacom Inkling is an innovative product that aims to allow artists to sketch on paper away from the computer, but the only thing the Inkling can do at this point is digitize very rough sketches.

From the ArtZ tablets of 20 years ago to the modern Intuos and Cintiq tablets today, Wacom has provided millions of artists with innovative tools that make working with computers feel natural. In late 2011 Wacom began shipping the Inkling, a new product that gives a glimpse into what the future may hold. It's a radical departure from the desktop digitizing tablets in that it allows artists to sketch away from the computer but still produce digital sketches.

The Inkling consists of a combined receiver and flash drive device about the size of a small box of paper clips and a pen. Both the receiver and pen are stored in a sturdy carrying case that doubles as a mini-USB-powered charging station and docking platform.

The Inkling requires some assembly straight out of the box. The rechargeable nickel-metal-hydrate pen battery has to be installed and the pen slipped into its slot in the case so that both the pen and receiver can be charged. Included is the Inkling Sketch Manager software, which is needed to calibrate the Inkling and work with sketches.

The Inkling's receiver is also a drive, which means that it must be ejected to avoid damage and not just yanked out from the USB port. Once ejected, the Inkling may be clipped onto paper or a sketchbook. Afterwards, you can just turn on the receiver and pen and begin sketching. The Inkling uses ultrasonic and infrared signals to track the artist's strokes and record sketches to the drive. As a result, Wacom cautions that the Inkling may not work well in direct sunlight or near ultrasonic noise sources. However, in normal office or studio environments, and even outdoors, the Inkling worked well.

The Inkling's pen has a similar feel to other Wacom pens, though it looks different because of the battery compartment. The main difference between the Inkling's pen and an Intuos or Cintiq pen is that the Inkling's actually uses real ink. The Inkling's pen should be gripped about an inch above the tip so as to not obstruct signals between the pen and receiver. This is a bit higher than other Wacom pens, and it could prove awkward for some artists.

Sketching is a personal business, as artists have different sketching styles and media preferences. However, the Inkling limits artists to ink, which may be a deal breaker for those who prefer pencil. The ink cartridges provided with the Inkling are standard ballpoint pens. Artists who sketch with ink almost never do it with ballpoint pens, which tend to glob up and smudge, skip, provide little control, and would be akin to sketching with a BIC or Paper Mate. Wacom would do well to provide artists with options for different pen types in the future.

Before discussing the Inkling's performance further, it's important to emphasize what it is and isn't. The Inkling is designed to digitize rough sketches, such as the preliminary drafts of game characters, while

## WACOM Inkling

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

### PRICE

> \$199

### SYSTEM REQUIREMENTS

> Windows 7, Vista, or XP (SP3, 32 bit, or 64 bit versions), Mac OS 10.4.0 (or later), Adobe Photoshop or Adobe Illustrator (CS3 or later), and Autodesk SketchBook Pro or SketchBook Designer (2011 or later)

### PROS

- 1 Can sketch with pen on paper
- 2 Innovative workflow
- 3 Layered sketches

### CONS

- 1 Misaligned strokes
- 2 Finicky pressure sensitivity
- 3 Sensitive to environment

away from the computer and still provide the artist with editable file formats. The Inkling is not designed to produce final drawings, and it will not replace an Intuos or a Cintiq tablet—at least not yet.

The Inkling should be calibrated before sketching, which involves selecting the receiver's position on the paper, selecting the paper size, and calibrating the pen's sensitivity. Assuming all goes well while working with the Inkling, the resulting digitized sketches are acceptable (albeit rough)

facsimiles of the original sketch. Nonetheless, regardless of how closely one adheres to the Inkling's best working practices, it too often produces sketches that have misaligned or missing strokes. And since there is no way to predict where these problems will occur, this is a constant hazard when working with the product.

## ROAD TEST

» Further evaluation of the Inkling reveals that it works best with a loose, sketchy style that favors many overlapping strokes which produce tone and texture. This appears to somewhat compensate for missing or misaligned strokes. The Inkling does not work well with a more structured sketching style, such as straightforward line art.

An impressive feature of the Inkling is its ability to create layered sketches. To create a new layer while sketching, simply press the new layer button on the receiver and continue sketching. When the sketch is previewed in either the Inkling Sketch Manager or with any of the designated programs like Illustrator, the file will contain layers.

I found that any nudge or bump of the receiver could frequently cause gross stroke misalignments, rendering the digitized sketch



**FIGURE 2:** The image on the left is a scan of a very quick sketch for a creature idea rendered with the Inkling's pen. On the right is the Inkling's digitized sketch opened with Autodesk SketchBook Pro. At first glance, the sketch on the right looks pretty good; however, on closer inspection, the misaligned strokes, gaps, and missing strokes become obvious. Also, the line quality in the original is more subtle than in the Inkling rendition.

useless. This is particularly annoying because you won't notice these serious misalignments until you view the sketch digitally. To minimize this problem, the receiver should be clipped onto several sheets of paper to create a snug grip. Avoid excessive jostling while sketching.

The Inkling boasts 1,024 levels of pressure sensitivity, which means that it should render strokes of

perceptible varying thickness. In practice, however, the Inkling did not perform well in this area. The first problem is that ballpoint pens are not designed to create strokes of different thickness. Thus, there is no visual feedback while sketching as to what the stroke thickness will look like in the final digitized sketch. Second, the only software in which there is any indication of varying stroke widths is in Autodesk SketchBook Pro, but even then the line widths were nowhere close to 1,024 levels. Further, only when the sketch was opened directly with SketchBook Pro, bypassing the Inkling Sketch Manager, were varying line widths noticeable. When exported to Photoshop or Illustrator via the Inkling Sketch Manager

there were little or no varying stroke widths visible in the sketches.

### SKETCH MANAGER

» The Inkling Sketch Manager has three main functions when it comes to the sketches. First, it organizes and displays all sketches as thumbnails. Second, it exports the sketches to other programs, such as Photoshop. Third, you can use it to edit layered sketches by hiding, merging, and deleting layers. In general, the Inkling Sketch Manager does its intended job, but the software appears unfinished. The user interface has a dated 1990s feel to it and usability is poor. For instance, the Inkling Sketch Manager has a feature that plays back a sketch as a movie, but there is no way to export the movie, making the movie feature somewhat pointless. So what is the point of the player? It would have been preferable for Wacom to focus on polishing the Inkling Sketch Manager instead of adding extra gimmicks.

Arguably, the best way to use the Inkling is directly with Autodesk SketchBook Pro, bypassing the Inkling Sketch Manager entirely. SketchBook Pro can open the .wpi Wacom file format (in which the sketches are saved) and the SketchBook Pro brush engine does a better job of parsing the stroke data, resulting in smoother, better-looking sketches. In addition, when an Inkling sketch is opened with SketchBook Pro, the sketch will take on the properties of the current

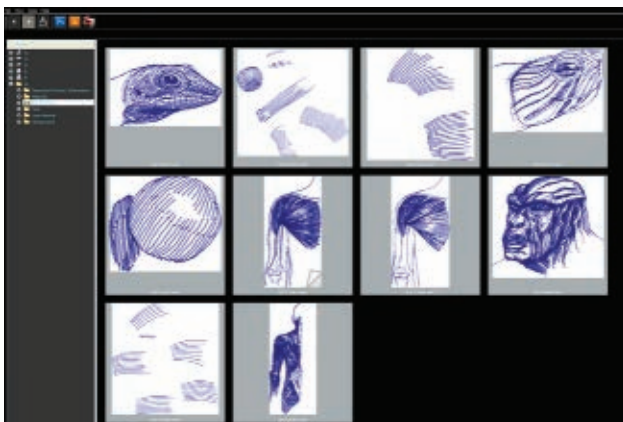
document and that of the selected brush. Even if the target platform is Photoshop, it's still better to go through SketchBook Pro than the Inkling Sketch Manager.

The obvious drawback to using just SketchBook Pro and bypassing Inkling Sketch Manager is that strokes will not be converted to vectors if the target application is Illustrator. For that task, the Inkling Sketch Manager is necessary. However, when exporting to Illustrator, the resulting vector drawing is unimpressive: line widths and quality are poor, lines are often so densely packed with control points that editing is impractical, and the resulting file sizes are extremely large for vector files.

All criticism aside, the Inkling is a remarkable and innovative tool, but the technology has not matured enough to make the Inkling practical. At this time, the Inkling is probably not a realistic addition for busy pipelines. It would be much simpler to scan a sketch or to use Illustrator's Live Trace to convert the scanned sketch to vectors. However, the Inkling is a welcome reprieve from being tied to a computer, and with further development, it could change how artists work in the future. 🎨

**MIKE DE LA FLOR** is a freelance medical illustrator, instructor, and writer. He's the co-author of the recent title *Digital Sculpting with Mudbox: Essential Tools and Techniques for Artists as well as other CG titles*.

**FIGURE 3:** The Inkling Sketch Manager software is used to calibrate the Inkling and manage sketches. While the software does its job, it has an unfinished feel to it. Further, SketchBook Pro does a better job at parsing the Inkling sketches than the Inkling Sketch Manager.





Developers helping developers

[www.igda.org/join](http://www.igda.org/join)



FAR CRY 3

# THE SPUS ARE HUNGRY!

## MAXIMIZING SPU EFFICIENCY ON FAR CRY 3

The **FAR CRY 3 PS3 engine is extremely SPU-intensive**, with over 60 different job types and 1,000 job instances executed each frame. Much development effort has been spent optimizing the scheduling and execution of SPU jobs—in particular in reducing the scheduling load on the PPU. This article will describe some of the techniques used in the FAR CRY 3 scheduling system APIs, which manage each of the dependencies and synchronization paths that a PS3 game needs: SPU job(s) to SPU job(s), SPU job(s) to PPU task(s), and SPU job(s) to RSX command buffer. Please note that all listings will be available at [www.gdmag.com/resources](http://www.gdmag.com/resources).

### PRELIMINARY NOTES

» The provided code samples are written in pseudo C/C++, and may not be functional as is. They are here as a reference to illustrate the given explanations.

All `AtomicXXX()` functions stand for the corresponding “XXX” atomic operations. The functions are considered available on both SPUs and PPU. In both cases they operate on references of main RAM variables. That means that dereferencing an EA in the argument list on SPU will just pass the corresponding EA to the function.

Be aware that writing `structureEA->SomeMember` on an SPU will be compiled as simple pointer arithmetic. It will be resolved as a reference EA of this member. This means that passing the statement to an atomic function on SPU is perfectly correct.

Types such as “u32” stand for “unsigned 32 bits” value.

Finally, some SDK functions may have different signatures for the PPU and SPU. Even if some code samples should build on both the PPU and SPU, only one version is shown for readability.

### PPU SCHEDULING

» In order to allow task scheduling on the PPU, a scheduling engine must be in place. Since that isn't the topic of this article, I won't describe the techniques we're using on FAR CRY 3. But I will assume that a task dispatcher is available and running on the PPU. The dispatcher should own a collection of threads and be able to retrieve waiting tasks from a shared container (typically lock-free). Scheduling a PPU task is as simple as pushing a task into this container.

