



United Business Media

FEATURE: 2009 GAME MIDDLEWARE SHOWDOWN

VOL16NO7 AUGUST2009

# gamedeveloper

THE LEADING GAME INDUSTRY MAGAZINE

# the conduit

postmortem

**CINEMATIC CAMERAS**  
and the art of manipulation

**DAMAGE ARBITRATION**  
the quick and the dead



# EYE TRONICS 3D

**3D SCANNING  
SOLUTIONS**



**CUSTOMERS INCLUDE:**  
**ROCKSTAR**  
**ELECTRONIC ARTS**  
**UBISOFT**  
**SONY**  
**2K INTERACTIVE**  
**MIDWAY, THQ, etc...**



**EYE TRONICS**  
[www.eyetronics.com](http://www.eyetronics.com)



**800.205.9808.EU +32.16.298309**



## POST MORTEM

### 30 HIGH VOLTAGE'S THE CONDUIT

THE CONDUIT represents High Voltage's first major foray into the world of original IP, after years of making licensed product. The lack of an external license to guide their choices and a publisher to keep them on track proved difficult—this postmortem provides good information for any company looking to make that leap. *By Josh Olson and Eric Nofsinger*

## FEATURES

### 7 MIDDLEWARE SHOWDOWN

More and more companies are using middleware to alleviate the pain of rising development costs. But there's not a whole lot of information out there. Our survey shows what developers want from middleware, and which packages are most popular in each field. *By Mark DeLoura*

### 18 BANG! ARE YOU DEAD?

Determining damage in online player-versus-player games requires an arbiter, not only to make sure that hits are correctly scored, but also to circumvent cheaters. Sony's Ronald Roy shares some tips. *By Ronald Roy*

### 23 THE FEARFUL EYE: CINEMATIC CAMERAS

Most games don't really aim for cinematic excellence, preferring instead to show as much of the game as possible. But it's possible to do both as this article proves through a careful dissection of the design considerations involved. *By Chris Pruett*

## DEPARTMENTS

2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]  
First-Person

### 4 HEADS UP DISPLAY

The best game mods of 2009, Amiga Forever updates, the 2010 IGF submission timeline, and more. [NEWS]

36 **TOOL BOX** *By Andrew Jones, Tom Carroll* [REVIEW]  
Corel Painter 11, Art DVD reviews

41 **THE INNER PRODUCT** *By Noel Llopis* [PROGRAMMING]  
Procedural Content Creation

45 **PIXEL PUSHER** *By Steve Theodore* [ART]  
Check Out That Asset!

49 **DESIGN OF THE TIMES** *By Soren Johnson* [DESIGN]  
Turn-Based Vs. Real Time

51 **AURAL FIXATION** *By Jesse Harlin* [SOUND]  
Retro Fitting In

56 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]  
Staying Sane





# FIRST-PERSON

## IS IT REALLY MORE IMMERSIVE?

### TALKING WITH OUR PREVIOUS ART DIRECTOR

recently, I recalled something I'd forgotten: First-person games can be quite intimidating. We tend to accept it as a de facto viewpoint for several popular genres today, and it also saves developers from having to develop a camera system independent of the player's control. But it is rather daunting, and has a high learning curve for those who haven't already experienced many first-person games.

The art director in question is a more casual player, and to her, first-person games seem disorienting and conceptually difficult. Talking about this reminded me of my first FPS experience, in WOLFENSTEIN 3D for the Atari Jaguar. This was my first interaction [in perhaps '97 or '98] with a "real" FPS. I tried playing the game for about an hour, and came away dizzy and unable to read, because my eyes were jumping around on the page.

This experience had me pretty convinced that first-person games weren't for me, all the way until HALO 2 hit consoles and someone convinced me to give it another shot. Perhaps that's not a good thing for an editor of *Game Developer* to admit, but it's true. I have since learned the power of the first-person viewpoint, in terms of what you can show on screen, and the interactions that become possible. But I spoke with a few of my fellow editors, and several had recollections of difficulty penetrating that first-person wall.

The reason is likely that we are used to seeing games and movies play out before us in a third-person view. Having an avatar gives us a strong frame of reference, and allows us to better navigate the world. If I see a little running guy, and I try to make him jump, I can gauge that distance. If I have to jump in first-person mode, where are my feet? Are they below the camera directly? How far can I jump, when everything feels like it's based on my perspective? If I look up a bit, the platform in front of me looks different than it did before.

A 14-year-old boy will take the time to figure this out, and will wind up having an excellent experience. An older or more casual user will likely be much more daunted, and less inclined to even pick up such a title.

### THE IMMERSION QUESTION

» Are first-person games inherently more immersive? A lot of developers seem to think so, but let's take a second look. Consider the last time you felt like you actually *were* the character in a game you played. I'd be willing to guess that most people will say "never." We don't generally take on the role of the character we're playing, except as children in imaginary play. What we do is identify with the character—and how can you identify with a character you can't see? A character that usually doesn't even talk, or have any opinions about the

horrible things going on around him? This goes back to the "silent hero" dilemma that has existed ever since role-playing made its way into the electronic world, notoriously perpetuated by the Japanese console RPG.

Almost all first-person games have this sort of silent character—one whose only interaction with others is usually taking orders until they turn their backs, and then just shooting and collecting things. That doesn't seem inherently immersive to me. It can be, but it isn't necessarily, as is often assumed. Western RPGs like FALLOUT 3 (or earlier games like ULTIMA IV) do a somewhat better job by at least allowing the player to make some dialog choices, but still, the character isn't you.

What makes a game immersive or otherwise is not the viewpoint, of course; it's the situations, external characters, and tasks that get you involved. One of the characters I've identified most with is the boy from ICO, and he doesn't even speak a real language. The oppressive environments and his seeming innocence simply made him a sympathetic character. It's difficult to empathize or identify with a camera or floating gun. I can empathize with De Niro's character in *Once Upon a Time in America*, even though I don't agree with what he does, simply because his world is so well-realized, and I can see how he reacts to events. In first-person games, there is no reaction on the part of the character, and it becomes difficult to feel anything about him or her.

First-person games are incredibly important to the industry, and have moved many genres forward in significant ways. The viewpoint is doubtlessly here to stay, and I want to emphasize that I am actually a fan of the concept. But I do think it's worth taking a step back. I feel that as an industry we've come to our own conclusion that first-person games are inherently intuitive and more immersive, simply by virtue of their camera position, and in spite of the problems they bring up. I would submit that just because we've gotten used to this style of game doesn't mean everyone has. It's important to realize that making a first-person game almost necessarily means making a game for the dedicated gamer.

### BREAK DOWN THE WALL

» Innovations on the interface side could help lower the casual block, perhaps through the Wii, Project Natal, or the PS3's new motion controller (THE CONDUIT does some work in this direction—see the postmortem on page 30). Regardless, it will take a lot of work and concerted effort to penetrate the casual audience with a first-person camera. The question is whether we even need to, when there are so many camera systems that games have yet to fully explore.

—Brandon Sheffield

Think Services, 600 Harrison St., 6th Fl.,  
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)

#### EDITORIAL

##### PUBLISHER

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

##### EDITOR-IN-CHIEF

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

##### PRODUCTION EDITOR

Jeffrey Fleming | [jffleming@gdmag.com](mailto:jffleming@gdmag.com)

##### ART DIRECTOR

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

##### SENIOR CONTRIBUTING EDITOR

Jill Duffy | [jduffy@gdmag.com](mailto:jduffy@gdmag.com)

##### CONTRIBUTING EDITORS

Jesse Harlin | [jharlin@gdmag.com](mailto:jharlin@gdmag.com)  
Steve Theodore | [stheodore@gdmag.com](mailto:stheodore@gdmag.com)  
Noel Lopis | [nlopis@gdmag.com](mailto:nlopis@gdmag.com)  
Soren Johnson | [sjohnson@gdmag.com](mailto:sjohnson@gdmag.com)  
Damion Schubert | [dschubert@gdmag.com](mailto:dschubert@gdmag.com)

##### ADVISORY BOARD

Hal Barwood | Designer-at-Large  
Mick West | Independent  
Brad Bulkley | Neversoft  
Clinton Keith | High Moon Studios  
Ryan Lesser | Harmonix  
Mark DeLoura | Independent

#### ADVERTISING SALES

##### GLOBAL SALES DIRECTOR

Aaron Murawski | [amurawski@think-services.com](mailto:amurawski@think-services.com)  
t: 415.947.6227

##### MEDIA ACCOUNT MANAGER

John Malik Watson | [jmwalson@think-services.com](mailto:jmwalson@think-services.com)  
t: 415.947.6224

##### GLOBAL ACCOUNT MANAGER, EDUCATION AND RECRUITMENT

Gina Gross | [ggross@think-services.com](mailto:ggross@think-services.com)  
t: 415.947.6241

##### COORDINATOR, EDUCATION AND RECRUITMENT

Rafael Vallin | [rvallin@think-services.com](mailto:rvallin@think-services.com)  
t: 415.947.6223

#### ADVERTISING PRODUCTION

##### PRODUCTION MANAGER

Robert Steigleider | [rsteigleider@ubm-us.com](mailto:rsteigleider@ubm-us.com)

#### REPRINTS

##### WRIGHT'S REPRINTS

Ryan Pratt | [rpratt@wrightsreprints.com](mailto:rpratt@wrightsreprints.com)  
t: 877.652.5295

#### THINK SERVICES

CEO THINK SERVICES | Philip Chapnick  
GROUP DIRECTOR | Kathy Schoback  
CREATIVE DIRECTOR | Cliff Scorsio

#### AUDIENCE DEVELOPMENT

GROUP DIRECTOR | Kathy Henry  
e: [khenry@techinsights.com](mailto:khenry@techinsights.com)  
DIRECTOR | Kristi Cunningham  
e: [kcunningham@techinsights.com](mailto:kcunningham@techinsights.com)  
LIST RENTAL | Merit Direct LLC | t: 914.368.1000

#### MARKETING

SERVICES MARKETING COORDINATOR | Laura Robison  
e: [lrobison@think-services.com](mailto:lrobison@think-services.com)

#### UBM TECHNOLOGY MANAGEMENT

CHIEF EXECUTIVE OFFICER | David Levin  
CHIEF OPERATING OFFICER | Scott Mozarsky  
CHIEF FINANCIAL OFFICER | David Wein  
CHIEF INFORMATION OFFICER | Kevin Prinz  
CORPORATE SENIOR VP SALES | Anne Marie Miller  
SENIOR VP, STRATEGIC DEV. AND BUSINESS ADMIN. | Pat Nohilly  
SENIOR VP, MANUFACTURING | Marie Myers





# STOP THE ZOMBIES

## LIVE 'EM BRAIN

We are tired of stupid zombies populating games. Help us and **give 'em brain**. Sign up **NOW** for **FREE** and download the **NEW** xaitment **BrainPack** SDKs

Set up and control complex game logic in a few steps. Generate your perfect navigation mesh with a single click. Create realistic behavior in no time.

Join the xaitment community now and turn your idea into a stunning prototype without paying any license cost. By signing up for free, you'll receive our complete modular AI Engine plus our world-class support. Experience the future of next gen game technology and work with the smartest AI technology available.

Contact xaitment today for more information about the BrainPack Program under [brainpack@xaitment.com](mailto:brainpack@xaitment.com) or visit our website [www.xaitment.com](http://www.xaitment.com)



GIVE 'EM  
BRAIN

 **XAITMENT**  
INTELLIGENCE ENGINEERED





## MOD SCENE 2009

NAVIGATING THROUGH THE DEPTHS OF MODDB'S MOD LISTINGS CAN BRING UP A MIXED BAG AT THE BEST OF TIMES. WE'VE SIFTED THROUGH THE ARCHIVES FOR YOU, AND FEEL THAT THESE MODS REPRESENT SOME OF THE BEST OFFERINGS RELEASED OR UPDATED IN 2009.