### LOW CONTENTION CIRCULAR POLLING JOB CHAINS (LCCP JOB CHAINS)

» Our job system is based on SPURS jobs, because they provide input/output data pipelining and don't waste too much of the precious SPU local store memory, leaving 234KB for user code and data. Jobs can be launched using job chains. This type of SPURS workload is based on a command buffer. Typically you insert a collection of jobs into the command buffer and then launch once it's filled.

We are using SPURS job chains in a different way in order to provide instant launch. Our job chain command buffer is filled with `JumpToSelf (J2S)` commands. This command blocks the job streamer in an active state. Our command buffer is also made circular, by inserting a “Jump to



first” as its last entry. Launching a job is just a matter of replacing a J2S by a “job” command. The job streamer will immediately launch the associated job and move to the next J2S command. The job itself will overwrite a J2S at the command buffer location used to launch it. The slot can then be reused when a full cycle of the circular buffer has been performed. This scheduling technique requires some code execution both at schedule time (to insert the job command), and also at execution time on the SPU (to overwrite the job command).

This means that the SPU job scheduled with this technique needs to embed this additional code and execute it at start-up. We provides a library that embeds the `cellSpursJobMain2()` job entry point, and performs those tasks automatically before calling an alternate entry point implemented by the end user. A programmer who wants to schedule a job through the system only needs to link against this library and rename its `cellSpursJobMain2` entry point.

So far we need the data you’ll find in code sample 1 (again, available at [www.gdmag.com/resources](http://www.gdmag.com/resources)). You’ll notice that the command buffer size is marked as optional. In FAR CRY 3, all job chains have the same size, defined with a compile time constant. This optional data will be omitted in the code samples we’re referencing going forward. The algorithm we have at this point may look like code sample 2—but there are still two major issues that need to be addressed before we continue coding.

The first is the prevention of buffer overrides. Our command buffer can easily be filled faster than it is emptied by the job streamer. Since it is circular, we could face the case where our insertion point has reached the position currently processed by the streamer. In such a case, the command at insertion index is not a J2S, but rather a pending job: We need to wait for this job to execute before reusing the slot. But while waiting, we can face the opposite situation, in which the job streamer has completely emptied the command buffer and has reached the very same position. If the job at that position has not started yet, nothing would prevent the streamer from rescheduling it (and the subsequent jobs). This example demonstrates the need for a technique to prevent the job streamer from processing part of the command buffer in which there are already jobs that have not started yet.

Our solution is to slice the command buffer into “guarded chunks.” Each chunk is a collection of  $\{n\}$  adjacent slots in the command buffer, for which we are keeping a count of pending jobs. Each time we schedule a job from a new chunk, we initialize an associated counter to the number of slots it contains. Each job will decrement the counter of its chunk upon execution. By simply forbidding the scheduling index to enter a busy



chunk (by checking that its counter is not 0) we can solve the buffer override problem.

For instance, let’s say we have chunk  $\{n\}$  and chunk  $\{n+1\}$ , and the position index for insertion is pointing to the last entry of chunk  $\{n\}$ . This last slot is necessarily a J2S because, as the rule states, we cannot enter a busy chunk. So when we entered that chunk, all slots were J2S; and all unused ones are necessarily still J2S. Consequently, the job streamer cannot go further than this slot. As soon as chunk  $\{n+1\}$  is free (i.e., its counter reaches 0), we know that all of its slots are filled with J2S. We can safely reload the chunk  $\{n+1\}$  counter, increment the position index (i.e., enter chunk  $\{n+1\}$ ), and insert our job in the last slot of chunk  $\{n\}$ .

We still have one more issue to consider here. Our job chain is meant to be shared by all PPU threads that want to schedule a job. Therefore, the scheduling algorithm accessing the job chain shared data (the insertion index and the guard indexes) must be thread-safe. Also, when dealing with the chunk frontier, we must ensure that the index and the guard counter are updated together, atomically. Even though the PS3 is capable of 128-byte atomic operations

on a full cache line, the SDK API only provides helper functions for 32-/64-bit atomic operations. Packing the insertion index and guard indexes into a single 64 bits will enable us to use those simple functions, while saving some space in our data structures. Our updated data structures are now as shown in code sample 3.

We could also have some optional data in here. To figure out which bits to update in the `m_guards` variable we need to know how big the command buffer is, and how many chunks we have. With this information, `cmdBufferSize/chunkSize` gives us the chunk count; `48 bits / chunkCount` gives us the number of bits per chunk, and the `insertionIndex%chunkSize` gives us the `chunkIndex`. Again, in FAR CRY 3, all those variables are compile time constants and so will be omitted from further code samples. We are using a 1024 slot command buffer sliced into four chunks of 256 slots. So our 48 bits are divided into four counters of 12 bits. We just need to pass the guard index to figure out which bits to update.

See the updated algorithm in code sample 4. You’ll notice that the concurrency is handled with a typical local copy-local update-atomic exchange loop. If you are not familiar with lock-

free algorithms, just see the for(;;) bloc as a critical section secured by a lock.

The second major problem we have to face is the unnecessary preemption of SPUs due to active polling. Our job chains are always running, and a J2S command is like a while(1); statement for the job streamer. Such a statement is consuming CPU, keeping it busy doing nothing. SPURS includes mechanisms to prioritize workloads of the same priority on a given SPU against a workload blocked on a J2S. But a workload with smaller priority will never get a chance to update.

To solve this problem, we are using the "contention" properties associated with a SPURS job chain. This property lets us specify the number of SPUs associated with the job chain, and can be changed on the PPU as well as SPUs. All we have to do is to keep track of the number of pending jobs for a given job chain. When it is below the number of available SPUs, we only need to adjust the contention accordingly.

When scheduling a job, we increment the counter and adjust the contention to  $\text{Min}(\text{numSPU}, \text{pendingJobCount})$ . When a job starts execution, it decrements the counter and makes the same adjustment. But this method causes a concurrency problem: We cannot update our counter and the contention in a single atomic operation. We have to perform two different steps: Increase the counter, and then change the contention. Between those two steps, several other jobs may have either started execution or have been scheduled, changing the contention. A solution is to append an "operation counter" to the pending job counter in a single 64 bit field. Each time the counter is changed, the operation counter is incremented. We can now perform a post contention-change read of the counter, and compare it to the pre contention-change read of the counter. If both are equal, the contention change is validated; if not, it needs to be performed again. Our updated data structures and algorithm now look like code samples 5 and 6.

And here we are! We now have a thread safe LCCP job chain that launches a job as soon as it is scheduled! On FAR CRY 3, we have an additional layer of management, to automatically create or select a job chain depending on the affinity and the priority we want to have for the job. This layer automatically creates JobSchedulingData and JobExecutionData and fills their appropriate fields.

### SCHEDULING SPU JOBS FROM THE SPU

» It is quite common for SPU jobs to have some dependencies on other SPU jobs. For instance, data generation tasks are often split into different jobs with a last job to merge the results into a final buffer. Such dependencies can be handled in a single job chain using a SYNC command. But this is not possible with our job

chains. In these cases it would be interesting to be able to schedule the "merge" job directly from the last "generation" job, enabling us to treat any kind of SPU-to-SPU dependencies directly from the SPU. It would also enable our job scheduler to operate without PPU intervention.

You may have noticed in the code samples that there is not a single pointer dereference. We are always accessing (set or get) values using an atomic operation using the effective address in main RAM. That means that this code can run

**Interrupt Handler** One way is to use an interrupt handler with the `cellGcmSetUserHandler()` and `cellGcmSetUserCommand()` functions. This will insert a command into the provided RSX command buffer. When the RSX reaches this command it will generate an interrupt on the PPU, which will call the corresponding handler. You can then schedule all the jobs you want from this callback. This simple mechanism has a drawback: It has a high cost on the RSX as well as on the PPU. If called just once or twice per frame then its simplicity makes

Scheduling a job takes about 2,000 cycles, and the job typically starts its execution in 10us. Waiting PPU threads (using passive synchronization) are usually woken within 50us. Since jobchains are shared, we use only four of them. All of the features only cost an additional 7KB of SPU code in each job, and take around 2us for execution.

on the PPU as well as on SPUs! Atomic operations take an EA on the data they need to change, and directly change it in the main RAM (actually, changing the full cache line).

That's it; we can now schedule a SPU job from the SPU! All you have to do is to retrieve (through a DMA, or directly from your job input data) a JobSchedulingData associated with the job you want to schedule. Simply call the schedule function on it. Congratulations! You are now able to manage SPU-to-SPU dependencies!

### SCHEDULING SPU JOBS FROM THE RSX

» On FAR CRY 3 we have a lot of rendering jobs that are performed to help the RSX. Some of those jobs take RSX-generated data as input. This means that they need to be launched only when the RSX reaches a given point in its command buffer. We need a way to schedule those jobs directly from the RSX. There are 3 different ways to do this.

It worth the performance cost. But when you start launching tens of jobs per frame, an alternative technique should be used.

**Polling Job chain** The second method uses a dedicated polling job chain. This kind of job chain has a small command buffer. The first slot is set to J2S, the second to the job. The job chain is started at schedule time on the PPU, and the job streamer stays blocked on the J2S. This J2S can be overwritten at any time with a NOP, enabling the job streamer to advance to the next command and launch the job. Writing a 64-bit value in the main RAM from the RSX is perfectly possible as long as the buffer is allocated in RSX-mapped memory. For example, using the `cellGcmInlineTransfer()` function will do the job. The downside of this method is that it involves some polling. If we don't want to waste an SPU to do this polling, we have to set a low priority on the job chain and a contention of one. The





RSX-scheduled job will only get a chance to be executed when an SPU becomes idle.

**Workload flag** The third method is to use the “workload flag.” Each SPURS instance has a workload flag that can be associated to a single workload. SPURS PPU threads will take care to start this associated workload as soon as this flag is set to 1. We can create a job chain filled with our RSX-scheduled jobs, and associate it with the workload flag. Then we can add a `cellGcmInlineTransfer()` to the RSX command buffer that will overwrite the workload flag. The downside of this method is that we can have only one RSX-scheduled job chain at a time, because there is only one workload flag. However this associated job chain can be set as high priority, which means that the associated jobs will start as soon as the workload flag is set.

### RSX SCHEDULING: THE SWITCH OF MIND

» The implementation details of FAR CRY 3 won't be disclosed in this article, but the described methods above say it all. We are using both the polling job-chain and the workload flag method to schedule numerous jobs from the RSX. Examples of RSX triggered jobs are edge culling, shader patching, and deferred lighting. But scheduling many jobs from the RSX raises an issue we had not foreseen. The PPU frame and the RSX frame are not in sync, but both units are now scheduling many SPU jobs. When both units try to launch a large batch at the same time, we may experience serious SPU contention. The problem started to appear for us when we moved the scheduling of our first batch of deferred lighting jobs onto the RSX. Those jobs were competing with culling jobs for much of the time.

Then we realized that scheduling some of our rendering jobs from the PPU, and others from the RSX will always create this kind of issue. At this point we decided that we would schedule all of our rendering jobs from the RSX. This makes a lot of sense: The SPU is used as a GPU coprocessor in that situation. It is completely natural to schedule those jobs directly from the RSX directly instead of going through unnecessary steps on the PPU.

### SCHEDULING PPU TASKS FROM THE SPU

» As introduced in the first section, we assume that PPU tasks can be scheduled by simply pushing them to a shared container. This container (a simple queue) should be lock-free, which means it can be operated with simple atomic operations using effective addresses. This container can be manipulated from an SPU simply by passing its EA. If the container methods `PushBack()` and `PopFront()` only operate on their members through atomics, no DMA or transfer is required. Given those

prerequisites, scheduling a PPU task is as simple as retrieving its EA on the SPU, and pushing it on to the lock-free container. [See code sample 7.]

In some cases we don't really want to schedule a complete PPU task, but rather execute a quick update on the PPU as soon as possible—to enable an update of several variables spread across the engine, for example. This prevents dozens of nasty DMA operations following a chain of pointers to finally update a single Boolean variable. That kind of update can be executed through a high-priority PPU tasks workload. This workload is a second lock-free queue, which can be consumed by all regular PPU threads, but also by another very high-priority thread. This thread is sleeping most of the time on a SPURS event, and when awakened it empties this secondary workload before returning to sleep. When an SPU wants to execute this type of task, it pushes it into this second workload, and sets the SPURS event. [See code sample 8.]

The downside of this method is that the high-priority thread, when awakened, will preempt another running thread. This will cause a thread switch and will probably trash the code and data caches. Using too many of those high-priority callbacks will have a non-negligible impact on the overall PPU performance. The system should be used with care and only for very specific use cases.

### SYNCHRONIZING PPU AND SPUS

» One last feature we're missing is a simple way to synchronize the SPUs and the PPU. We can differentiate two types of synchronization: passive and active. A passive synchronization puts the waiting PPU thread to sleep until the waiting condition is satisfied (i.e., the job(s) we are waiting for, is (are) done). An active synchronization is performed by polling the condition until it is satisfied. The passive synchronization allows the system to give the CPU to another thread, which maximizes the overall occupation. However it suffers from slow response time. The active synchronization wastes some CPU cycles but has an excellent response time. So both techniques have their pros and cons. Depending on the situation, one will be more appropriate than the other, but both are necessary.

Active synchronization is quite easy to perform using an atomic counter. Each job that needs to be synchronized can be launched through a synchronization object, which internally holds this counter. It will be incremented for each job scheduled and decremented by the job itself when it terminates. At any time, a PPU thread can read the counter to calculate how many jobs are still executing or waiting for execution. [See code sample 9.]

Passive synchronization is a bit harder. In addition to the counter, we add a pointer to a SPURS event flag. When a job decrements the counter to 0, it will set the event, waking up any

PPU thread sleeping on it. But this method has associated multithreading issues: Manipulating the counter and the event flag is not done atomically. A lot could happen between the counter update and the event flag update. To make sure that the event flag stays in a correct state, we absolutely need to reset it on the PPU if it has been set by an SPU. But the counter could reach “0” many times while scheduling a batch of jobs. For example, consider the case where jobs terminate faster than we are scheduling them. We need a way to enforce the following pattern: An SPU that decrements the counter to 0 will set the event flag only if a wait operation is in progress on the PPU. That way, we will ensure that the corresponding reset will be performed on the event flag from the PPU. Packing a flag with the counter in a single 64-bit field is a solution. Each time a PPU thread performs a wait it sets the flag, and then waits on the event. The last SPU job decrementing the counter to 0 will see if the flag is set or not, and will only set the SPURS event flag if necessary. In this case a reset needs to be performed by the waiting PPU thread, since it has raised the waiting flag. [See code sample 10.]

### CODE TILL YOU SPU

» The techniques described in this article are just the foundation upon which we can build a complete job scheduler. The FAR CRY 3 scheduling system lets the user create a complete dependencies chain, mixing SPU jobs, PPU tasks, and an RSX command buffer, while providing APIs to synchronize to any step of the scheduled execution. One great advantage of the system is its ability to freely mix up jobs with different affinities and priorities, which is impossible to do with a single job chain. By bringing a layer of abstraction between the actual job and the SPURS workload, we can freely mix in some properties that are usually bound to the SPURS workload to a new type of workload.

Finally, I could not end this description without providing a few figures. Scheduling a job takes about 2,000 cycles, and the job typically starts its execution in 10us. Waiting PPU threads (using passive synchronization) are usually woken within 50us. Since job chains are shared, we use only four of them. All of the features only cost an additional 7KB of SPU code in each job, and take around 2us for execution. Overall, all these systems have allowed us to hit a mean 80% of active SPU usage, while reducing PPU latencies and wait times to zero. 🎮

The author would like to thank Jeremy Moore for his assistance in editing this article.

**REMI QUENIN** is the technical architect of FAR CRY 3 at Ubisoft Montreal. He acts today as a multithreading and platform expert in the studio, and has 10 years experience in the game industry. Remi can be reached at [remi.quenin@ubisoft.com](mailto:remi.quenin@ubisoft.com).



# LEARN TO CREATE GAMES WITH UNITY GAME ENGINE

Get Unity 3D Online Training with  
**UNITY WORKSHOP**

**UNITY COURSES**

**01 - UNITY FUNDAMENTALS**  
Beginner course for developers, educators, and students that are new to Unity. The learner is introduced to the Unity workspace and typical project workflow.

**02 - GAME ENVIRONMENTS**  
Intermediate course using a guided digital studio approach to introduce students to techniques for creating art for game environments using Unity.

- 100% Online
- Project-Based Courses
- Digital Video Lessons
- Live Expert Help

**unityworkshop** WWW.UNITYWORKSHOP.COM • 1.877.895.6882

Br brought to you by Serious Game University • www.seriousgameworkshop.com



# MAKE MORE ENEMIES

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

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

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

**VFS** 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: THE THIRD

VFS student work by Brendan Boyd





# THE HAPPY COG

## WHERE DO YOU FALL ON THE SPECIALIST VERSUS GENERALIST DEBATE?

If you're a dinosaur fan—and the grand traditions of game industry nerddom suggest you probably are—you may have seen those diagrams that show how the first lungfish that made their way out of the Devonian ooze gave rise to family upon family of ever more exotic and specialized creatures. The proper name for this kind of family tree is cladogram. If you're interested in the evolution of the games business—and again, you should be—it might serve you to reflect on our own cladogram.

In the 1980s, there were only a handful of jobs. Programmers wrote code, designers designed games, and artists made bitmaps. As time rolled on, however, each job became more and more specialized. Nowadays, it seems like everybody has a compound title: technical animator, cinematic camera artist, or environmental effect specialist. Big AAA teams have people whose only job might be placing lights, adding secondary animation to hair, or optimizing shaders for a particular platform. It's easy to spend the bulk of your professional life in a subspecialty of a specialist role.

A lot of us find the complexity of the modern cladogram a bit disheartening. Like the dinosaurs, in larger teams, we've evolved to fill a variety of exotic niches—and sometimes have adapted ourselves into a corner. Explaining to friends and family what, exactly, we do verges on the impossible. Mentally toting up the value of your individual contributions to a finished game is often humbling. Parceling out the work so finely is not very gratifying to the artistic ego, and (as many big team veterans will privately admit) it doesn't seem very hospitable to great art, either. To top it all off, many specialist jobs are just boring—just ask anyone who's spent the last year applying custom UVs for waterfalls, say, or tweaking constraints on physics props, or cleaning up mocap five days a week.

### THE PIN FACTORY

» The discontents of hyperspecialization aren't unique to artists. Technical pigeonholing and emotional disconnection from work were already old news when Charlie Chaplin made *Modern Times* in the thirties. Since everybody loves to grouse about it, it's worth pondering why this kind of organization is so common. It must be good for something... but what?

The classical explanation for our troubles goes back to the works of Adam Smith, the enlightenment philosopher and granddaddy of modern economics. Back in the 1770s, Smith pointed out that traditional blacksmiths took a long time to make pins. The task, he said, required 18 distinct steps, and a solitary pin maker could barely be expected to make more than one pin per day. However, by breaking up the steps into distinct jobs and assigning specialists to them, productivity was increased enormously: a small factory with 10 people could crank out thousands of pins per day. This specialization, which economists call the division of labor, was one of the cornerstones of the Industrial Revolution and, with it, modern life.

Game studio executives aren't usually experts on 18th-century social theory, but most of them embrace the division of labor even if they don't use the term. It's the standard way things get done in the modern world. Specialized work, broken down into clearly defined tasks and processes, is efficient. It allows workers to concentrate on a small number of key skills, rather than mastering many. Specialized workers don't have to constantly pause to reacclimate themselves between jobs, which most of us have been through when revisiting an obscure corner of Max or Maya for the first time in months. Specialization puts work into the hands of people who know the details intimately. Responsibilities are clear, so it's easy to see who's doing what (and how well they're doing it).

From a manager's perspective, the division of labor has another key advantage: it makes managing a lot easier. Individual people, with all their strengths and weaknesses, are tricky to manage and use effectively. Narrowly-defined roles, on the other hand, are easier to fill and to manage. For example, deciding whether animator X is up to the task of rigging the new alien octopus-thingy is a tough task for a busy lead. It's hard to know what's necessary and whether X has the technical skills. Perhaps animator Y has a better handle on the expressions and constraints side, but isn't as talented a keyframer; or maybe X doesn't like to work with other people's rigs and insists on going it alone. When these kinds of considerations start to pile up, many managers can conclude that passing the task off to a dedicated character rigger seems like the better option.

Finally, narrowly-defined roles are easier to fill. It's a lot easier to specify your needs in terms of widgets rather than fully-rounded human beings. "We need two more lighting artists to make sure we can get the E3 demo finished" is easy. "Our new hires have to find the right balance between ambient and direct lighting, have a good eye for shadow composition, and know how to make good compromises between performance and aesthetics" is hard. You can see this principle at work in the job postings on Gamasutra. Look at the difference in verbiage between the specialist hires and the ads from smaller teams that need all-rounders. Specialists can be easier to find (just do a keyword search on LinkedIn!) and less risky to hire than good generalists, for the same reason it's easy and cheap to find a plain-vanilla cell phone than it is to pick the perfect smartphone with dozens of features.

### LEVEL 10 DRUDGE

» This all adds up to a pretty straightforward observation: Over time, teams tend to get more specialized and jobs become narrower. For those of us actually doing the work, this isn't always pleasant or fulfilling. Adam Smith saw that one, too. He wrote in *The Wealth of Nations*:

"The man whose whole life is spent in performing a few simple operations ... has no occasion to exert his understanding or to exercise his invention in finding out expedients for removing difficulties which never occur. He naturally loses, therefore, the habit of such exertion, and generally becomes as stupid and ignorant as it is possible for a human creature to become."