—Ryan Anderson, ModDB

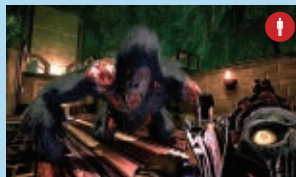


### AGE OF CHIVALRY

#### HALF-LIFE 2

[www.age-of-chivalry.com](http://www.age-of-chivalry.com)

Parry, stab, and slash your way to victory in this medieval, class-based Source mod. Not many mods pull off melee combat, but AGE OF CHIVALRY provides a satisfying (and very gory) representation. Extensive Steamworks support makes this the first mod to offer detailed stats and achievements.

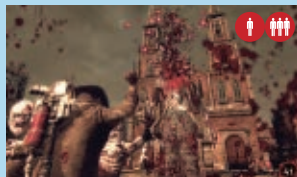


### THE BALL

#### UNREAL TOURNAMENT 3

<http://theball.toltecastudios.com>

THE BALL has you exploring ancient Aztec-style ruins and fighting the animated remains of its former inhabitants with nothing more than a large, magical orb. The game's puzzles are nicely varied, and play on the connection between you and the ball; it's not until you wander into trouble without it that you realize how much of a friend it really is.



### THE HAUNTED

#### UNREAL TOURNAMENT 3

[www.moddb.com/mods/the-haunted](http://www.moddb.com/mods/the-haunted)

Zombies and demons will probably never go out of style, and THE HAUNTED offers a nice take on the subject in terms of both aesthetic and pacing. The quick arcade action rewards your kills with weapon upgrades and teammate revival checkpoints keep you from camping in any one place too long. Gorgeous to look at, fun to play, and free (for UT3 users)—what more could you ask for?



### RADIATOR

#### HALF-LIFE 2

[www.radiator.debaclue.us](http://www.radiator.debaclue.us)

RADIATOR is a series of short-form, experimental single player experiences. The first two episodes play on themes of love, astronomy, memory repression, and gay marriage. The mechanics in each of these episodes are quite limited, but the emotional impact justifies this artsy mod's placement on our list.

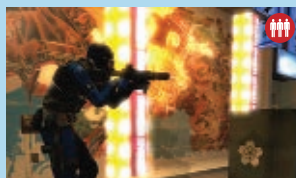


### THE NAMELESS MOD

#### DEUS EX

<http://thenamelessmod.com>

A colossal undertaking several years in the making, NAMELESS drops you into Forum City as an avatar named Irestkon. The environment itself is a "physical embodiment of Internet forums and bulletin boards," but is filled with just as much conspiracy and intrigue as the world of JC Denton. A branching, responsive storyline with high quality voiceovers and high definition graphics upgrades make this mod a must-play.

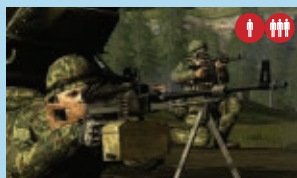


### NEOTOKYO\*

#### HALF-LIFE 2

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

At first glance, NEOTOKYO\* seems to be another COUNTER-STRIKE clone. Spend some time with it and you'll find there are several nuances that set it apart from the pack. The combination of vision modes and therm-optic camouflage offers unique checks and balances, while the "Capture the Ghost" game mode encourages tight team coordination. Impressive visuals assembled by a team of industry pros speak for themselves.



### PROJECT REALITY

#### BATTLEFIELD 2

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

PROJECT REALITY is a mod that turns BATTLEFIELD 2 into a perfect blend of arcade and simulation elements. Strategic planning and team coordination are required, but the action focuses around specific capture zones. Utilizing a force of combined arms, whether it's infantry, armor, or air support, is the key to winning. The best part: you don't have to spend half the match traveling to where the action is.



### FALL FROM HEAVEN II

#### CIVILIZATION IV: BEYOND THE SWORD

[www.civfanatics.com/fff](http://www.civfanatics.com/fff)

FALL FROM HEAVEN plunges you into a world of dark fantasy built on the mechanics behind CIVILIZATION IV. It adds an incredible amount of content; choose from two dozen different factions—each with their own heroes, magic, religions and technology. There's even a card-based diplomatic mini-game! FALL impressed Soren Johnson (CIV IV Designer) enough to write up an interview with the creators ([www.designer-notes.com/?p=120](http://www.designer-notes.com/?p=120)).

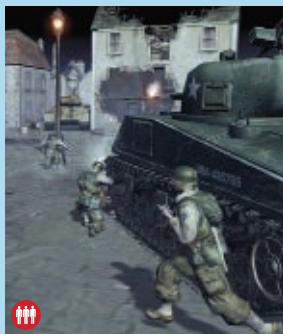


### THIRD AGE – TOTAL WAR

#### MEDIEVAL II: TOTAL WAR KINGDOMS EXPANSION

[www.twcenter.net/forums/forumdisplay.php?f=654](http://www.twcenter.net/forums/forumdisplay.php?f=654)

THIRD AGE is a fan-made imagining of the Lord of the Rings universe. A complete and detailed campaign map of Middle Earth, along with 12 factions straight from the lore allow you to create large-scale battles of epic proportions. The mod features a plethora of other custom content, from new voice work to cinematic cut-scenes that round out the entire experience.



### EUROPE IN RUINS: REINFORCEMENTS

#### COMPANY OF HEROES/COMPANY OF HEROES: OPPOSING FRONTS

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

EUROPE IN RUINS transforms CoH into a persistent environment in which you command your own battalion. Experience points carry over from previous battles, allowing you to unlock new abilities and increase your options on the battlefield. Custom maps and units tailored to EIR's unique style of play make this a deeper strategic experience than the original game.



THE 12TH ANNUAL  
INDEPENDENT  
GAMES FESTIVAL



## 2010 INDEPENDENT GAMES FESTIVAL OPEN FOR SUBMISSIONS

**ORGANIZERS OF THE INDEPENDENT GAMES FESTIVAL** (OWNED by Think Services, which also publishes Game Developer) have announced that the 2010 edition of the venerable festival and awards show honoring the best in indie games, is open for submissions.

Finalists will be playable on the GDC 2010 show floor, and winners will compete for almost \$50,000 in prizes. The IGF is an excellent place not only to get noticed, but also a place to launch a legitimate career as an independent developer. Below is the timeline of dates for the competition.

- November 1st, 2009** – Submission Deadline, Main Competition
- November 15th, 2009** – Submission Deadline, Student Competition
- January 4th, 2010** – Finalists Announced, Main Competition
- January 11th, 2010** – Finalists Announced, Student Competition
- March 9th–10th, 2010** – Indie Games Summit at GDC
- March 11–13th, 2010** – IGF Pavilion at GDC
- March 11th, 2010** – IGF Awards Ceremony (Winners Announced)

—Staff

## PAX 10 ANNOUNCED

**THE PENNY ARCADE EXPO IS FRIEND TO ALL MANNER OF INTERACTIVE** creative endeavor, and to that end, the organizers have chosen the PAX 10, ten indie games that will be featured during the popular expo. The list was culled from over 150 entries, and is displayed below.

- CARNEYVALE: SHOWTIME by the Singapore-MIT GAMBIT Games Lab (Xbox 360)
- CLOSURE by Tyler Glaiel and Jon Schubbe (PC)
- FIELDRUNNERS by Subatomic Studios (iPhone)
- LIGHT by Studio Walljump (Wii)
- MACHINARIUM by Amanita Design (PC)
- OSMOS by Hemisphere Games (PC)
- PUZZLE BLOOM by Team Shotgun (PC)
- TAG: THE POWER OF PAINT by Tag Team (PC)
- TRINO by Trinoteam (Xbox 360)
- WHAT IS BOTHERING CARL? by Story Fort (PC)

PAX 2009 takes place in Seattle from September 4–6.

—Staff



## AMIGA FOREVER

### AMIGA FOREVER, A ROBUST

emulator of Amiga software and its operating system, has just released a new version for 2009. ([www.amigaforever.com](http://www.amigaforever.com)) Cloanto, an Amiga developer since 1986, has updated the package with Windows 7 support, smaller file sizes for games (using the RP2 format), new system ROMs, and an enhanced focus on usability, a traditional complaint from users of the software.

Amiga Forever runs almost all versions of Amiga software, and most importantly, games. The Plus Edition even supports the Amiga CD32, the ill-fated CD-based home console. There are three available packages: Value, which comes with the most common OS and 50 games and demoscene productions (respectively); Plus, which adds all ROM and workbench versions, 100+ games and demos, and a

historical gallery; and Premium, which adds two DVDs comprising 20 gigs of games, history, videos, and errata.

Some companies such as Factor 5 have made some of their Amiga titles available for free on their own web site ([www.factor5.de/downloads.shtml](http://www.factor5.de/downloads.shtml)), and other titles are legally downloadable through a variety of sites such as Amiga Future and Amigaland.

Efforts to preserve classic game culture like this are to be commended, though some might argue it should be a free service (those who feel that way might try the UAE emulator: [www.amigaemulator.org](http://www.amigaemulator.org)). While the old ways of computing may be dead and gone, their legacies resound throughout the game and software industries, and are certainly worth revisiting.

—Brandon Sheffield

### CALENDAR

#### SIGGRAPH 2009

- Ernest N. Morial Convention Center
- New Orleans
- August 3–7
- Price: \$45–1,175

[www.siggraph.org/s2009](http://www.siggraph.org/s2009)

#### GDC EUROPE

- Cologne Congress Center East

- Cologne, Germany
- August 17–19
- Price: 180–795 Euros
- [www.gdceurope.com](http://www.gdceurope.com)

#### DIGRA

- Brunel University
- West London
- September 1–4
- Price: £196–£372

<http://digra2009.newport.ac.uk>

#### PENNY ARCADE EXPO

- Washington State Convention Center
- Seattle
- September 4–6
- Price: \$30–\$55
- [www.pennyarcadeexpo.com](http://www.pennyarcadeexpo.com)



# KNIGHTS HAVE FACES TOO



Each year, countless game characters have their faces hidden from the world. This cruelty has got to stop!

It's our mission to end helmet tyranny by animating facial performances better and more affordably than any other solution on the market.

Join our cause today by calling **310.656.6565**  
or by visiting [www.image-metrics.com](http://www.image-metrics.com).

**image metrics**

© 2009. Image Metrics, Ltd. All rights reserved. Image Metrics is a trademark of Image Metrics, Ltd.



Proud Sponsor of:

**FRAHG**  
FRIENDS AGAINST  
HELMETS in GAMES



# middleware showdown

what middleware packages studios use and why

**THE CHOICE OF WHETHER TO USE LICENSED MIDDLEWARE LIBRARIES HAS TRADITIONALLY BEEN** a controversial one. We engineers are a proud lot, and there is a long tradition of “do it yourself” that goes back to the days of single designer/engineers cranking out Atari 2600 games. Game engines have become quite massive as platform capabilities have increased, but many of us still want to be able to write all the engine code, and feel it is a badge of honor to do so. Why rely on someone else?

These days, however, there are many useful libraries available—both free and licensed—and using them in your game can be a smart decision that frees you up to concentrate on building features that make your game unique. Instead of spending a bunch of time writing a cross-platform audio engine and tools, perhaps you can just license one. Why rewrite the inverse kinematics code again if you can go out and buy a version that works reasonably well for your style of game? Of course, there are financial considerations, and the available libraries may not do quite what you had envisioned. But with so many options available, the wise team leader will do a thorough middleware

exploration before making too many decisions about the game’s tech.

It can be tough to narrow down the choices. Which libraries have proven the most useful? Which middleware companies are the best to work with? And how can you quickly evaluate a handful of libraries to find the one that is best for your game? Unfortunately the answers to many of these questions are typically only found late at night over drinks with a friend. As one developer told us, “It’s harder than it should be to find other developers to talk to who are using the tech.”

A few months ago we put together a survey on game engines and the results were quite useful. Many smart





# middleware showdown

people were willing to share information and anecdotes about their experiences, knowing that the data would be anonymized, consolidated, and shared for the betterment of all. Here, we've done the same for middleware libraries.

## DEMOGRAPHICS

» As with the game engine survey, the responders to our middleware survey skewed toward the "core" game market. Over 100 senior developers from a variety of development houses responded to the survey, and most were working on PC, PlayStation 3, or Xbox 360 titles. Likewise, most considered their games to be more hardcore, but there was a heavy amount of casual games as well, with some crossover per developer. (See Figures 1 and 2.)

## CHOOSING MIDDLEWARE

» What drives a developer to choose middleware over writing internal software? A majority of the developers indicated saving time was the strongest motivation. The pressure to create a title of high production quality was indicated as the second strongest influence—as one responder noted, although high-fidelity titles can frequently be created more quickly by using middleware, for unique features of the game. "... [that] require doing something that off-the-shelf middleware isn't really tuned for, we may do something in-house." Fortunately, as that same person noted, the ability to innovate within the constraints of middleware continues to grow each year. (See Figure 3.)

The decision to use middleware is largely made by the technologists on a project—89% of responders called out engineers as the key influencers. Producers and other business managers also have an influence, but at a much lower level. (See Figure 4.)

It's long been appreciated that using game engines and middleware libraries can get the team up and running more quickly on the target platform during the development process, allowing artists and designers to mock up the game much earlier. Commenters noted, "While middleware is not perfect, it allows us to concentrate on the areas that really make our game different, rather than implementing basics. But programmers tend to not like [middleware] because it makes their lives more difficult (and the naïve ones think they can do it better)."

The most straightforward way to evaluate potential libraries is to have an engineer sit down with each one and work with it, integrate it, and experiment with it—and indeed this is called out as the most common solution. (See Figure 5.) But of course it is also extremely slow and costly. "That costs a LOT of money, but it is better than learning too late that it's the wrong solution," said one commenter. Another common technique for vetting the usefulness of a middleware library is to talk with other people who have used it. This is sometimes easier said than done, since there are usually a lot of NDAs in the way. But icons on boxes,



HELLGATE: LONDON made use of middleware from Havok.

## Figure 1 platform

### WHICH PLATFORMS ARE YOUR CURRENT PROJECT(S) FOR?

PC	68.6%
PLAYSTATION 3	61.9%
XBOX 360	61.0%
NINTENDO WII	32.4%
MOBILE	21.0%
XBOX LIVE ARCADE	20.0%
PLAYSTATION NETWORK	18.1%
SONY PSP	18.1%
MAC	14.3%
WIIWARE	11.4%
NINTENDO DS/DSI	10.5%

## Figure 2 classifications

### HOW WOULD YOU CLASSIFY YOUR CURRENT GAME?

CORE	63.5%
CASUAL	44.2%
SOCIAL	20.2%
MMO	13.5%
MOBILE	21.0%

## Figure 3 influences

### HOW MUCH DO THE FOLLOWING FORCES INFLUENCE YOU TO USE MIDDLEWARE (OUT OF 5.00)?

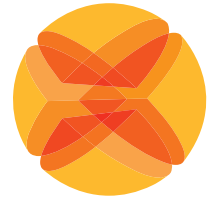
TIMELINE PRESSURES	3.87
FEATURE-SET OR PRODUCTION QUALITY PRESSURES	3.70
COST PRESSURES	3.37
ABILITY TO SHARE TECHNOLOGY AND STAFF ACROSS PROJECTS	3.27
DIFFICULTY OF FINDING STAFF WITH APPROPRIATE EXPERTISE	3.10
ABILITY TO PURCHASE CUTTING-EDGE TECHNOLOGY	3.09

## Figure 4 roles

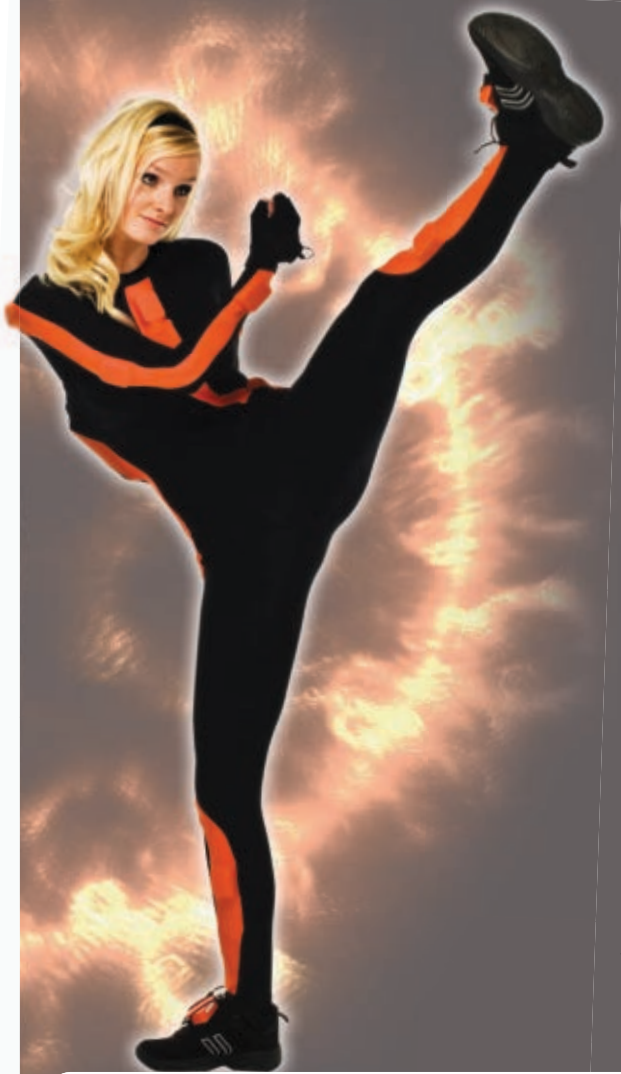
### WHO MAKES THE DECISION WHETHER TO USE MIDDLEWARE OR NOT AT YOUR COMPANY?

TECHNOLOGISTS	89.0%
BUSINESS MANAGEMENT	36.3%
PRODUCERS	31.9%
ARTISTS	24.2%
DESIGNERS	13.2%
PUBLISHER	13.2%

# Freedom of movement



**XSENS**



## Xsens MVN (formerly known as Moven) inertial motion capture

- Flexibility
- Quick turn around times
- Unlimited capture volume
- Real-time, On-set previsualization
- No occlusion
- Saves up to 80% post-processing time
- Clean and smooth data

Meet Xsens at:



**SIGGRAPH**2009  
Booth 2819

## JOIN THE MOVEMENT!

- Film:** Industrial Light and Magic, Sony Pictures Imageworks, The Third Floor, Double Negative, Filmayer Producciones Audiovisuales...
- Games:** Electronic Arts, THQ, Sony Computer Entertainment, 2K, Insomniac, Gearbox Software, Big Huge Games, Xaviant, Rocketbox Studios, Bigpoint...
- Television:** Animated Storyboards, Pipedreams 3D, Simaestudio...







# middleware showdown

## Figure 5 evaluate

HOW DO YOU TYPICALLY EVALUATE MIDDLEWARE LIBRARIES?

TECHNICAL PERSON WORKS WITH IT	4.47
TALK TO OTHER PEOPLE WHO HAVE USED IT	3.68
READ THROUGH THE DOCUMENTATION	3.63
EXAMINE OTHER GAMES THAT HAVE USED IT	3.57
TALK TO MIDDLEWARE SALES AND SUPPORT STAFF	3.08

## Figure 6 choosing

WHEN CHOOSING MIDDLEWARE, HOW IMPORTANT ARE THESE (OUT OF 5.00)?

PLATFORM AVAILABILITY	4.66
PERFORMANCE (SPEED)	4.33
FEATURE SET	4.19
SOURCE CODE ACCESS	4.10
KNOWN TO EASILY INTEGRATE WITH TECH I'M USING	4.05
PRICE	3.93
SUPPORT RESOURCES	3.90

## Figure 7 licensing

WHAT IS YOUR PREFERRED METHOD FOR PURCHASING MIDDLEWARE (OUT OF 5.00)?

PER-PROJECT LICENSE FEE	3.76
PER-PLATFORM LICENSE FEE	3.11
ANNUAL SITE LICENSE FEE	2.61
PER-USER LICENSE FEE	1.74

## Figure 8 royalties

WOULD YOU PAY BACK-END ROYALTIES TO REDUCE FRONT-END COSTS FOR MIDDLEWARE?

NO, FLAT-RATE ONLY:	86.0%
YES, BACK-END ROYALTIES:	14.0%

## Figure 9 pay for mods

WOULD YOU BE WILLING TO PAY FOR AUGMENTATION/MODIFICATION OF AN EXISTING LIBRARY?

DO IT OURSELVES	72.4%
PAY FOR MODIFICATIONS	27.6%

comments on web sites, and conversations over drinks certainly help. And what's a better recommendation than seeing a particular piece of middleware used in a game similar to your own? "We look at how long it has been in the marketplace and if AAA titles not developed by the middleware company have been [released]," said a respondent.

There are a lot of elements to consider when choosing middleware—the price, the features, and the support infrastructure are all important. [See Figure 6.] Many of those surveyed responded the criteria vary significantly based on the kind of middleware they're looking for. Perhaps for a piece of AI middleware the demands are different than for audio middleware, since the audio system is less tightly integrated into the game engine. Several also responded directly that trust in the middleware company and the level of support resources are incredibly important, since developers will be working with the firm closely. We'll discuss support in more depth later, but the issue of being able to trust the company you're working with came up repeatedly. Many mentioned having been bitten by middleware companies going out of business, or being purchased. Having a plan in place in case of such occurrences is vital.

### PURCHASING

» Middleware companies have a variety of different licensing plans, and we were curious what developers thought was most convenient. [See Figure 7.] As it turns out, the common per-project and per-platform licensing models are the most popular. What is surprising is the incredible dislike of the per-user licensing model—this hasn't been used by many middleware companies, but is common elsewhere in the industry, and one developer noted, "... per-user licenses don't make sense because you purchase as few as possible and it hampers development productivity." Good point!

Another common question is whether studios might be interested in paying a smaller up-front cost for middleware by allowing some royalties on the back-end after a title has shipped. [See Figure 8.] This is a more typical proposition with game engines, but a few middleware library companies have floated the idea as a possibility as well. Overwhelmingly, our survey responders said "no." A few noted that for small developers, being able to gain access to middleware without a huge up-front cost would be beneficial, however "...it is difficult for us to convince the publisher to pay back-end royalties for middleware." Several commented "As a developer, we rarely see back-end royalties [ourselves]." And one responder noted that with most middleware companies, "everything is negotiable," so there may be other options available for cash-strapped young studios.

With regard to purchasing middleware, our final query was, "What other services might a middleware company be able to provide?" Would developers be interested in paying a little extra to get a custom build with some special features? [See Figure 9.] Reaction was largely negative, although it varied depending on the extent of changes and the cost. "Simple changes we make; hard changes you make," seemed to be the overwhelming reaction. But many also said something along the lines of, "I'd like to answer 'both' here—sometimes we do it ourselves, but some modifications would benefit the larger user base so could be rolled back in..." and another reply asked, "Is it fair for them to charge us for the modifications, and then turn around and include that change into their middleware and sell it as a new feature to other developers?"