Things aren't quite as dire for us—most of the time, anyway. Even so, the division of labor, for all its benefits, can result in jobs that are economically efficient and completely humdrum at the same time. Moreover, there are times when the needs of the company and the needs of your own career don't align very well. The company may be happy with a reliable, competent button-pusher cranking out content, but even if you can handle the boredom, you don't want to become so dependent on a narrow job niche that you wind up obsolete. For your own protection, decide how you want to fit into a world in which the division of labor and all of its accompanying irritations is a fact of life.

First and foremost, you need to figure out your own tolerance for a narrowly defined role. Some people are natural specialists. If you enjoy the pleasures of really mastering a tightly-defined job, you can embrace the division of labor. Having expertise in a marketable specialty can be very rewarding both personally and financially. Specialist jobs tend to be more technically defined, which can make the job hunt less stressful—though art hiring always involves subjective tastes and preferences. The range of opinions on, say, good ragdoll setups or a well-tuned particle system is a lot narrower than the range of opinions about a character design or a cinematic scene. Last but not least, specialist jobs tend to generate communities of like-minded artists from across the business, which allows specialists access to the all-important grapevine where jobs are informally aired long before they show up on Gamasutra.

If you're a specialist, grooming your specialist credentials is a key part of your career. Don't leave your professional profile up to chance (or to your willingness to do the tough jobs other people on the team avoid!). Embrace your identity as an expert. Get involved with online communities that cater to your trade, from Animation Mentor to techartists.org. Even if you're not much of a networker, you can pick up invaluable comparative knowledge about how your job is done on other teams and with other tools. A solid strategic understanding of your specialty (not just your current job!) is the defining difference between being an expert and a low-skilled software jockey. Without it, you risk becoming dangerously dependent on one workflow or one company's ways of doing things.

Staying in touch with fellow specialists across the industry is also critical for navigating the current and future market for your skills. MEL scripters, for example, had better keep an eye on the demand for Python; level designers who specialize in Hammer need to know how many companies are using the Source engine; and so on. Examples can be multiplied, but the principle is always the same: don't let yourself become so specialized that you can prosper in only one environment. When you embrace the division of labor, you are exposed to the ongoing churn of technical and business change. Plan accordingly.

On the other end of the spectrum, of course, there are many artists who actively resist the idea of focusing on a narrow niche. Some want the kind of creative control that comes with owning something from start to finish. Some get bored without a steady supply of new and novel challenges.

If you're one of these committed generalists, you need to recognize how this influences your job prospects. Thriving as a generalist is hard work in large teams, since it requires keeping up with more than one set of skills. Most successful switch-hitters maintain a steady stream of nights-and-weekends solo projects, since the workplace often subtly—or not so subtly—tries to push them into narrower roles. Above all, though, generalists have to keep pushing their visual art skills. Artists with a technical focus can get by on their expert knowledge and tech savvy, but successful generalists need to let their portfolios do the talking. It's tough to compete with dedicated specialists across the board: Being able to sculpt and animate is impressive, but you may be competing with talented sculptors and practiced animators. Pick projects and subjects that showcase your versatility as well as your skills.

A broad skillset steers you toward different kinds of companies. Big teams hire more specialists, partly because they have deeper pockets and partly for all the organizational reasons we outlined earlier. Smaller companies are more hospitable to generalists, since they can't afford to maintain a stable of experts for every contingency. Indie, mobile, and web development have fostered a mini-renaissance for disaffected artists tired of toiling anonymously in the AAA salt mines. There are some options



Quetzalcoatlus was one of the most highly specialized dinosaurs ever—paleontologists believe it was evolved to prey only on the eggs of large dinosaurs. Of course, its success was dependent on the presence of a lot of large dinosaur eggs, so it could not survive on its own.

for generalists at bigger companies as well, but they won't come on the generic production line. Prototyping groups, firefighting teams that provide emergency support to projects on a deadline, and in-house training are all roles where generalists thrive in big organizations.

There's no "correct" balance between the need for autonomy on the one hand and the pleasures of mastering a craft on the other. Besides personal inclinations, the right answer can also change with circumstances. The stimulation and excitement of small-team life may also seem like perpetual insecurity to a new parent. The fun of diving deep into a new specialty may run out when there's not much left to learn and the future starts to look like an endless repeat of the past. Even the most laser-focused specialists need to respond to the market for what they do. Even the most agile generalists can't do everything equally well.

The unchanging constant is the need to think hard about what you want to achieve in your career. Know what kind of work makes you happiest and most productive, and know what you're good at and where your weaknesses are. Keep an eye on the market for your strengths. But above all, don't leave it up to chance; the inner logic of division of labor wants to turn you into an anonymous, interchangeable, and ultimately disposable widget on an assembly line. If you don't make some deliberate decisions for yourself, the system will make them for you. 🗣️

---

**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.



# COPYFIGHT

## AN INVESTIGATION INTO WHETHER COPYRIGHT LAW IS STILL SERVING ITS INITIAL PURPOSE

The start of 2012 saw a lot of news around the issue of intellectual property and copyright. At the forefront was the protest over SOPA and PIPA, two pieces of proposed legislation that had heavy backing and support from entertainment industry associations like the RIAA and MPAA, and of course, the game industry's ESA.

In late January, SOPA and PIPA seemingly became "dead letter" legislation as political support for them waned, following protests that included participation from online heavyweights such as Google and Wikipedia. This, together with the collective protest of thousands of citizens, gave many of the politicians supporting these bills pause.

While these actions were laudable, it's worth pointing out that the protests were largely taking issue with implications of the proposed bills' heavy-handed methods for enforcement of copyright violations. There was very little discussion about whether the copyrights they aimed to protect were effectively serving their purpose in the first place.

### PURPOSE OF COPYRIGHT

» A commonly held misconception is that copyright's sole purpose is to protect the creator of original works. This is one purpose, but copyrights, like patents, seek to balance the benefit to creators given by a limited-time monopoly, against the good done for society as a whole when those works are created. The limited monopoly is granted, as stated in the U.S. Constitution, in order to "promote the Progress of Science and useful Arts," and should go as far as is necessary to serve that purpose.

The history of copyright dates back to 17th-century England, and protections that were afforded book authors and printers. In the 18th century, the United States established copyright protection. In both cases, limited periods of monopoly were granted.

### COPYRIGHT HAS LOST ITS WAY

» Copyright, especially in the United States, has tipped heavily in favor of the rights-holders, and away from the societal good that it's meant to serve in the first place.

The laws have become distorted by what William Patry, in

his book *How to Fix Copyright*, calls the "piracy-industrial complex." Drawing parallels with Eisenhower's military-industry complex, he describes the copyright industry's need to further the justification of its own existence through an ever-growing specter of pirates.

By inciting a moral panic about entertainment industry jobs lost due to theft of copyrighted materials, these industries are able to lobby for further control over their markets and for public funds to go toward their enforcement. However, the data proving impact of copyright violation, especially in terms of jobs lost, is tenuous at best.

The time has come to admit that the current system of copyright isn't serving society in the way it is supposed to. There are several issues with the way copyright stands today:

**Terms are too long:** Copyright in the USA today is granted for a period of 120 years or 70 years from the death of the author. It is absurd when keeping in mind the goal, which is to incite the creation of more original works. It's hard to imagine that a would-be auteur somewhere is refusing to create because his work may fall to the public domain many decades after his demise. Worst still, as terms have expanded, they've done so retroactively, as though they could somehow further encourage creations from authors long-since dead. As Patry put it in his earlier *Moral Panics and the Copyright Wars*, these authors aren't composing, they are decomposing.

**One size fits all:** Further aggravating the issue with lengthy terms is the fact that different media may have different timescales over which the majority of their investments are recouped, and after which they become obsolete. This is especially true with a technology-related medium like video games. For example, lengthy copyright terms interfere with the ability to make copies of decades-old games for purposes of historical preservation, which is something that will become even more problematic as DRM-protected works fade into history.

**Benefit to rights-holders, not creators:** Because copyright often ends up in the hands of publishers, distributors, or other gatekeepers, it doesn't always benefit creators. Certainly, creators are usually compensated in exchange for those rights, but it leads to situations where, for example, works unpublished for lack of commercial interest may sit dormant, with the author unable to access them for self-publishing.

**Not backed by objective data:** All of the copyright legislation enacted in recent history has leaned heavily on moral panics and industry-influenced data. There are many cases where objective assessments were done and ignored because they didn't back up the copyright industry's position. [Two examples of such data are the Rappaport report on Copyright Term Extension: Estimating Economic Value reported to US Congress in 1998, and the Gowers Review of Intellectual Property done in the UK

in 2006, both of which concluded that term extension does not result in the creation of new works. In both cases the data was largely ignored.]

**Abuse of the system:** With the advent of the Digital Millennium Copyright Act, a safe-harbor provision was provided to protect service providers from liability provided they responded quickly to notifications of allegedly infringing material. This was certainly a useful exercise that allowed services such as content portals or app stores to function. However, since the easiest path to take in reacting to such requests is to just remove the material, that is sometimes the path taken. Getting the content reinstated can often be an uphill climb. This provides an incentive to abuse the system, but with no likely repercussion for doing so.

For example, after Atari issued takedown requests on a number of indie titles resembling some of their retro arcade hits in late 2011, there were accusations that Atari's relationship with Apple allowed the titles to be taken down without opportunity for rebuttal or independent evaluation. Whether Atari was in the right is not the issue. The issue is that actions were taken without transparency and opportunity for recourse. Without transparency and opportunity for recourse, especially for independent developers without means to seek legal help.

### WHY SHOULD YOU CARE?

» Creators and innovators in all creative arts, including video games, should benefit from the

investments they make and risks they take. Copyright should serve that end. However, it's clear that the current system of copyright has been distorted to serve the creators less than the gatekeepers to which they end up beholden.

SOPA, PIPA, ACTA, and legislation that will undoubtedly follow them risk giving additional control to gatekeepers, while potentially reducing the visibility you might have into processes by which your games are assessed against any copyright violation claims. In addition, they might place additional burdens upon your business if it in any way serves as a host of content (user-created content, for example).

#### WHAT CAN YOU DO?

» People are busy, and game developers especially so. Still, there are some relatively low-bandwidth ways you can contribute toward

improving the future of copyright.

1. Get informed: There are a few online resources worth spending some time reading. The Wikipedia entries on SOPA, PIPA, and ACTA are a start. The Khan Academy has an excellent briefing available on YouTube. Those looking for a meatier background on the subject would do well to read William Patry's *How to Fix Copyright*. Another good book recently published is *Uncensored* by Hunter Walk and Eric Ries.

2. Exercise your rights as a citizen: Write your representatives to express your opinion on the subject, especially when any of these bills come up for a vote. SOPA and PIPA have been shut down for the time being, but they'll surely emerge in another form. The threat of repercussions in the form of votes lost has shown to be a remarkable force against the corrupting influence of lobby dollars.


3. Start the conversation on

copyright reform: We need to reboot copyright. Copyright needs to serve the purposes for both creators and society for which it is crafted. This starts with objective, evidence-based conversations about policy.