### POPULAR LIBRARIES

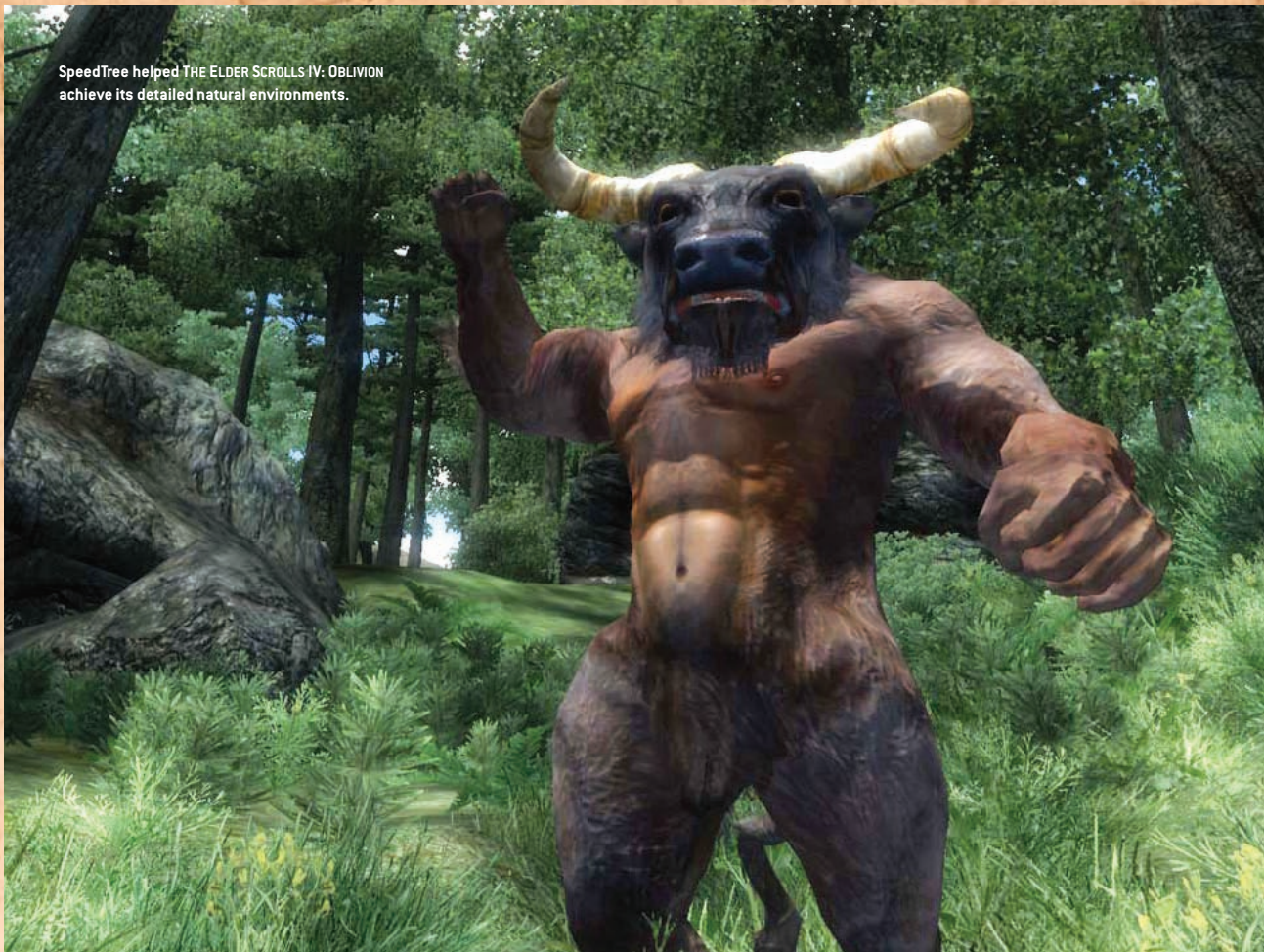
» So which middleware libraries should be on your short list? The survey asked developers to share with us which middleware they are using in their current projects, as well as which they have used or evaluated in the past and, frankly, which they had never heard of. The leaders in each middleware category were very clear from the replies we received. [Libraries not being used by any of the survey respondents were not included in these charts—see Figure 10, pg 12.]

**Physics.** Nearly 2/3 of the responders to our survey are using physics middleware. Among all the responses, Nvidia PhysX and Havok Physics are the most popular. Open source physics library Bullet has proven quite popular as well, though not quite as strong as the licensed physics middleware.

**Networking.** Networking used to be a very popular category of middleware library, but with some platform providers integrating a level of network services into their SDKs, this category has shifted significantly in recent years. About one quarter of our responders are using networking middleware in their current titles. Interestingly, 6.2% claim to be using Demonware, though Demonware is no longer licensable, as it has been purchased by Activision. It should also be noted that much of the Wii's networking capability is provided by GameSpy, so the percentage using GameSpy is technically somewhat higher.



SpeedTree helped THE ELDER SCROLLS IV: OBLIVION achieve its detailed natural environments.



**Audio.** Over 90% of our developers are using some kind of run-time audio middleware in their games; this certainly makes audio the most popular category for middleware assistance. FMOD is the most used audio middleware, with the open-source Ogg Vorbis codec also coming in quite high.

**Video.** In the video middleware category RAD Game Tools' Bink dominates, and over half of survey responders say that they use some sort of video middleware. 44.3% of those surveyed are using Bink. CRI Movie is used by 6.2% of the surveyed developers, and open source format Ogg Theora was mentioned as another format option by a few responders—although no one claimed to be using it in a shipping title.

**Artificial Intelligence.** Using middleware for AI is still a fairly controversial choice, but one that is slowly gaining traction, especially for fundamental AI techniques which can then be built upon for game-specific uniqueness. There are a lot of AI middleware libraries available now, but the leader by far is Autodesk's Kynapse. With 14.3% of our developers using Kynapse, it is the strongest middleware in this field by a substantial margin.

**Animation.** Nearly 2/3 of our responders claim to be using some kind of animation middleware, and there is certainly no shortage of software available here.

**Rendering.** There are fewer libraries available that assist with rendering tasks. These don't seem to be used as frequently as other categories of middleware, but SpeedTree has been available for quite some time and is most dominant at 15.3%.

**User Interface.** There is not much of a battle for user interface middleware at the moment; although there is a small handful of libraries available, all of the developers surveyed who use user interface middleware are using Scaleform Gfx (35.4%).

**Other.** There are other middleware libraries available which do not fit well into the above categories—script languages and compression libraries, for example. The open-source compression library Zlib is a very popular choice,

and scripting languages Lua and Python are also seeing heavy use in titles under development.

We also asked what other technologies developers wish were available as middleware solutions. There were many ideas shared by the survey responders, but the most popular were streaming systems and state machine libraries. Cross-platform profiling kits, live-link capabilities, and text-to-speech/speech-to-text systems were also popular suggestions.

## SUPPORT

» Using someone else's library, even if you have source code, can be really painful without documentation describing how it works and a solid support team standing behind it. Yes, you can read through every line of code, and you just may do that before your game ships, but support and docs give you a jumpstart and are especially helpful if you need to make modifications. They can make the biggest difference in whether a particular library is worth the price. So how strong do developers find current support for middleware libraries in general? (See Figure 11.) The answer is resoundingly mediocre (2.88/5.00). This number on its own is not terribly useful, so let's dive into this topic a bit deeper.

The two most common expectations in terms of support are developer forums and unlimited phone and email support. Some noted that developer forums are not the end all and be all ("Thank you for buying this car from me. Now go and consult with other car owners when your car breaks," one respondent analogized.) Forums can also create problems due to their openness: "We don't want to share secrets with other developers," said one of those we surveyed. Private, threaded conversation systems between the game team and the support team represent one useful alternative for delicate issues. It was noted by some responders that "unlimited" when it comes to phone/email support is only expected to be "within reason," but there was mixed reception to the idea of paying for enhanced support (3.06/5.00).

CONTINUED ON PAGE 12





# middleware showdown

CONTINUED FROM PAGE 11

One service provided by a few middleware vendors is on-site help integrating their solution with the team's game engine. But when developers were asked whether they liked the idea of having someone else integrate a library into their game engine, the answer was an overwhelming no—88.9% would rather do it themselves. [See Figure 12.] As one commenter noted, “[A] library you haven’t written [being] integrated into your code by someone else sounds like a bad recipe for shipping.” However, others noted that having support staff come on-site to help them integrate by pairing up with engineers or just being around to answer questions can be very useful. “That keeps them from hitting stonewalls or missing big opportunities because they don’t yet understand how the middleware is intended to work,” said one respondent.

We were very curious what developers think about groups of middleware libraries that are pre-integrated, or game engines that come with middleware bundled. Our game engine survey results seemed to imply that this would be a popular option. However, the response was actually mixed [2.99/5.00] and fairly flat across the range from “not interested” to “definitely interested.” [See Figure 13.] Several responders noted that the quality of the individual packages in a bundle can vary, creating challenges; another person noted that he’d been pushing the idea of a bundle himself with some of the major middleware vendors. But the holy grail was explained by one developer: “I’m more interested in having clean standard interfaces so that you can plug and play with any other middleware component in the market.”

When it comes to acquiring the source code to middleware libraries, there are a lot of differing opinions. Programmers usually want it, but many middleware companies are reluctant to share all their algorithms. How do our surveyed developers feel?

Overwhelmingly, the survey responders want source code. [See Figure 14.] A plurality said that they expect it as part of their base license, while others are willing to pay a little extra to get it, but a grand majority wanted source code in some fashion. There were a lot of comments about this, most along these lines: “The primary reason is for debugging purposes and as a hedge against lackluster documentation or support.” It was also noted by the responders that the need for source code varies depending on the type of middleware. For component systems such as video playback, a binary-only version can be fine. But, “above a certain complexity, lack of source is usually a deal-breaker.” One developer noted, “Not having the source code for [xxx] is currently biting us in the ass, so I’m biased today.”

With such a strong desire for source code, how do these same developers feel about using open source libraries? Most of them are very interested, but “due to licensing issues, we can only use them in our tools [i.e. things that don’t ship with the game].” At many companies, using open source libraries in the game can cause legal turmoil. With so many varieties of open source licenses, it can be quite confusing, and many responders

## Figure 10 popular libraries

### PHYSICS

NVIDIA PHYSX	26.8%
HAVOK PHYSICS	22.7%
BULLET	10.3%
OPEN DYNAMICS ENGINE (ODE)	4.1%

### NETWORKING

GAMESPY TECHNOLOGY	7.2%
QUAZAL (ANY PRODUCT)	7.2%
DEMONWARE	6.2%
RAKNET	4.1%

### AUDIO

FIRELIGHT FMOD	29.2%
OGG VORBIS	21.9%
AUDIOKINETIC WWISE	16.7%
OPENAL	15.6%
RAD GAME TOOLS MILES SOUND SYSTEM	7.3%
FONIX VOICEIN	3.1%
CRI AUDIO	2.1%

### ARTIFICIAL INTELLIGENCE

AUTODESK KYNAPSE	14.3%
BABELFLUX NAVPOWER	3.1%
PATHENGINE	2%
AILIVE LIVEMOVE	1%
HAVOK AI	1%
PRESAGIS AI.IMPLANT	1%

### VIDEO

RAD GAME TOOLS BINK	44.3%
CRI MOVIE	6.2%

### ANIMATION

OC3 FACEFX	14.4%
HAVOK ANIMATION	9.3%
NATURAL MOTION MORPHEME	8.2%
ANNOSOFT LIPSYNC	8.2%
RAD GAME TOOLS GRANNY 3D	6.2%
HAVOK BEHAVIOR	4.1%
AUTODESK HUMANIK	2.1%
EMOTION FX	1%

### RENDERING

SPEEDTREE	15.3%
ILLUMINATE LABS BEAST	10.2%
UMBRA	4.1%
FORK PARTICLE	2%
ALLEGORITHMIC PROFX / SUBSTANCE AIR	1%

### USER INTERFACE

SCALEFORM GFX	35.4%
---------------	-------

### OTHER

ZLIB	42.7%
LUA	36.5%
PYTHON	29.2%
BOOST	13.5%
SDL	3.1%



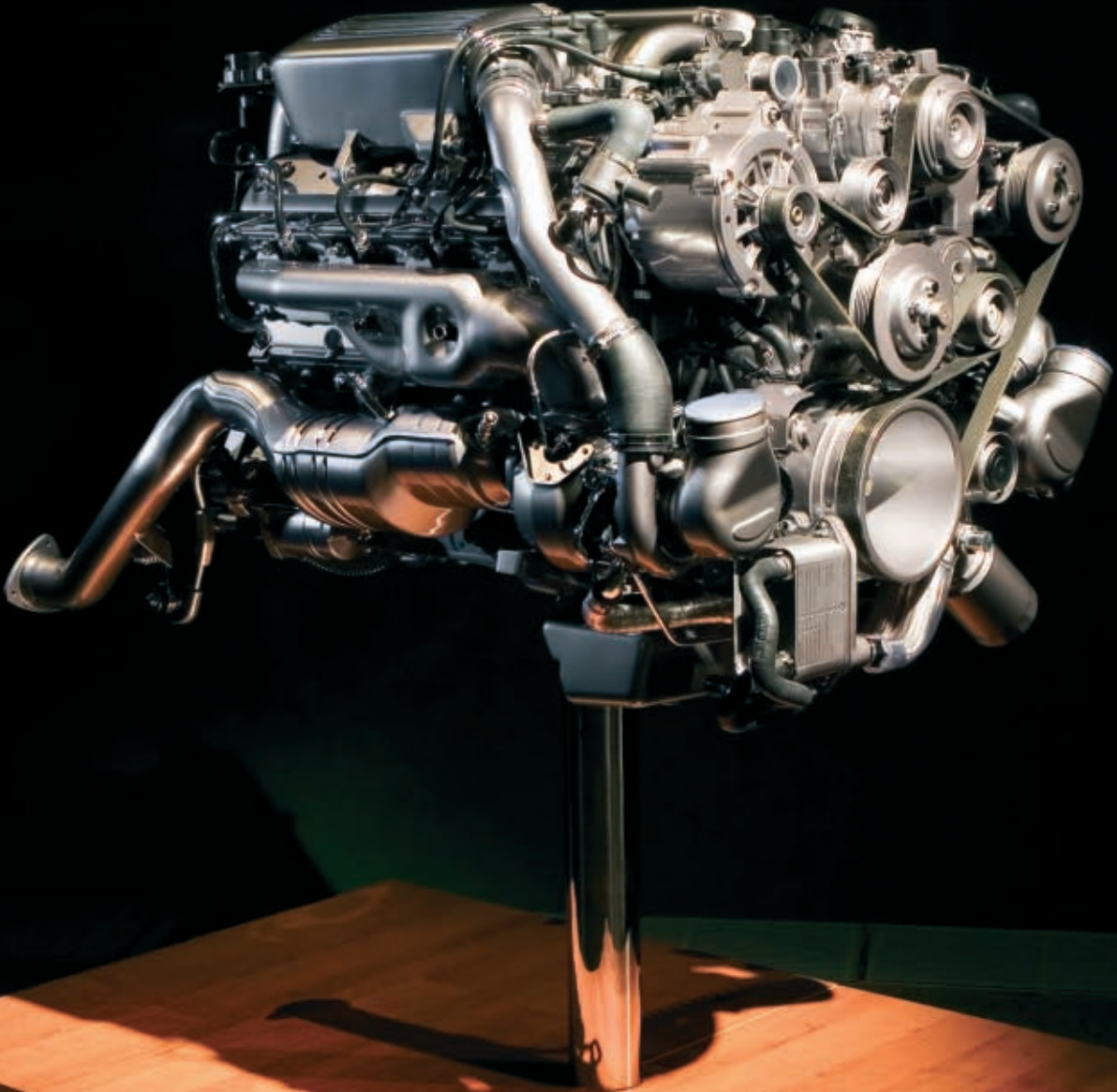
NavPower provided TOMB RAIDER: UNDERWORLD with AI pathfinding.





# ACHIEVE YOUR VISION

CRYENGINE – the maximum game development solution



XBOX 360

PLAYSTATION 3

MMO

PC

**NEXT  
GEN  
ready**

## CRYENGINE® 3

**CRYTEK®**

[www.cryengine3.com](http://www.cryengine3.com)

© 2009 Crytek GmbH. Crytek and CryENGINE are registered trademarks or trademarks of Crytek GmbH in the USA, Germany and/or other countries. All rights reserved. All other trademarks are the property of their respective owners.





# middleware showdown

## Figure 11 support

WHAT SUPPORT DO YOU EXPECT AS PART OF YOUR LICENSE?

DEVELOPER FORUMS	90.8%
UNLIMITED PHONE/EMAIL SUPPORT	75.9%
OCCASIONAL ON-SITE CHECK-UP VISITS	31.0%
FIXED AMOUNT OF PHONE/EMAIL SUPPORT	20.7%
SUPPORT STAFF SENT TO HELP INTEGRATE	19.5%

## Figure 12 integration

WHEN INTEGRATING MIDDLEWARE INTO YOUR GAME ENGINE, WOULD YOU RATHER:

DO YOUR OWN INTEGRATION	88.9%
HAVE THE MIDDLEWARE COMPANY DO INTEGRATION	11.1%

## Figure 13 bundles

HOW INTERESTED WOULD YOU BE IN MIDDLEWARE BUNDLES, WHERE YOU CAN PURCHASE SETS OF LIBRARIES PRE-INTEGRATED, OR INTEGRATED WITH A GAME ENGINE?

1—LEAST INTERESTED	20.0%
2	18.9%
3	22.2%
4	20.0%
5—MOST INTERESTED	18.9%

## Figure 14 source

HOW DO YOU FEEL ABOUT SOURCE CODE FOR MIDDLEWARE, IN GENERAL?

I MUST GET SOURCE CODE AS PART OF THE BASE LICENSE	52.9%
I AM WILLING TO PAY EXTRA FOR SOURCE CODE ACCESS FOR SOME LIBRARIES	35.6%
I AM HAPPY TO USE BINARY-ONLY LIBRARIES	11.5%

## Figure 15 open source

OPEN SOURCE LIBRARIES: DO YOU USE THEM?

YES, AS MUCH AS POSSIBLE	14.3%
YES, WITH CAUTION	56.0%
WOULD LIKE TO, BUT AM CONCERNED ABOUT LICENSES	27.5%
NO, NOT INTERESTED	2.2%

## Figure 16 frequency

HOW OFTEN DO YOU WANT LIBRARY UPDATES?

OCCASIONAL (QUARTERLY) MAJOR-RELEASE UPDATES	76.5%
ACCESS TO THE LIVE CODE BASE SO YOU CAN UPDATE AT WILL	38.8%
FREQUENT (WEEKLY) MINOR-RELEASE UPDATES	7.1%



called out software that is under the GPL as an absolute no-no. (See Figure 15.) Fortunately there are still numerous “truly free” open source middleware libraries out there. But knowing which is which (and educating one’s legal department) can be painful. Further, choosing and using one properly can be challenging as well; as one developer noted, they are “seldom shippable quality.”

The last question we had about support was regarding frequency of library updates. (See Figure 16.) What is ideal: quarterly updates? Weekly? Live access to the source code control system? The ideal scenario definitely varies based on circumstance, but there was an overwhelming preference toward simply receiving major updates. “We usually buy middleware for the features it already has,” one developer noted. Developers commented on the incredible pain of updating: “Anything more frequent than monthly is a waste—I can’t spare the bandwidth to integrate a new version that frequently.” However, when middleware is being ported over to a new platform, or new features are in development and may be buggy, more frequent updates may be necessary, despite how painful it can be to integrate them. Over a third of responders expressed a desire for live access to the library source code, regardless of how frequently formal updates are released.

## POPULAR CHOICES

» Through the comments and stories shared via the survey, we’ve learned a lot about companies that are easy to work with, and provide software that is designed well and easy to integrate. The company that was called out most frequently by developers for good quality software was RAD Game Tools, in particular the video software Bink. It is “well priced,” and “works well,” and RAD offers “great support.” Havok too was frequently mentioned as “a pleasure to work with,” and there were repeated compliments for the company’s support and docs, with one developer calling Havok “the gold standard in support.” Audiokinetic’s Wwise was also consistently called out as being “wonderful, awesome.”

There were a lot of horror stories about middleware use, reinforcing how important it is to test both the software and the support infrastructure before licensing middleware. The most common complaints centered around new platform launches, new library launches, middleware companies going out of business or being purchased, slow technical support, or unstable and buggy code bases. Several very popular pieces of middleware were called out repeatedly in the survey as having buggy code bases and horrible support, and yet people are still using their libraries due to the price. The rule of the day is definitely “try before you buy” (and get the source code).

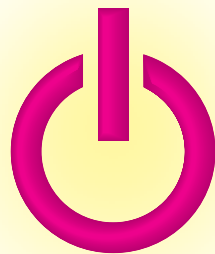
The developers who selflessly responded to our survey pointed out that middleware companies “try to position themselves as a magic bullet for all your problems.” But that in general, middleware “gets us 80% of the way there.” It may not solve all your problems, but it’s another great arrow in your quiver. The keys are to evaluate the available options intelligently, choose what you license wisely, and to work closely with your vendor to integrate it as painlessly as possible. If you do that, you’ll be way ahead of the game! 🎯

**MARK DELOURA** is a video game technology consultant residing in San Francisco, California. He has held leadership roles at Sony, Nintendo, and Ubisoft, and was at one time editor-in-chief of Game Developer magazine. Email him at [mdeloura@gdmag.com](mailto:mdeloura@gdmag.com).





**Game, it's so energetic!**



# TOKYO **GAME SHOW** 2009

SEPT **9** **24** **25** **26** **27**  
THU FRI SAT SUN

Business Day

Open to the Public

<http://tgs.cesa.or.jp/english/>

Venue: Makuhari Messe

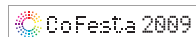
Organizer: Computer Entertainment Supplier's Association (CESA)

Co-organizer: Nikkei Business Publications, Inc. (Nikkei BP)

Supporter: Ministry of Economy, Trade and Industry (METI)



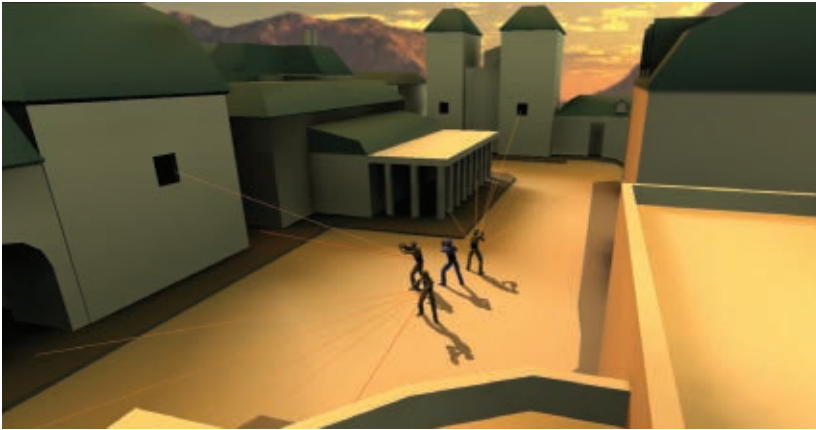
Nikkei Business Publications, Inc.





# Autodesk Games Insight

The latest scoop from Autodesk Media & Entertainment



“Our vision is for a unified workflow where your art tools work harmoniously with run-time technology, so that getting to the game is not only quick and painless, but a creatively rewarding process.”

— Marc Stevens  
Vice-President, Autodesk Games  
Media & Entertainment

Image: Autodesk® Kynapse® helps give characters a highly cognitive understanding of surrounding environments.

Hello and welcome to Autodesk Games Insight, our monthly column on Autodesk in the games industry. In this issue, I'd like to introduce the new Autodesk Games group and highlight our middleware solutions Autodesk® Kynapse® and Autodesk® HumanIK® middleware.

## Autodesk Games – Focused on games

Autodesk helps serve the needs of customers worldwide and recently realigned the company with dedicated teams for each industry. With this, the new Autodesk Games group was born. Autodesk Games is a team focused entirely on serving the needs of the games industry through cutting edge middleware technology and powerful game art tools.

Developers trust Autodesk for its 3D packages: Autodesk® Maya®, Autodesk® 3ds Max® and Autodesk® Softimage® software. We expanded on our games offering with middleware, bringing reliable technology to programming teams. Autodesk® Kynapse®, a leading artificial intelligence middleware solution, became part of the Autodesk family with the acquisition of Kynogon. This past GDC, we also released a new version of Autodesk® HumanIK®, a middleware solution for creating more believable run-time character animation.

## Character-Centric Middleware

Game teams are pushing the boundaries of putting compelling characters in open worlds to tell stories and create new experiences. This presents numerous

problems. Art production requirements such as modeling, sculpting, texturing, shading and animation have increased but this is only one part of the problem. Game teams need run-time technology that is able to bring characters to life in the game through realistic animation performances and decision making capabilities. Autodesk wants to bridge the gap between art and science, and make it much easier for people to create amazing, interactive character performances without technology getting in the way. Our vision is for a unified workflow where your art tools work harmoniously with run-time technology, so that getting to the game is not only quick and painless, but a creatively rewarding process.