4. Support organizations doing the freedom fighting: Support organizations like the Electronic Frontier Foundation and Creative Commons, even if it's only a fraction of what you spend on content bought through gatekeepers.

5. Blow the whistle on abusers. We are fortunate enough to work in an industry where many of the consumers of games care about the people that create them. For this reason, the gatekeepers fear, more than they care to admit, the possibility of negative publicity that comes with abuse of power. When such abuses do happen, help shine the light on them.

There are those that advocate for the outright abolition of

copyright. I'm for reform, not abolition. Copyright serves a purpose. As this goes to print, my friends at Spry Fox have filed a copyright suit against 6waves-Lolapps for what appears to be a legitimate grievance. This is exactly the purpose copyright law is supposed to serve, and the method by which it was intended to be resolved. If an independent two-man game developer can seek justice through the legal system, is it not reasonable to expect that large media corporations might do the same, rather than doing it at the taxpayers' expense? 

*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.*



the art & business of making games

take control of your future



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





# RANDOM IS RANDOM

## A DESIGNER'S LOVE/HATE RELATIONSHIP WITH THE DICE ROLL

**A simple die roll can be one of the single most vexing tools in the designer's repertoire.** Rolling dice is fun. Landing double-sixes is a lot of fun. Seeing big critical hit numbers pop across a big bad boss monster's head is deeply satisfying. And yet, the experienced designer knows that—much of the time—the dice are his enemy. This is tragic, of course. Most game designers, at the core, love absurd amounts of dice rolling.

There are few ways to get a room full of RPGers more excited than to bring back the fond memories of the Rolemaster Critical Hit Tables, where simple acts like trying to climb a ladder might result in accidentally dropping your sword and removing your own arm. The sheer magic and hilarity of events like this is transcendental. They're stories that will be described to other gamers for years. However, passing that point and actually playing further as a one-armed Paladin is often only fun in retrospect.

### LUCK VS SKILL

» Randomness is considered a cornerstone of games. In fact, few items better symbolize games than dice, the history of which goes back at least 5,000 years, to an ancient Backgammon set found in the Burnt City excavation site in Iran. But even the ancients could get past randomness. Chess, a game that emerged from the sixth century as perhaps the finest game design of all time, has no randomness whatsoever beyond deciding which player should go first.

In the absence of randomness, games are entirely about skill, which means that the debate between skill and luck goes back at least 1,500 years. That game design distinction also happens to be one of the few that actually have a legal definition, as well—whether your game is considered to be a game of skill or a game of chance determines whether players can bet money on it on a Vegas gaming floor.

A game without a significant luck element is almost entirely skill-based—and most of the video games played professionally, like STARCRAFT and TEAM FORTRESS II, have little or no luck component. Both hardcore electronic and strategy board gamers tend to eschew too much luck in their games. Two board game classics, Diplomacy and Puerto Rico, have extremely limited luck components.

What's not obvious is that the converse is not necessarily



true: Games with heavy chance elements may also have strong biases toward skill, provided that skill can be used to make sense out of the chaos being thrust upon the game. Magic: the Gathering has a huge luck component to it, primarily based on the cards you draw, and this random component allows new players to have a fighting chance

against more experienced players. That said, pro Magic players go to great lengths to assemble decks designed to reduce this element of chance as much as is humanly possible.

### RANDOM IN SOLO PLAY

» In some games, randomness is a necessary design element.

In particular, many single-player games are particularly hard to design without some random element to it, whether it's the AI, or the challenges faced within. One of the most common examples is randomizing the initial board state. This is done in games as casual as Solitaire and MINESWEEPER, or as hardcore as CIVILIZATION. The randomness is a huge part of why these games can be replayed hundreds or thousands of times (and conversely, many modern games that abandon randomness for tightly scripted experiences have little or no replayability at all).

But this randomness also can present the player with wildly variable problem sets. CIVILIZATION on the hardest difficulty is (at least for me) pretty much impossible without a favorable initial world set up. Conversely, a straight game of Klondike solitaire has a pretty good chance of being completely unwinnable from the moment cards are dealt. Still, there are opportunities for the enterprising designer—there are versions of both Solitaire and Mah Jongg available online that promise to present a winnable problem to the player.

The psychology of a known solvable game of solitaire is, as a designer, interesting to contemplate. Once you know the game is solvable, is the player more likely to replay a vexing board? Does the game feel more like a true skill-based game at this point? Or does removing most of the

unwinnable boards make the game, on average, trivial to beat?

### THE DOWNSIDE OF RANDOM

» Imagine if you swung your sword and did 10 points of damage with every swing, every second. This would very quickly start to get stale and monotonous, so game designers find ways to add variance to the system—making the damage range from 5 to 15 points instead, or adding in random hits and random misses, occasional chances for critical hits.

One streak of bad luck can make this all fall apart. If random rolls are truly random, it's entirely possible that our hypothetical swordsman might hit for 5 or 6 point every time—hypothetically halving his DPS in this example. Even worse, if his opponent is having the opposite luck, she might have triple his DPS, even if they both have the same sword and skill. Furthermore, on the next fight they might have exactly the opposite results. The end experience would feel—well, random. So much so that it would feel like player skill had no role in the fight whatsoever.

Streaky behavior constantly has to be controlled for in games. In MMOs, it's somewhat common to give a raid monster an enhanced critical attack that does so much damage (or has some other debilitating effect) that the tank has to do extra trickery, or the healing brigade has to drop non-stop healing in a never-ending chain. These powers that are extremely devastating if they happen once become unsurvivable if the boss happens to land 2 or 3 of these in a roll. Designers then have to go in and put in cooldowns, or otherwise manage the randomness to keep the fight survivable.

### RANDOM IS RANDOM

» One casual takeaway is that randomness is more fun when it breaks in the player's favor. Critical hits are fun—and streaks of them are giddy fun—but rolling misses is not. This is a useful insight, but still an oversimplification—especially as these reward events get rarer.

As an example, it's decidedly hard to make loot both very rare,

and reliably drop. Take the odds of dropping a Phat Purple from 1 to 1,000 to 1 in 10,000, and you've increased the odds that some player will get unlucky and never see the item. On the flip side, a one in a million chance of hitting the jackpot ceases to be rare when you have more than a million players pulling the lever more than a billion times. The jackpot is going to drop, and some lucky players are going to get multiple bonuses. Players will see the streaks, and perceive that your random number generator is broken, even when it's working as designed.

On top of it all, in a true random system—each player's rolls are truly independent. If I've flipped a coin 19 times and gotten heads every time, the odds that the 20th will also be a head is still exactly fifty-fifty, even if players might feel differently. Similarly, without artificially mucking with things under the covers, there is no guarantee that multiple players playing the game won't get lucky and strike it rich, either flooding the economy or bankrupting the house as a result. There's also a chance that the event will fire so rarely that its existence is a mere myth to the playerbase, which may or may not be a bad thing.

### WHEN RANDOM ISN'T

» Is true randomness a good idea at all? Designers may find that true randomness is not well understood or is often misinterpreted. True randomness can, for example, give you the same loot 13 times on 15 boss attempts—it's unlikely, but entirely possible. And while designers love to roll dice, it's more important to provide something resembling a predictable player experience.

This is actually trickier than it seems. The human brain is designed to seek patterns, and instill order, logic, or superstition on what it sees. Need for greed rolls in MMOs perfectly show the failures in psychology here: The player won't notice a full night of loot drops being distributed evenly, but will glom onto the fact that his party mate managed to win 3 things in a row. The player will utterly discount when he's rolling well if his party

mates are rolling better than him. And the average player is only subtly aware that larger party sizes decrease the odds that he'll win any given roll.

Still, there are ways to give the illusion of randomness, while still finding ways to keep the math under control. One option is to give the loser bonuses that improve their odds of winning the next roll, which evaporates when you finally land the roll. This not only potentially solves the problem, but also rewards perennial losers with some fun, ludicrously large rolls to brag about.

Another way to approach the problem is by removing successful rolls from the probability space. Imagine that your 1d100 die roll is represented by a randomly shuffled deck of cards with those numbers on them. When you choose a number, remove that card from the deck. Once the last number has been chosen, all cards are shuffled back together and are selectable again. This approach has several benefits to it, the least of which being that a 50% chance of success is, over time, pretty much guaranteed to be 50%, and win streaks will almost certainly be counterbalanced over time.

There are pitfalls. Mechanics such as these can be leveraged and potentially exploited by a savvy player. An experienced card counter knows to adjust his game once he knows the blackjack deck is disproportionately full of face cards—at that point, the deck is no longer truly random. This may not be something that has to be stopped, and may actually be a design opportunity. Strategy in deckbuilding games like *Magic the Gathering* and  *Dominion* heavily leverage awareness of what remains in your deck, and the designers of both games have learned to embrace this and give players tools with which to further finesse these possibility spaces. It turns out that if random is fun, cheating that randomness is even more fun.

### RANDOM CONCLUSION

» Five card draw is a game that is, at its core, a pretty random

experience. The odds that you will get a good hand are pretty low, and whether your hand is better than your neighbor's is pretty much a total crapshoot. The only cards that are excluded from the possibility space are the cards that you've drawn. The odds of two players having great hands (i.e. a full house beating a straight flush) are pretty low. The game is, effectively, entirely about bluffing.

Texas Hold'em is a lot less random. Each player's random elements are reduced to two cards, plus the undrawn cards in the river. Because of that pool of shared cards, there is a high correlation between the quality of players' hands; if you are close to a flush, the cards in the river mean that other players are also within sniffing distance. Conversely, if there are three 8s in the river and you have the fourth, the possibility space of how your opponents can compete with you shrinks to nearly nothing. The game still has strong random elements, but these random elements can now be managed and strategized. This design element is a key reason why *Poker* recently took the world by storm.

So is randomness good or bad? The answer is that, like many things, randomness is an element of a designer's toolbox that can be used for good or evil, and can dramatically change the tenor of the game. Some poker players lament how Texas Hold'em puts a premium on math and probabilities over the skill of bluffing. Conversely, some strategy gamers refuse to play any game with more than a negligible amount of randomness. The answer will be different for each game. It's up to the designer to figure out how to wrangle that randomness in a way that he can guarantee an interesting and challenging experience to the player. 🎲

---

**DAMIAN SCHUBERT** is the lead systems designer of *STAR WARS: THE OLD REPUBLIC* at BioWare Austin. He has spent nearly a decade working on the design of games, with experience on *MERIDIAN59* and *SHADOWBANE* as well as other virtual worlds. Damian also is responsible for *Zen of Design*, a blog devoted to game design issues. Email him at [dschubert@gdmag.com](mailto:dschubert@gdmag.com).

# THUMBS UP TO THE WINNERS

GameSpot gives props to the creative minds recognized at the Independent Games Festival. Without talent like you, the world of gaming just wouldn't be the same. Check out our exclusive coverage of the IGF Awards and GDC at [gdc.gamespot.com](http://gdc.gamespot.com).