## Kynapse – Put the brain in your game

Autodesk® Kynapse® is an artificial intelligence (AI) middleware solution used in over 80 high-end gaming titles. Kynapse handles dynamic 3D pathfinding, 3D spatial awareness and team coordination. What differentiates Kynapse from other AI middleware is that it helps give characters a highly cognitive understanding of surrounding environments and the ability to interact with it in a dynamic way. Its tool chain is also unique in that it is highly automated and flexible for iterative changes, which means that production teams can work more efficiently.

## HumanIK – Realistic character animation performances

While Kynapse helps give in-game characters artificial intelligence,

Autodesk® HumanIK® helps characters move in a realistic way and interact with the environment. Realistic animation performances that would have been extremely difficult through traditional keyframe animation, baking and blending techniques are now easier to achieve with HumanIK. With its procedural motion adaptation technology, characters adapt their motion at run-time responding to the user or the environment. Not only does HumanIK enhance your team's animation in real-time, it also helps solve the problem of animation data complexity. Production teams can focus on key performances with HumanIK working procedurally at run-time, reducing the number of animations necessary for believable results. Animators can now focus on their characters rather than mundane technical work, building compelling stories for the game.

## Conclusion

We are excited about the new Autodesk Games group, supporting developers with 3D software and robust middleware that you can trust. If you have any questions or would like to arrange for an evaluation of our middleware technology, we'd love to hear from you. Please contact us at [middleware@autodesk.com](mailto:middleware@autodesk.com) or visit [www.autodesk.com/games](http://www.autodesk.com/games).



Marc Stevens  
Vice-President  
Autodesk Games,  
Media & Entertainment  
[marc.stevens@autodesk.com](mailto:marc.stevens@autodesk.com)

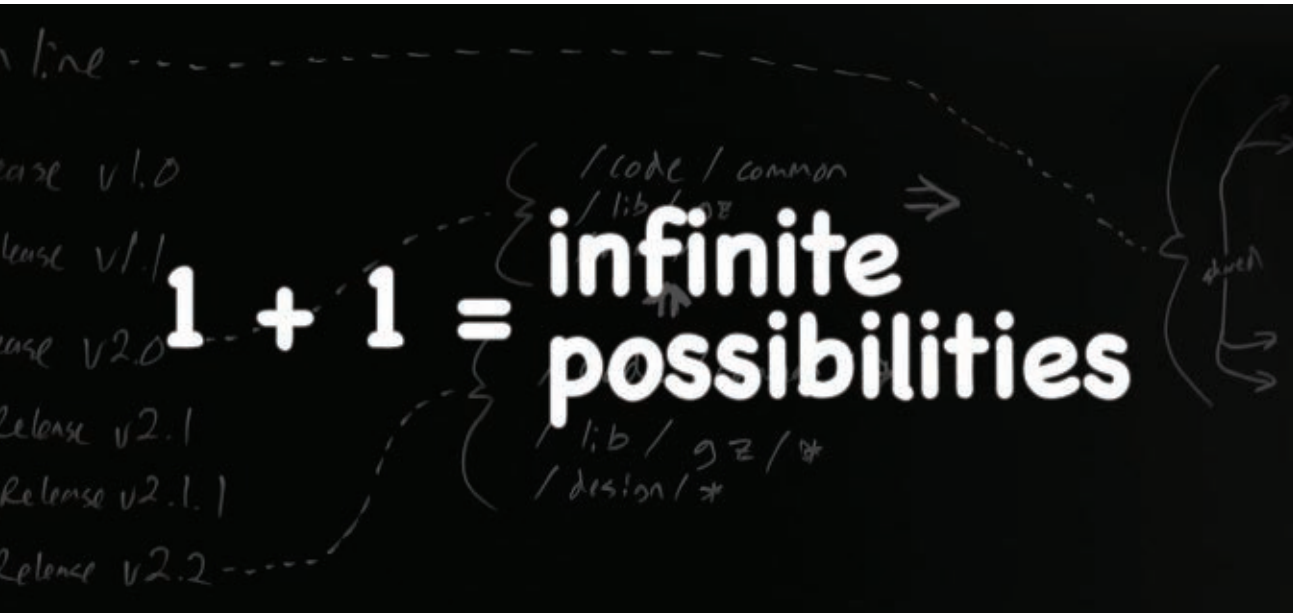
# Autodesk®

Autodesk, HumanIK, and Kynapse are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and/or other countries. All other brand names, product names, or trademarks belong to their respective holders. Autodesk reserves the right to alter product offerings and specifications at any time without notice, and is not responsible for typographical or graphical errors that may appear in this document. © 2009 Autodesk, Inc. All rights reserved.



[www.seapine.com/gamescm](http://www.seapine.com/gamescm)

Satisfy your quality obsession.



© 2009 Seapine Software, Inc. All rights reserved.

## Get More Change Management with Seapine Integrated SCCM

**TestTrack Pro + Surround SCM = infinite SCCM possibilities.** Seapine's integrated software change and configuration management (SCCM) tools do much more than competing tools, and at a much lower price point. Start with TestTrack Pro for change management and add Surround SCM for configuration management—two award-winning tools that together give you the best integrated SCCM solution on the market.

- Link issues, change requests, and other work items with source code changes.
- Manage simple or complex change processes with flexible branching and labeling.
- Coordinate distributed development with RSS feeds, email conversation tracking, caching proxy servers, change notifications, 3-way diff/merge, and other collaboration features.
- Enforce and automate processes with incredibly flexible work item and file-level workflows.

Built on industry-standard RDBMSs, Seapine's SCCM tools are more scalable, give you more workflow options, and provide more security and traceability than competing solutions.

Get more, do more with Seapine tools. Visit [www.seapine.com/gamescm](http://www.seapine.com/gamescm).

 Seapine Software™

**QA Wizard® Pro**  
Automated Testing



**Seapine CM®**  
Change Management



**Surround SCM®**  
Configuration Management



**TestTrack® Studio**  
Test Planning & Tracking



**TestTrack® TCM**  
Test Case Management



**TestTrack® Pro**  
Issue Management







# damage ar

## BANG! ARE YOU DEAD?

**ON A SATURDAY AFTERNOON IN THE PARK, TWO BOYS ARE PLAYING COPS** and robbers, chasing each other and shooting their toy guns. One boy aims his gun at the other boy, pulls the trigger and yells, "BANG! You're dead!" The other boy continues running and shooting his gun, and yells back, "No, you missed me!"

It's the age-old dilemma in shooting games; how do you determine if someone is actually hit? While apparent in a physical game with toy guns, it becomes more difficult in the online world of multiplayer games. In the physical game of cops and robbers, fair play is based on the core mechanics of the game and the arbiter – friendship. An arbiter is a person or entity appointed, or chosen, by parties to referee or judge a dispute between them where the parties may disagree on the correct outcome.

In the online world of gaming, the arbiter is a computer, console, or system in the network that can change depending on the game action. This arbiter must determine the resolution through the process of Damage Arbitration.

Damage arbitration decides which player is awarded the score of an attack, along with the actual calculation or point for the attack. The game design dictates the complexity of the arbitration system required; however, multiple factors are considered in the arbiter choice: player perception, arbitration functions, calculation rule sets, and the management of cheaters.

This article discusses the key elements of the damage arbitration process in online games to help you determine the arbitration system best suited for your product.

### PLAYER PERCEPTION

» A fundamental part of online games is the management of player perception. Player perception is game-specific, since every game has a different set of visuals and game mechanics. These define how the player interacts with the game world. Each player is separated by the network, and has a different visual representation of the game world.

Methods such as dead reckoning help predict where players are in the game and compensate for network latency on an object. Objects with motion, such as vehicles, are indicated by location and some motion data, such as velocity and instrumentation data. Motion data is used to estimate the current location of the vehicle in the game, and the instrumentation data determines how the player is controlling or maneuvering the vehicle, for example, doing barrel rolls or drifting. These components combine to create the impression of consistency within gameplay.

Maintaining the illusion of an identical game world amongst players helps to sustain a sense of fair play. An arbiter is necessary to include the player's interactions into the illusion of fair play. Players want to know they are receiving points for any damage they incur or instant kills they cause on a target.

### ARBITER CHOICE

» The choice of an arbiter is based on the game design, the network topology used for the game, and the method for managing cheaters. There are four main archetypes when choosing an arbiter: the attacker, the target, the server, and a pre-determined player.

### ATTACKER

» This archetype provides the best illusion of success to the attacker without the effects of latency. The target immediately receives the damage request upon the attack, and the player assumes credit for the kill. Multiple attackers shooting at the same target may create difficulties in determining who gets credit for any potential kills.

Figure 1 displays an example of this configuration. A1 and A2 are the attackers and T0 is the target. Both A1 and A2 damage the target T0, and assume the kill. Each attacker sends a kill message to T0 with the expectation of receiving credit for the kill.

A risk of using this approach is that one client system is determining the outcome of the damage. A trusted client system, one in which a user cannot install their own custom software, must be used with this configuration to prevent cheating. Once the installation of custom software becomes a possibility, the likelihood of cheaters appearing in the game increases.

### TARGET

» This archetype provides the best sense of fair play to the role of target by using the target's perception to determine how to score an attack. A central handler provides score arbitration for all damage requests for the same target. Each target must determine its own damage or the amount of damage to award. Calculations compensate for inconsistencies between the data provided by the attacker and the perception of the target, such as who attacked first.

Figure 2 displays the configuration of the target as the arbiter. A1 and A2 are the attackers, and T0 is the target. Both A1 and A2 send a damage





# Arbitration

request. The request from A1 is received first and kills the target. As the arbiter, T0 records the damage, determines the kill, and scores A1 accordingly. When A2 sends its request, T0 responds with a failure response since the target is already killed.

The target serves as a good choice for most games. Both the attacker and the target are involved and handshaking is used to detect cheating. However, a cheater could modify his game client to reject all damage requests as the arbiter, and create an unfair advantage. Player-kick or player-ban systems can help mitigate cheaters.

## SERVER

» Using the server as the arbiter is well-suited for games with a centralized server performing physics calculations and game logic. Servers offer time-stamped positional data, which can be used to rewind the state of the game to accurately score players for their kills.

This configuration is complex and targets may experience latency, but it provides the best defense against cheaters. Generally, cheaters do not have access to the server, which prevents them from manipulating the game.

Figure 3 illustrates the server as the arbiter. A1 and A2 are the attackers, and T0 is the target. A1 sends a damage request, which is received first. The server processes the request to determine the kill and sends a kill message to T0. A1 is awarded the point. When the damage request from A2 is received, the server already knows T0 is killed, and sends a failure response back to A2.

## PRE-DETERMINED PLAYER

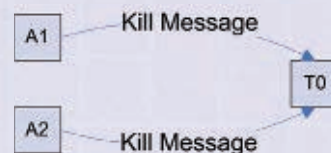
» The use of a single, pre-determined player as the arbiter is best described as a hybrid configuration between using the target and the server. This configuration provides a tradeoff between cheater mitigation and latency under most network topologies; however, inconsistencies in data associated with the server may occur.

Games using a player system as the host, which also designates the damage arbiter as the host, benefit from this selection. Players have more control over their gaming experience and can either ignore the designated host, or select a different host if they suspect the host is cheating. Player-kick or player-ban systems can provide enough cheater management to maintain a sense of fair play.

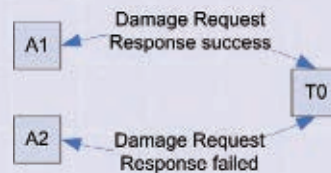
## CALCULATIONS

» The selected arbiter uses a rule set that includes a variety of calculations defined by the game mechanics to calculate the amount of damage. The rules used determine whether the request from the attack is correct, assigns damage, informs the other attackers, and then scores the attack. The choice of arbiter does not affect the calculation process, but it may affect how the results are perceived.

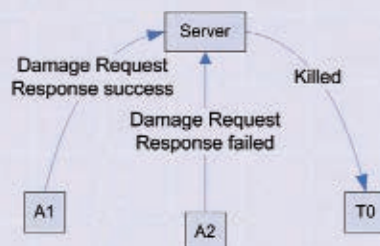
In calculating the amount of damage, the arbiter performs a line-of-sight analysis between the attacker and the target to determine the accuracy of the attack. Game mechanics, such



**FIGURE 1** Attacker as Arbiter. Both attackers (A1 and A2) send a kill message to the target (T0); each expecting credit for the kill.