**GAMESPOT**

FUEL FOR PLAYERS





# IT'S ABOUT PEOPLE

## THE GREAT GAME AUDIO COMMUNITY

It's amazing. People are amazing. It always comes right back to that for me ... I love a challenge as much as the next person, but when it's all said and done, my biggest takeaways from any project are the relationships I formed and experiences I had working with others. The ability to collaborate and create with people means that the accomplishments will be greater than when setting out alone. When creative thinkers and resolute problem solvers apply themselves, there "ain't no mountain high enough" and no task too formidable. This is what leads me to the Game Developers Conference every year—meeting up with old friends, and engaging new folks in the game audio conversation. That's where some of these discussions start, but it doesn't have to be the end. Read on for ways to keep the conversation going throughout the year.

### LEVELING

» In what has become a yearly pilgrimage, the San Francisco GDC brings together a community of people centered around the pursuit of sharing and gaining knowledge that might otherwise be lost in a sea of search results. The singular magic that happens when like-minded people get together and rally around a single pursuit cannot be contained within the confines of sessions and exhibit halls. Even the conversations in the hallways comprise some of the finest and most relevant thinking between both professionals and hopefuls.

Whether you're just starting in games or running in the same circles year after year, it's important to know where you stand in relation to others. Taking part in the largest conglomeration of Bes and Wannabes in the industry is a fantastic way to see the levels of skill, dedication, and talent that exist, and helps to frame your passion. People who are working (and want to be working) in games are only a handshake away. Most are hungry to be either recognized or validated for their hard work, and taking the time to get to know someone is what it's all about.

### CONNECTED

» It should come as no surprise that people in the game industry are well connected. Through years of company jumping, project shipping, and studio closings there is now a web of relationships in the industry that lies like a blanket over the conference. At times it can feel a

bit like a class reunion, and that is definitely one of the great things about coming back year after year. There is a danger in that familiarity, though, if it keeps people from reaching out to newcomers. Part of our responsibility as professionals in the community is to be part of a supportive network for people who are interested in learning the art



of our craft. While sessions often provide a one-sided conversation between the presenter and the audience, it's the interaction outside the sessions that allows for sublime moments of inspiration.

There comes a time during those conversations when it's no longer cool to keep talking shop; I'm the worst at this. I'm always the last one to let go of any conversation about game audio. I can happily ramble on endlessly about freakish details and solicit strangers for their perspectives on esoteric audio intricacies for days on end ... that is, until I realize that there is more to life than cool tricks and industry

secrets. It's amazing what you can find out about a person if you ask. I find that it's easy to forget simple conversational technique while steeping in the geekery. Balancing the fine art of conversation with the intense desire to learn can be a struggle when you're surrounded by so many intelligent people. Don't forget to be human.

### ONLINE

» When the conference is over, the ecstasy of communication doesn't have to be! Here are a few other places where you can engage the game audio community:

IASIG, G.A.N.G., and the IGDA Audio SIG continue to be communities dedicated to furthering the art of interactive sound and music. Through initiatives, conversations, working groups, and in the forums, these entities give you a place to exchange ideas. Becoming a member gives you the opportunity to volunteer your time and energy to accomplish something greater than you could on your own—instead of just sitting on the sidelines.

A looser-knit band of sonic surgeons can be found over at GameAudioForum.com, where discourse has been ongoing since 2006. I've seen this community, which has a steady mix of professionals and hopefuls, provide intelligent perspectives and supportive feedback, since its inception. Whether wrestling with implementation, showing off accomplishments, or musing on the state of the industry, its open door policy allows for valuable

insight from a caring and varied community.

For those of you who have discovered the singular oddity that is Twitter, it should come as no surprise that the conversation over there never stops. Whether you see it as an interwoven fabric of friendships, cult of personality, or something else entirely, the game audio community is well represented. Just search for the #GameAudio hashtag to find a steady stream of game-audio related articles, video, and conversation topics that routinely spark a lively debate.

### OFFLINE

» Don't think I'm advocating for a Neanderthal's life, all blinking pixels and wide eyes on the wild frontier. There's plenty of community outside the box if you know where to look. While game audio may be in short supply where you live, there's a good chance that a pocket of independent game development is just around the corner. The IGDA has chapters all over the world that exist "to advance the careers and enhance the lives of game developers by connecting members with their peers, promoting professional development, and advocating on issues that affect the developer community." I've always seen my local chapter meeting as a melting pot of enthusiasts coming together to celebrate games and discuss the hard work that goes into making great ones ... which sounds strangely familiar. 🎧

**"Big City, Bright Lights, Cool Cool People, Big City. Everybody I know can be found here."**

—Spacemen 3

DAMIAN KASTBAUER is a technical sound design outlaw and confused chocolatier who can be found musing on game audio at [LastChocolateLab.com](http://LastChocolateLab.com) and on Twitter @LastLab.



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



# GDC Vault

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

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



# A Hayward Soul

## CODIES QA PROFESSIONAL MAKES HIS WAY TO DESIGN

*Darren Hayward worked in Codemasters' QA department for many years, but dreamed of doing design. That path is less clear than it once was, and he found himself applying outside of as well as within the company. Ultimately, he landed a design job in Codemasters' Racing Studio, and we quizzed him on the transition.*



**Brandon Sheffield:** How difficult was it for you to bust out of QA? Was that indeed your intention?

**Darren Hayward:** I'd been trying to land a development job for 12 months or so before I got a position with the design team in the Racing Studio. It feels like it's getting harder and harder to break into design, with a lot of experienced designers in the industry already, and a constant trickle of talented graduates looking to take up entry-level positions. It's difficult to communicate to employers why they should choose you over the mountain of other candidates that will inevitably apply for a design role. A solid, varied portfolio is a good start, and it can't hurt to pick up QA experience while you wait for an opportunity.

**BS:** Do you feel the job situation is particularly difficult in the U.K., with many studios shuttering and many talented people looking for work?

**DH:** It is, and it looks like it's only going to get more difficult. There aren't many big studios left in the U.K., and as you say we've lost a few in recent years. There are new start-ups, but it takes time to build job opportunities. It's easy to see why so many people are emigrating to places like Canada, especially

with the tax breaks and better wages out there.

**BS:** Did you apply externally as well as internally?

**DH:** It was getting to the point where I didn't think I'd get the opportunity at Codies, so I was looking elsewhere. It's not easy to make the transition from QA to design, so I was keeping my options open. There are a lot of developers around here, and I got a great offer from another company—but once I was almost out of the door, I was offered a spot on the racing team. I couldn't turn down the opportunity to contribute to the design of AAA games at a company I'd had a blast at for nearly 5 years, so I ended up staying. I'm happy to say I'm really enjoying myself so far!

**BS:** How have you found the transition? Do you find yourself thinking about design from a test-driven perspective?

**DH:** Always. How will QA try to break our new mode? Does this message meet platform-holder guidelines? You've got to think about these sorts of questions for all your features. If it breaks in-house, it'll break out in the wild for sure. That's not to say we don't try to innovate and bend the rules as much as we can, but from a production point of view if a feature gets to QA and it's not going to work, it's expensive to tear it up and start again.

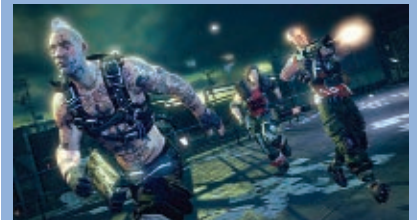
**BS:** What, to you, makes a good test environment?

**DH:** Communication with the development team is absolutely critical. I've seen testers waste days working tirelessly on buggy areas of a game, only to be told that fixes were already on their way in a new build. Luckily we've got development QA working in the studio here, as well as a full team of experienced testers on the campus in another building. This means we can keep everyone in the loop about what should work and when, as well as allowing for a speedy build delivery process. **GD**

# who went where

After one year of leading Disney Interactive's games business, Bungie and Wideload cofounder Alex Seropian has left the company altogether.

Andrew Graham, one of the key designers and programmers for the Codemasters-developed MICRO MACHINES series, has joined a handful of other Codemasters veterans at U.K. mobile studio Kwalee.



BRINK developer Splash Damage has hired veteran programmer Marc Fascia, who previously worked at Ninja Theory, as its new technical director. The company has also hired the FIFA series' former senior development director Griff Jenkins as its new studio production director.

Susan Panico, senior director of the PlayStation Network in North America, has left Sony Computer Entertainment after more than 17 years with the company.

# new studios

Eat Sleep Play cofounder David Jaffe has left the company to found a new developer in San Diego.

Innovative Leisure is a new L.A.-based video game company led by former Xbox man Seamus Blackley, which combines the talents of 11 Atari development superstars who will make games for the iPad under publisher THQ.

Yasuhiro Wada, formerly a producer at Marvelous Entertainment and the creator of long-running farming game series HARVEST MOON, has formed a new company called Toybox in Japan.



# GAME DEVELOPER MAGAZINE

the best of  
postmortems,  
product reviews,  
and standout  
columns

NOW AVAILABLE FOR  
DIGITAL DOWNLOAD AND  
FOR iOS DEVICES.

**SUBSCRIBE TODAY!**

[GDMAG.COM/SUBSCRIBE](http://GDMAG.COM/SUBSCRIBE)

DOWNLOAD THE GAME  
DEVELOPER APP  
[bit.ly/gdmag\\_ios](http://bit.ly/gdmag_ios)



# gd



FOLLOW GAME DEVELOPER

## GDC AWARDS HONOR MISSILE COMMAND CREATOR, FIRST AMENDMENT GAME LAWYERS, WARREN SPECTOR

The 12th Annual Game Developers Choice Awards have revealed the recipients of three of its Special Awards—the Pioneer Award, given to developers for creating breakthrough video game genres or concepts, the Ambassador Award, given to those who have helped the game industry advance to a better place, and the Lifetime Achievement Award.

### PIONEER

» The Pioneer Award celebrates individuals responsible for developing a breakthrough technology, game concept, or gameplay design at a crucial juncture in video game history, paving the way for the many developers who followed them.

This year's honoree, Dave Theurer, began his trailblazing career in the video game world in 1980 with the release of *MISSILE COMMAND*, a seminal trackball-based shooter that was a milestone in early interactive games.

Following on from this in 1981, Theurer created the timeless vector-based tube shooter *TEMPEST*, which inspired a slew of other innovations in arcade video games and was an early title to use 3D perspective in gameplay.

As his final title in the game industry before moving to a successful career in enterprise software, Theurer designed the cult, ground-breaking arcade title *I, ROBOT*. This 1983 arcade game, not

commercially successful at the time, is legendary for being the first commercial video game with filled 3D polygon graphics, as well as the first video game to feature camera control options—and was years or even decades ahead of its time.

"It's very difficult to find a game developer who doesn't have a single memory of *MISSILE COMMAND* or his other classic, *TEMPEST*," said Meggan Scavio, general manager of the Game Developers Conference. "We're delighted to honor Dave Theurer for his work as a designer which resulted in shaping so many developers' creative drive in the genre."

### AMBASSADOR

» The Ambassador Award recognizes individuals who have helped the game industry advance to a better place, either through facilitating a better game community from within, or by reaching outside the industry to be advocates for video games to help further the art form.