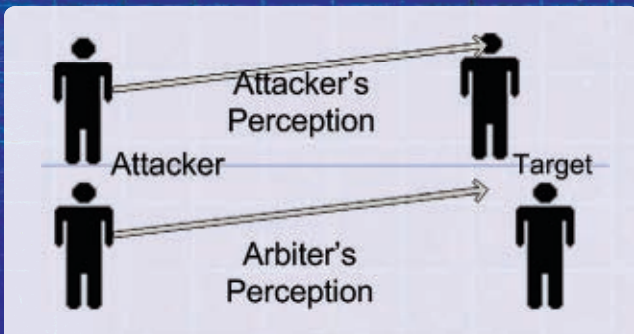


**FIGURE 2** Target as Arbiter. Both attackers (A1 and A2) send a damage request to the target (T0). As the arbiter, T0 records the damage, determines the kill, and credits A1 as the first to hit the target. T0 sends a failure response to A2.



**FIGURE 3** Server as Arbiter. Both attackers (A1 and A2) send a damage request, with A1's request received first. The server processes the request to determine the kill of the target (T0) and sends a success response to A1. Knowing T0 was killed, the server sends a failure response to A2.





**FIGURE 4** Damage Calculations. The perceived hit location of the attacker and the arbiter can be different. The attacker displays the perception of a scored headshot while the arbiter displays the perception of a missed hit. The arbiter can score the shot as a kill, or reduce the estimated damage instead of scoring a kill.

as whether a player can shoot through walls in a single-player mode, are considered to ensure both the attacker and the target perceive the result as fair.

Hit location of a target may also have an impact on the amount of damage awarded, generally to the advantage of the arbiter. The amount of damage to award becomes difficult if the attacker perceived the attack as a headshot, but the arbiter calculated it as a miss. The arbiter can use the difference between the attacker's perceived target hit location and the actual hit location. The arbiter may compromise and reduce the amount awarded based on how far off the attacker's request was from the hit calculations.

Figure 4 displays the perceived hit location of both the attacker and the arbiter. The attacker displays the perception of a scored headshot; the arbiter displays the perception of a missed hit. The arbiter can score the shot as a kill, or compromise the amount of the shot as doing significant damage, but not as an instant kill.

Choosing the server as the arbiter enables the use of additional resources to store historical time-stamped data for calculating the results. The arbiter can perform a rollback on the game environment to the time of the attack and complete the calculations using the snapshot of the past. This provides the best results of perceived fairness.

## ANTI-CHEAT

» The goal of a cheater is to disrupt fair play, and impose his or her own sense of play. Cheaters find holes in the geometry of the game and ways to bend the physics. They use devices to manipulate network traffic, and even change the game code. Depending on the arbiter, cheaters can compromise the arbiter system and find ways around the detection methods.

Multiplayer online games must have a strategy with which to manage cheaters. A check of the link state of the network device before sending damage arbitration requests serves as a simple method.

Sending messages at regular intervals is a more complex method, but an effective one. The messages are considered heartbeat messages. When the arbiter receives a damage request, it checks the rhythm of the attacker's heartbeat by calculating the running standard deviation of the time between received heartbeats. The rhythm is abnormal if the newest time between heartbeats varies by more than two standard deviations. If the rhythm is abnormal, the arbiter can reject the request or reduce the amount to award. Latencies of the Internet may interfere with the rhythm of the heartbeat, creating false abnormalities, but validation of the data can assist. A time stamp on the damage request allows the arbiter to adjust the damage accordingly.

## EXAMPLE

» In a hypothetical online game, the player's weapon is a water pistol containing a specific color of water. The game designers have determined the attacker will perform the initial hit calculation, and the target will perform the damage calculation. The game defines six hit sections: two arms, two legs, a chest, and a head. Damage is defined by the size of the area of wetness on a particular section of clothing, with headshots causing temporary blindness in varying degrees.

When a player fires the water pistol, his or her console uses standard collision detection algorithms to determine if the stream of water projected as a ray from the water pistol connects with the target player's character model. If the attacking player's console determines a hit, the game sends a damage arbitration request to the target. The request contains the perceived hit location as defined by the collision detection, the originator of the attack, and potentially other game-specific data, in this case the color of the water.

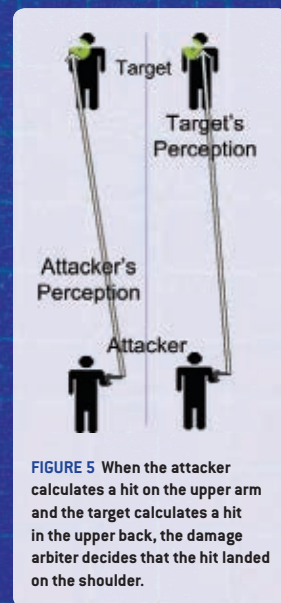
The target player's console then acts as the damage arbiter. It uses the information stored in the request to duplicate the original collision detection. It is unlikely for the two consoles to calculate the same hit locations; it is expected that the two results will differ at least fractionally. If the arbiter agrees on the same hit section, then the damage can be determined as normal. In the case of misses, the damage arbitration should contain calculations for the distance of the attacker's ray to the target model. If the target disagrees greatly, then the design of this game dictates that the hit only results in a few drops of water appearing on the section hit.

To further this example, assume the attacker's console calculates the hit in the upper arm and the target calculates the hit in the upper back, as shown in Figure 5. The damage arbiter, the target player's console, decides the target was hit in the shoulder. It also decides this was a direct hit and creates a sizeable wet area.

In another example, assume the attacker calculates the hit in the lower leg, and the target's console calculates the attack as a miss. The damage arbiter, the target player's console, decides this means the target was only splashed in the lower leg. The hit only scores a few drops on the lower leg.

## SOLUTION

» The overall goal is to have the highest number of players perceive a game as fair. Different arbiters can be used to complete every step of the damage arbitration processes; however, it is important to mix arbiters cautiously. Improperly managing player perception can lead to both sides feeling as if they have been cheated. Using a mixed configuration of the attacker for hit arbitration and the target for damage and death arbitration provides the quickest perception of events while managing the difference in perception for most situations. 🎮



**FIGURE 5** When the attacker calculates a hit on the upper arm and the target calculates a hit in the upper back, the damage arbiter decides that the hit landed on the shoulder.

*The author wishes to thank Rhonda Salvestrini for editing help.*

**RONALD ROY** is a staff game integration engineer at Sony Computer Entertainment America. Email him at [rroy@gdmag.com](mailto:rroy@gdmag.com).



# Look Who's #1



## Top Products for the Top Games



Award Winning Products & Services.  
Over 250 Shipped Games.  
Cross Platform Optimized.

The Proven Solution.

**havok™**



PHYSICS



ANIMATION



BEHAVIOR



CLOTH



DESTRUCTION



AI





# Unreal Technology News

by Mark Rein, Epic Games, Inc.

Canadian-born Mark Rein is vice president and co-founder of Epic Games based in Cary, North Carolina.

Epic's Unreal Engine 3 won Game Developer magazine's Best Engine Front Line Award for three consecutive years, and it is also the current Hall of Fame inductee.

Epic's internally developed titles include the 2006 Game of the Year "Gears of War" for Xbox 360 and PC; "Unreal Tournament 3" for PC, PlayStation 3 and Xbox 360; and "Gears of War 2" for Xbox 360.

## Upcoming Epic Attended Events:

### SIGGRAPH

New Orleans, LA  
August 4-6, 2009

### GDC Europe

Cologne, Germany  
August 17-19, 2009

Please email:  
mrein@epicgames.com  
for appointments.



## SOFTMAX AND NAMCO BANDAI BUILD FANTASY RPG MAGNACARTA 2 WITH UNREAL ENGINE 3

Korean developer Softmax first gained experience using the Unreal Engine to develop its 2005 PlayStation 2 and PSP role-playing game, *Magnacarta: Tears of Blood*, which was powered by Unreal Engine 2.

This time around, a team of 40 at Softmax partnered with Namco Bandai to create *Magnacarta 2* for Xbox 360 using the latest Unreal Engine 3 technology.

"There were some big improvements added to Unreal Engine 3 regarding toolsets and rendering abilities," said Yoshihisa Kanesaka, producer, *Magnacarta 2*.

"We frequently consulted the Unreal Developer Network during the early stages of the development process. Using the UDN archive search is very useful in getting immediate answers on everything imaginable."

On top of the processing power that Microsoft's Xbox 360 brought to the table, Kanesaka believes his team benefited from two key aspects of Unreal Engine 3.

"The Unreal Editor, which has been vastly improved from Unreal Engine 2, has always been a long-time merit of the software," said Kanesaka.

"The other advantage would be the engine's versatility. It would have been possible to develop an engine on our own if we wanted to, or to use a different middleware engine. But Unreal Engine 3 contains everything we needed to develop the game, and we thought it was useful since there is no risk in adding any other middleware to it."

From a gameplay perspective, Softmax was able to use UE3 to create a powerful loading system that allowed for the creation of huge environments. In addition, Kanesaka said they were able to create a new battle system that is both seamless and occurs in real-time.

All of the game's elaborate cut scenes were developed using Unreal Engine 3, which Kanesaka said saved Namco Bandai time and money.

"We also developed this system that blows away enemies with physical attacks using PhysX," added Kanesaka.

"Big RPG titles developed by Japanese publishers use fancy effects. That's the style that is currently popular, and we can easily develop these effects with Unreal Engine 3 by using its particle and material editors."

Kanesaka believes Unreal Engine 3 can bring great RPG stories like *Magnacarta 2* to life. He said one of the best features of UE3 from a development standpoint is its

excellent editing capabilities, which are mission critical for game engines today.



*Magnacarta 2* for Xbox 360

Thanks to a talented development team and publisher, and to the technology within UE3, Kanesaka hopes that *Magnacarta 2* shows critics that RPGs are, in fact, evolving. He said the team's goal is to create a new standard in excellence and presentation that sets the bar for what is considered a great RPG game.

"I think it is mandatory to use a middleware engine which minimizes the risks of developing next-gen console games, so I assume more developers in Japan will use the Unreal Engine," said Kanesaka.

Japanese studios that have shipped Unreal-powered titles include Feelplus, which licensed UE3 for its collaboration on *Lost Odyssey* for Xbox 360 with Mistrwalker. Square Enix also licensed UE3 to develop *The Last Remnant* for Xbox 360, PlayStation 3 and PC.

Last fall, Grasshopper Manufacture licensed UE3 for its new multiplatform action horror game directed by innovative game designer Suda51 and produced by legendary *Resident Evil* creator Shinji Mikami.

*Thanks to Namco Bandai for speaking with freelance reporter John Gaudiosi for this story, which will be posted in full at [www.unrealtechnology.com](http://www.unrealtechnology.com).*



For UE3 licensing inquiries email:  
[licensing@epicgames.com](mailto:licensing@epicgames.com)

For Epic job information visit:  
[www.epicgames.com/epic\\_jobs.html](http://www.epicgames.com/epic_jobs.html)

WWW.EPICGAMES.COM



# THE FEARFUL EYE

## design considerations for a cinematic camera

**MARION IS TAKING A SHOWER. CUT. WE SEE THE SHOWER head with water pouring out. Cut. Marion runs her hands through her hair. Cut. On the other side of the shower curtain we see a figure enter the room. Marion does not notice as the figure approaches. Cut. The silhouette rips the shower curtain back and raises a knife. Cut. Marion turns around. Cut. Marion sees the figure and screams. Cut. We see a close-up of her screaming mouth. Cut. The figure raises the blade. Cut. Marion is stabbed. Cut. The knife rises again. Cut. It cuts through the stream of water toward Marion's body. Cut. She is still screaming. Cut.**

70 shots in the span of 45 seconds comprise one of the most famous and frightening scenes in the history of cinema. The shower scene from *Psycho* was filmed in seven days in December of 1959, but the impression it made on the film industry is evident even today. The film relies heavily on innovative cinematography because there was not much else for the director to work with; *Psycho* was shot on a shoestring budget and was designed not to rely on gore, special effects, or even color film. Yet to this day it is considered one of Hitchcock's best, and one of the best films of all time.