This year, the Choice Awards Advisory Committee voted the First Amendment lawyers in the historic U.S. Supreme Court case *Brown v. EMA* as recipients of the Ambassador Award. Ken Doroshov and Paul M. Smith led the legal team that convinced the Court that content-based restrictions on games are unconstitutional. The landmark ruling established First Amendment rights for those who create, develop, publish, and sell video



games, and is incredibly important to the past, present, and future of video games as a creative medium.

Ken Doroshov is senior vice president and general counsel of the Entertainment Software Association (ESA) in Washington, D.C. As the ESA's General Counsel, Ken oversees all the association's legal matters, including litigation, business affairs, and intellectual property policy.

The lead external lawyer on the case was Paul M. Smith of Jenner & Block LLC, chair of the appellate and Supreme Court Practice and Co-Chair of the Media and First Amendment, and Election Law and Redistricting Practices at his firm. He has had an active Supreme Court practice for nearly three decades, including oral arguments in 14 Supreme Court cases involving matters ranging from free speech and civil rights to civil procedure.

"The dedication that both Doroshov and Smith brought to the *Brown v. EMA* case will forever make them heroes to anyone who understands the value of this industry," added Scavio of the duo. "With

their legal teams, these two lawyers advanced the games industry in such a way that developers' livelihoods and intellectual properties are protected."

### LIFETIME ACHIEVEMENT

» With a career in games spanning nearly 30 years, Warren Spector has earned a reputation in the industry as a seminal designer and a champion for the proper execution of ideas in video games.

His work on the career-defining *DEUS EX* took place while he was serving as a partner at development company Ion Storm and running its Austin-based office. Upon its release in 2000, *DEUS EX* received wide critical and industry acclaim and in 2009 was named "Best PC Game of All Time" among a list of 100 other titles in *PC Gamer* magazine.


In 2004 Spector left Ion Storm, and the following year established Austin-based video game development firm Junction Point. Junction Point was acquired by Disney Interactive Studios in 2007. Immediately following, Spector began leading the design of *DISNEY EPIC MICKEY*, which released in 2010 and marked his first title as part of Disney Interactive Studios.

The game featured Spector's hallmark style of choice and consequence gaming, which he refers to as "Playstyle Matters," and was praised for its unique storyline, charming art design, and the tribute it paid to 80 years of rich Disney history.

Since beginning

his gaming career at Steve Jackson Games in 1983, Spector has played a key role when it comes to redefining genres. As a producer and designer on titles like *TSR, Inc.'s TOP SECRET/S.I.* and *MARVEL SUPER HEROES*, Origin's award-winning *ULTIMA* game series, including *ULTIMA WORLDS OF ADVENTURE 2: MARTIAN DREAMS*, *ULTIMA VII: SERPENT ISLE*, and *ULTIMA UNDERWORLD*, as well as Looking Glass Technologies's critically acclaimed *SYSTEM SHOCK*, Spector demonstrated his ability to open up new avenues in the role-playing arena and provide players with a fresh gameplay experience.

The Lifetime Achievement Award recognizes the career and achievements of a developer who has made an indelible impact on the craft of game development and games as a whole, and Warren Spector, who has earned a sterling reputation as an innovator able to merge the deep gameplay elements of multiple genres, stands as a shining example of those principles.

"Warren, whose rarified 'big picture' thinking and ideals have done a great deal for the games industry, exemplifies the exact qualities that a Lifetime Achievement Award recipient should possess," said Meggan Scavio. "In presenting him with this honor, we continue a tradition of highlighting individuals whose work stands as a benchmark for the next generation of developers." 



# DEITY

<http://www.deity-game.com>

THERE'S NO DENYING THAT STUDENT PROJECT DEITY HAS AMBITION. THE GAME TAKES CUES FROM THE LIKES OF DIABLO, ASSASSIN'S CREED, AND BATMAN: ARKHAM ASYLUM, BLENDING ISOMETRIC ACTION WITH STEALTH-BASED STRATEGY. PLAYERS WORK THEIR WAY THROUGH A DANGEROUS MEDIEVAL CASTLE, ALL THE WHILE HUNTING GUARDS, LURKING IN THE SHADOWS, AND OTHERWISE ESCAPING CERTAIN DOOM. GAME DEVELOPER RECENTLY SPOKE TO THE DIGIPEN STUDENT TEAM BEHIND THE GAME TO LEARN MORE ABOUT ITS INSPIRATION AND ORIGIN.



## Tom Curtis: How did you all come up with the concept for DEITY?

During the early stages of the project, most of our ideas focused on creating a stealth game emphasizing light and shadows. The first playable prototype used a third-person camera, where the player had to manipulate the environment's lights and hunt guards holding flashlights, but we settled on the isometric perspective because we identified a nostalgic charm attached to it—it also removed the need for additional camera controls.

## TC: For a game of this scope, DEITY is surprisingly complex, both aesthetically and in terms of its stealth- and action-based gameplay. Was this part of the original intent? How did you all coordinate/delegate the workload to get everything done on time?

We definitely [wanted] to make DEITY feel strategic, but our main priority was to ensure that the game was fun. Continuous playtesting resulted in the team making gradual improvements to the mechanics and level design that revolved around our central ideas. Oftentimes, we would make

drastic gameplay changes several days before milestones because we realized that it just wasn't as fun or engaging as we hoped. Most of DEITY's artistic inspiration came from dungeon crawlers, and we wanted to maintain that same gritty look and feel that most of these games feature.

The team would have an in-depth meeting after each milestone to decide what our individual goals were for the next project milestone. We would also officially meet twice a week during the times that our schedules didn't conflict (conducting SCRUM-like sprints so we could quickly discuss our progress, problems, and work plans).

Having to deal with schoolwork and a full game project wasn't easy for us, and we would often have to endure unforeseen delays. As such, most of our ideas and concepts were planned with scalability in mind. Our art assets were reusable (environment pieces were modular and tile-able) and our gameplay features could be cut.

## TC: What games were your greatest influences, and how did they impact the final project?

BATMAN: ARKHAM ASYLUM and ASSASSIN'S CREED were some of our sources of inspiration for the stealth experience. Being able to swoop in and out and eliminate your targets without being caught was something we loved about those games. The idea was to bring a similar experience to DEITY: You had visible superiority over your enemies, but your movement and attack strategies determined your success. We wanted to deliver a unique experience while preserving a similar stealth feel; gameplay mechanics and level designs had to be constantly tweaked to achieve that.

DIABLO and TORCHLIGHT had an influence on our artistic and interface design for DEITY. Those references helped us to identify the potential camera viewing problems inherent with our viewing angle (tall objects and walls had to be avoided) and we gained a greater understanding of darker, organic atmospheres. The artists on the team did a phenomenal job executing our vision, and really made the game stand out visually.

As for the controls, it was initially modeled after a combination of WORLD OF WARCRAFT and DIABLO, but we realized that the added complexity could have been devastating for players who were unfamiliar with those titles, so we streamlined the controls onto the mouse. This had a tremendous effect on lowering the learning curve, which led to non-gamers picking up and enjoying our game.

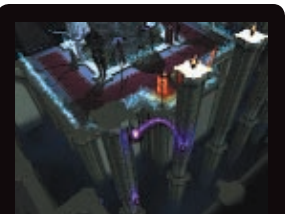
## TC: Any lessons you took from DEITY's development?

One of the most important lessons we took away from the project was that developing an engaging, polished game requires

teamwork. Being able to trust in each of our abilities to deliver, respecting personal schedules, and valuing the opinion of every team member was crucial to improving productivity.

We learned to ask what each of us could do and not to impose tasks. The various roles in the game's production were volunteered for, not assigned, which led to every person putting in amazing effort without being compelled by others to work.

DEITY has been the most rewarding and successful project for all of us on the team, and a large part of what made that happen was the synergy and perseverance we managed to uphold. 🎮



**Developer/Publisher:** DigiPen Institute of Technology  
**Release Date:** 16-Dec-2011  
**Platform:** Windows PC  
**Number of Developers:** 8  
**Length of Development:** 15 months  
**Budget:** \$0  
**Lines of Code:** Over 70,000 lines of C++ code  
**Fun facts:**  
 Everyday Value Slams @ Denny's weekly consumption: 10  
 The Longest Day: 52 hours  
 Instant noodle packs consumed: Approximately 500

**Team Members**  
 Ryan Chew  
 Caroline Sugianto  
 Christopher Mingus  
 Ryan Hickman  
 Michael Travaglione  
 Ying Liu  
 Matt Frederick  
 Ariel Hall

# TURN YOUR PASSION FOR GAMING INTO A CAREER

## Game Art

Bachelor's Degree Program  
Campus & Online

## Game Development

Bachelor's Degree Program  
Campus

## Game Design

Master's Degree Program  
Campus

## Game Design

Bachelor's Degree Program  
Online



**FULL SAIL**  
UNIVERSITY.

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

Winter Park, FL

800.226.7625 • 3300 University Boulevard

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

To view detailed information regarding tuition, student outcomes, and related statistics,  
please visit [fullsail.edu/outcomes-and-statistics](http://fullsail.edu/outcomes-and-statistics).

## 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

\*Title IV federal student financial aid funding is not available for this program. Full Sail has applied for approval to award Title IV funds for this program and if granted, students will be advised of the availability of such funds as applicable.

## Online Degrees

### Master's

- Creative Writing
- Education Media Design & Technology
- Entertainment Business
- Innovation & Entrepreneurship\*
- Internet Marketing
- Media Design
- New Media Journalism
- Public Relations\*

### Bachelor's

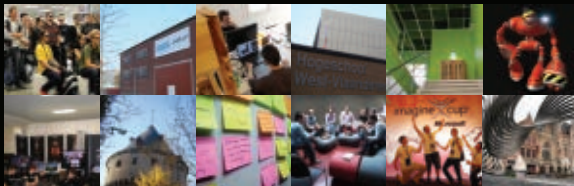
- Computer Animation
- Creative Writing for Entertainment
- Digital Cinematography
- Entertainment Business
- ▶ Game Art
- ▶ Game Design
- Graphic Design
- Internet Marketing
- Media Communications\*
- Mobile Development
- Music Business
- Music Production
- Sports Marketing & Media
- Web Design & Development

# 3D SQUARE

Where ideas take shape.

3D Square is a competence center of Howest in the gaming and interactive 3D sector. 3D Square has two major goals: at the one hand supporting enterprises and knowledge institutions active in the gaming and interactive 3D sector; at the other hand guiding enterprises from other sectors in realizing their 3D projects.

A complete description of the activities of 3D Square can be found at [www.3DSquare.be](http://www.3DSquare.be)



Howest University College | Belgium | [www.howest.be](http://www.howest.be)

# INTERNATIONAL DIGITAL ARTS AND ENTERTAINMENT

MAJOR GAME DEVELOPMENT  
MAJOR 3D ARTS



## Become a technical 3D artist

UNIQUE IN EUROPE

Education: Bachelor's degree (3 years) Language: English

Location: Belgium-the centre of Europe

Curriculum: Industry-approved & award-winning

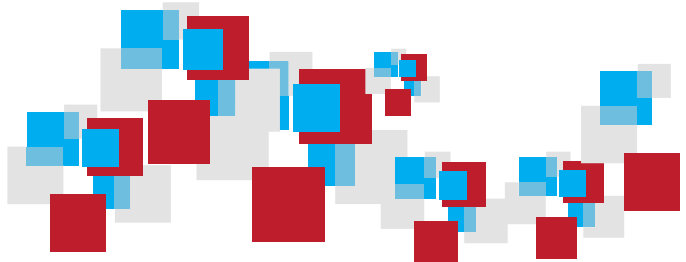
**Now accepting applications: limited entry**

Get your international career started and apply today.

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

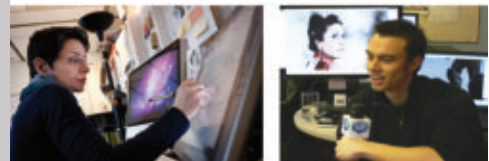
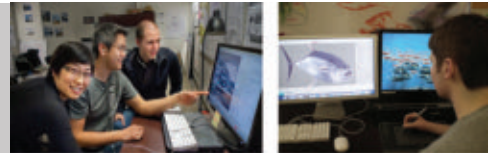


# PREPARING NEW LEADERS FOR THE DIGITAL MEDIA INDUSTRY



## MASTERS OF DIGITAL MEDIA PROGRAM

The Masters of Digital Media program (MDM) is Canada's premier professional graduate degree program in digital media and technology. Offered at Vancouver's Centre for Digital Media, the program includes internships and engages students in real world projects where they gain valuable leadership experience, hands-on training, and top industry connections.



**CENTRE FOR DIGITAL MEDIA**  
MASTERS OF DIGITAL MEDIA PROGRAM



### VISIT US

[mdm.gnwc.ca](http://mdm.gnwc.ca)  
[facebook.com/CentreforDigitalMedia](https://facebook.com/CentreforDigitalMedia)  
[twitter.com/CentreDigiMedia](https://twitter.com/CentreDigiMedia)

### SPEAK WITH AN ADVISOR

Yasmeeen Awadh  
Tel: +1 778.370.1010  
Toll Free: 1.855.737.2666



# ACADEMY of ART UNIVERSITY®

FOUNDED IN SAN FRANCISCO 1929 BY ARTISTS FOR ARTISTS



## TAKE CLASSES ONLINE OR IN SAN FRANCISCO

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

## ENROLL NOW

---

### EARN

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

### ENGAGE

IN CONTINUING ART EDUCATION COURSES

### EXPLORE

PRE-COLLEGE SCHOLARSHIP PROGRAMS

---

**[WWW.ACADEMYART.EDU](http://WWW.ACADEMYART.EDU)**

**800.544.2787 (U.S. Only) or 415.274.2200**

79 NEW MONTGOMERY ST, SAN FRANCISCO, CA 94105

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

*\*Acting and Landscape Architecture degree programs not currently available online.*

Visit [www.academyart.edu](http://www.academyart.edu) to learn about total costs, median student loan debt, potential occupations and other information.

*Photo credit: Joseph Taylor, Chris Haejin Chu*



# Start Living The Dream!



## A.S. Degree in Game Production

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

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

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

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

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

Scan for more Information



THE  
**LOS ANGELES**<sup>®</sup>  
FILM SCHOOL

ANIMATION + AUDIO + FILM + GAMES

Create Your Future Today. Call:

**800.406.7485**

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



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



## THE EDUCATION EXPERTS IN GAMES, 3D ANIMATION & VISUAL FX

Now Accepting Applications  
for Summer Semester

Limited Availability. Apply Now!

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

t: 206-428-6350 or 225-288-5227 e: [uscampus@aie.edu.au](mailto:uscampus@aie.edu.au)



ACADEMY OF  
INTERACTIVE ENTERTAINMENT

# GAME DEVELOPER MAGAZINE

the best of postmortems, product reviews, and standout columns

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

**\$49.95** /YEAR

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

**INCLUDES:**



PRINT SUBSCRIPTION



DIGITAL + GAME DEVELOPER APP



**BONUS!**



BEST OF POSTMORTEMS  
PRINT ISSUE

**SUBSCRIBE TODAY!**

**GDMAG.COM/SUBSCRIBE**



## ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
ACADEMY OF ART UNIVERSITY .....	53	MAGIC PIXEL .....	26
ACADEMY OF INTERACTIVE ENTERTAINMENT .....	54	MASTERS OF DIGITAL MEDIA PROGRAM .....	52
EPIC GAMES .....	19	RAD GAME TOOLS .....	C4
FMOD BY FIRELIGHT TECHNOLOGIES .....	6	RESEARCH IN MOTION .....	C2
FULL SAIL REAL WORLD EDUCATION .....	51	RED 5 STUDIOS .....	3, 12
GAMESPOT .....	44	TWOFOUR54 .....	11
HOWEST UNIVERSITY 3D SQUARE .....	52	UMBRA SOFTWARE .....	14
IGDA .....	32	UNITY WORKSHOP .....	37
INTEL .....	28-29, C3	VANCOUVER FILM SCHOOL .....	37
LOS ANGELES 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 UBM LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is \$49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: \$59.95; all other countries: \$69.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to gd Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate gd Game Developer on any correspondence. All content, copyright gd Game Developer magazine/UBM LLC, unless otherwise indicated. Don't steal any of it.



THE COMPLETE GUIDE

# HOW TO ANNOY YOUR TESTERS



Sick and tired of the shaky foundation upon which your game rests? Lots of open bugs in the database getting you down? Below, you'll find our complete guide to ensuring your game testers are just as miserable as you are!

## PART 1: KNOWN ISSUES.

» Don't read your bugs. Read only the first five words of the bug, and then route it to other people on the team based on those five words. Send back random bugs as fixed without reading the bug. When they're returned open, send them back as not a bug. Write "That's a known issue" on a bug and send it back, thinking the tester will close the bug because now you know about it. Write "It works fine for me" in the comments and send it back. Write "It works fine for me" in the comments and then sit on the bug for the rest of the project.

Let the bugs pile up in the database for months. Then, when you finally do get around to looking at the bugs, throw up your hands and say, "I don't have time to deal with these!" Afterwards, ask the testers to do a full regression pass on anything that's open in the database.

Tell the testers to come in early so they can have a build report ready by the time the rest of the studio gets in. Don't pay attention to the build report. Waste time getting builds that were indicated as broken in the build report.

## PART 2: THE ISSUES.

» When the testers raise performance issues, say "Well, duh, we haven't done the optimizations yet. Wait until we do optimizations to flag this as a problem!" Don't do any optimizations until the last possible moment. When

the optimizations fail to fix the performance issues and it's too late, say, "We should have been testing this."

When they raise compliance issues, write, "That won't be an issue." Forget that the discussion took place when the game fails its certification.

"When they raise game balance issues, tell the testers to stop raising game balance issues.

## PART 3: THWART PROGRESS.

» Create an Achievement that can be unlocked only by playing the game actively for 300 hours. Create Achievements that reward completion on the hardest difficulty setting in one sitting in a game with



one-hit kills and no checkpoints. Don't put in any debug commands to unlock them.

Ask the testers to do a pass through the full game progression, in all modes and difficulty levels, and then update the build with a small code change as soon as they're done and ask for it again.

## PART 4: WORD GAMES.

» Call the QA team "the Q and A team." Ask them, "So you guys just sit around playing games all day?" Ask them, "So do you fill out, like, a form or something when you're done?" Jokingly ask if they can smuggle you a pre-release build of so-and-so upcoming game—they've definitely never heard that one before.

Ask them if they watch PlayStation Network's "The Tester." Ask if PlayStation Network's "The Tester" is what it's like to test a game.

When reviewing games, assume that any bug found by a tester is automatically being fixed by someone. Ignore the idea that hundreds of bugs could be found and documented by the test team and not fixed by the development team. In your review of the game, write, "I can't believe the testers didn't catch this problem!" or, "What were the testers doing over there? Not their jobs, clearly."


When someone says they're a tester, smile and tell them, "Actually, testing is an important part of the game development process!" even though they didn't say anything to denigrate themselves in the first place.

## PART 5: THE SHAKEDOWN.

» Get your bug numbers down by reducing the number of testers. Replace many of your testers with outsourced testers. Make

the few testers you've kept try to figure out what the bugs from the outsource testers actually mean. Make your in-house testers retest the bugs written by external testers.

Split hairs about the difference between code and content. Say, "At the end of the day, isn't code just another kind of content?" Mention that the code and content will be finalized at some point that you cannot articulate right now. Mention that the final spec of the game is in a state of flux. Mention that all the features that the testers are testing could change at any moment. Tell the testers that all your documentation is out of date and that there is no up-to-date documentation on how the game should actually work.

When you encounter a particularly difficult bug, send it back and tell the tester they must be lying. Question the tester's testing methodologies. Question the tester's eyesight and ability to perceive what actually happened in the build. Accuse them of flat-out lying. Challenge them to reproduce the bug right in front of your eyes. When they do, fume impotently, but don't ever credit them, even in your own mind, for exposing the holes in your grand design. 

**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).

ZOMBIE  
STUDIOS

intel®  
Software

# Intel® GPA<sup>†</sup> helps Zombie Studios deliver *Blacklight®: Retribution*

## ZOMBIE STUDIOS OPTIMIZES WITH INTEL® GRAPHICS PERFORMANCE ANALYZERS



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

*"Ultimately, the results of optimizations we did with Intel GPA tools made the Blacklight: Retribution experience better and better for our players."*

— CHANCE LYON, LEAD DEVELOPER,  
ZOMBIE STUDIOS

### Intel® GPA System Analyzer

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

### Intel® GPA Frame Analyzer

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

### Intel® GPA Platform Analyzer

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

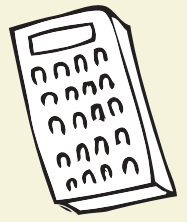
**DOWNLOAD INTEL® GRAPHICS PERFORMANCE ANALYZERS  
FOR FREE at [www.intel.com/software/gpa](http://www.intel.com/software/gpa)**



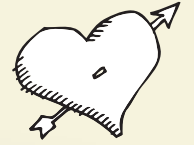
IF YOU WANT TO BE AN EVEN MORE

# AWESOME

I ♥



# GAME DEVELOPER



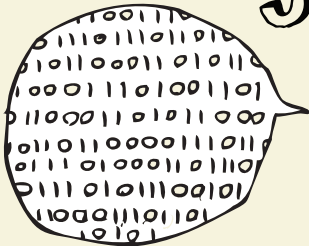
THEN YOU NEED TO BE USING

# Granny 3D



You get a run-time dynamic

## 3D ANIMATION SYSTEM with



AMAZING content exporters

EASY data manipulation

and a brand **NEW** blend graph editor.



# MUSTACHES



USING OUR TOOLS NOT ONLY MAKES YOU

MORE awesome, THEY MAKE YOU rad!



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