*Psycho* and films like it have helped make cinematography a respected art form, and yet the art has not been widely adopted by video games. Though many modern games use some sort of a virtual camera, most game cameras are designed to be utilitarian: they must

show the player where he is going without getting snagged on collision surfaces, flying through walls, or revealing the boundaries of the game world. Games, especially third-person real-time 3D games, have it a lot rougher than film; while a director can spend days composing the perfect shot for any particular scene, a game camera must select an appropriate shot based on a dynamically changing game environment sixty times per second. Camera systems are notoriously difficult to program; it is no wonder that most games settle for a camera that is simply capable of looking in the right direction consistently. On top of that, the mapping between the player's controller and the direction that his avatar moves is often dependent on the angle of the camera, and so the difficulty in creating a camera system that produces both usable and artistic shots is compounded. Perhaps this is why the game industry does not yet have an equivalent to *Psycho*.

Despite these challenges, many games have sought to employ some degree of cinematography. A common (and effective) technique is to simply contort the design of a level so that a simple follow camera will end up placed somewhere interesting as the player moves forward (EA's *DEAD SPACE* does this very well, as do many first-person shooters). But there are also titles that have gone all out in their attempt to leverage key film techniques. Games like *RESIDENT EVIL*, *SILENT HILL*, and



# THE FEARFUL EYE



GOD OF WAR eschew many standard game mechanics in order to enable non-standard camera systems. The result has dramatic control and gameplay implications, but it also lends these games a certain power that other titles struggle to match. In this article I will discuss the history of these camera systems and how they affect the way that these games are played, especially as seen in horror games, arguably the game genre with the most effective cinematography.

## MAKING A CINEMATIC GAME

» I know what you are thinking. “Let’s make this game totally cinematic” sounds like something a clueless producer might say (it’s almost as bad as “totally epic”). But let us say that you really do want to make a cinematic game—you want to make a game that can use shot composition and camera movement to affect how the player feels. There are plenty of ways to go about this, of course, but for the purposes of this article I am going to concentrate on camera systems, specifically those for third-person games.

Before you make the decision to go with an experience that is “totally cinematic,” you should be aware of some of the trade-offs that a cinematic camera may require. Most 3D third-person games rely on what I will call a procedural camera, a camera that is programmed to dynamically move and respond to the state of the world. The classic third-person follow cam is a good example; this type of camera system is usually built upon heuristics that allow it to choose a worthwhile vantage point given arbitrary world geometry. Since FADE TO BLACK, TOMB RAIDER, and SUPER MARIO 64, the procedural camera has been the norm for the vast majority of 3D third-person games.

Procedural cameras are not perfect, however. In addition to being prone to calculation errors, they generally cannot cut and are difficult to control for dramatic effect. Developers often treat third-person follow cameras as an extension of their character’s eyes; though the camera has been pulled back away from the avatar, it is really a first-person viewpoint. Perhaps this is why we usually show our characters’ heads moving when the player pans the camera around.

On the other end of the spectrum is what I will call the hand-crafted camera. These are cameras that are placed by an artist or designer by hand rather than produced as the result of procedural heuristics. They may be fixed in place, tethered to splines, or allowed to move only in certain axes. Typically several different camera angles will be set up to cover a particular region in the world, with collision volumes placed in the space to trigger cuts from shot to shot. Games like RESIDENT EVIL and SILENT HILL have helped define this sort of camera, but they are now used by a variety of titles. In terms of cinematic effect, the hand-crafted camera is extremely powerful; it allows an artist to play the role of the cinematographer and populate the game with carefully designed shots. Even more importantly, these types of cameras can typically cut from one angle to the next, which enables meaningful sequencing of shots. The introduction of the Licker in RESIDENT

**Camera systems are notoriously difficult to program; it is no wonder that most games settle for a camera that is simply capable of looking in the right direction consistently.**

EVIL 2 (see sidebar, pg 28) is a good example of a well-designed hand-crafted camera sequence. By film standards it is a very basic foreshadowing sequence, but it is the kind of visual storytelling that is extremely difficult to produce with a procedural camera.

Though hand-crafted cameras can offer an advanced degree of control over the way a game is presented, they are also a whole lot of work. Not only does each shot need to be set up by hand, but collision volumes to control transitions from shot to shot are also necessary, and this task can be harder than it sounds. Perhaps as important are the gameplay and control restrictions that these types of camera systems often imply. I will return to that point in a minute.

Are hand-crafted cameras worth the effort? Of course, the answer to that question will differ with each title, but it should be clear that they are a



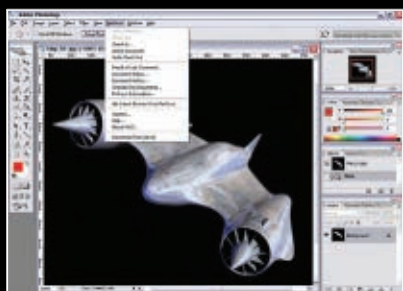


## Introducing P4GT, a productivity feature of Perforce SCM.

The Perforce Plug-in for Graphical Tools, P4GT, makes version control painless by seamlessly integrating Perforce with leading graphical tools. Drop-down menus allow access to Perforce from within 3ds Max, Maya, Softimage XSI, and Adobe Photoshop.

Art and development teams can standardize on Perforce to version and manage both source code and digital assets. Enhanced collaboration during the design process helps teams to work together in real time to release small patches or create whole new worlds.

P4GT is just one of the many productivity tools that comes with the Perforce SCM System.



P4GT

**PERFORCE**  
SOFTWARE

**Download a free copy of Perforce, no questions asked, from [www.perforce.com](http://www.perforce.com). Free technical support is available throughout your evaluation.**

# THE FEARFUL EYE

straightforward way to apply cinematographic techniques to games. Indeed, many games that are made into films are those that use hand-crafted cameras; RESIDENT EVIL, SILENT HILL, ALONE IN THE DARK, and the upcoming CLOCK TOWER film are all based on hand-crafted camera games. Perhaps this is because when filmmakers look at these games they are able to easily imagine shooting the same shots with live actors on real film (and indeed, the SILENT HILL film does match a number of shots and angles to the game in significant early scenes).

## CAMERA TYPES

» A game engine has a lot of potential camera behaviors at its disposal once it has committed to using a hand-crafted camera system. Giving a designer or artist free reign over a camera system that he or she can absolutely control opens the doors to all kinds of interesting and original scenes. Still, there is a small set of common camera behaviors that make up the majority of hand-crafted camera games. I will touch on a few of them here.

The “fixed camera” is exactly what it sounds like: a camera that is locked in place and cannot move. RESIDENT EVIL uses these types of cameras by necessity, as its level art is entirely pre-rendered. A variation is the “panning camera,” which remains affixed to a single spot but may rotate to track the player’s movement. Another common type is the “trucking camera,” which moves to follow the player but is restricted to a 2D plane. This sort of shot is often used to produce a wide, side-scrolling perspective. The first scene in SILENT HILL 2 uses an interesting trucking camera where the X axis of



Capcom's GameCube remake of RESIDENT EVIL

movement is tied to the player’s X position but the Y axis of movement is tied to the player’s Z position, causing the camera to sink toward the ground as the player approaches it. Finally, many hand-crafted camera games use a “spline camera” to control the motion of the camera through the scene with a spline. A further refinement is to use a second spline that defines the camera’s focal point so that the player does not always have to be in the center of the frame.

## CONTROL QUAGMIRE

» The huge advantage to hand-crafted cameras, at least from the cinematographer’s point of view, is that they can cut from one interesting shot to the next. Instead of being locked behind the player’s avatar, a hand-crafted camera system can select any angle with each shot, even one looking straight at the avatar’s front. However, from a game play and control perspective, this feature is also problematic. Most 3D third-person console games define “up” on the control stick as “forward” in the game world, which is a direction that is relative to the camera’s current angle. To put it another way, the mapping between the 2D analog stick and the 3D environment is usually performed by transforming the 2D input to match the view matrix so that the vector along which the camera is looking is also the direction about which control will be relative. So “forward” in most games depends on where the camera is looking

at any given time. Because procedural cameras are almost constantly on the move, the direction that the player moves when he presses “up” on the stick is also constantly changing. This method of control makes sense to players, probably because third-person follow cameras are so often just first-person cameras that have been pulled back.

But when the shot can cut without warning, as it does in most hand-crafted camera games, the traditional 2D-to-3D player control mapping suddenly ceases to work. If the player is running forward and the camera cuts to a shot that is perpendicular to the previous shot, the new definition of “forward” will suddenly change and the player’s avatar will turn 90 degrees even as the stick itself remains pushed up. If control is to be camera relative, then the camera cannot go changing its angle dramatically at unpredictable times. But unpredictable changes in camera angle are exactly what we need to produce interesting camera sequences like the one from RESIDENT EVIL 2, so the most logical conclusion is that the player’s controls can no longer be relative to the camera.

This is where the infamous RESIDENT EVIL “tank” control scheme comes from. In RESIDENT EVIL (and ALONE IN THE DARK), control is relative to the player avatar’s facing direction rather than the camera’s facing direction. Left and right rotate the character in place, and forward moves him in the direction he is currently facing. The advantage to this system is that the controls are thus decoupled from the angle of the camera, which makes it possible to run straight continuously regardless of how often or how dramatically the camera angle changes. The major drawback to this system is that it makes no intuitive sense; camera-relative controls are immediately understandable to most people, but character-relative control schemes have a steep learning curve.

Fortunately, there are several solutions to this problem. One uncommon but effective solution is to move the “forward” control off of the analog stick and put it on a button. In FATAL FRAME 2 moving forward can be accomplished using the analog stick, but the protagonist can also be made to run in a straight line by holding down the run button. The GameCube remake of the original RESIDENT EVIL takes this idea one step further with its “Type C” control scheme. In that case the deep analog trigger on the GameCube controller is used to drive the character forward (shallow push to walk, full click to run) and the control stick is only used for adjusting the character’s heading left and right. These types of control schemes feel a bit like a driving game, but they work well and remove most of the pain of character-centric controls.

An even better solution is the one introduced by Square’s PARASITE EVE. The developers of that game made a key observation: camera-centric control systems only fail when the camera cuts. Their solution was to tie changes in the character’s forward direction to the player’s input rather than to the camera. Whenever the control stick is released, it is immediately re-calibrated so that “up” becomes “forward” relative to the current view direction. As long as the player is moving, the definition of “forward” will not change even if the camera cuts. This allows a player to hold “up” on the control stick and run through any number of arbitrary camera angles, but still experience movement within any given shot as relative to the direction of the camera producing that shot. A couple of years after PARASITE EVE came out, Capcom used the same scheme for DEVIL MAY CRY, and it has remained the standard ever since.

In fact, a movement system that relies on static cameras is incredibly beneficial to a brawler like DEVIL MAY CRY. Combat in that game is designed to look smooth, fast, and sexy, and part of that equation is the way that the controls map directly to the current camera angle. When Dante decides to cut an enemy in half, he automatically rotates and swings exactly in the direction that the analog stick is pointing relative to the current camera shot. This method allows the player to target enemies simply and quickly with much more precision than would be possible if the camera were in constant motion. In fact, other brawlers that use moving procedural cameras often have to



invent schemes to allow the player to target his opponents without relying on precise analog input (consider the target mapping system in MARK OF KRI or the transition to target-relative movement in PRINCE OF PERSIA when the prince draws his sword). DEVIL MAY CRY's combination of PARASITE EVE-style sticky control calibration and precise combat mechanics has made it the basis for modern brawler game design.

#### OTHER CONSIDERATIONS

>> Let us assume that a hand-crafted camera system is right for your game, and you have chosen to implement the PARASITE EVE/DEVIL MAY CRY sticky-calibration input model. Here are a few things to keep in mind.

Being able to cut to arbitrary angles is a powerful feature, but it is probably best from a control perspective to never cut to a camera angle that is 180 degrees rotated from the previous camera angle. Even with sticky calibration controls, a 180-degree cut causes the player's controls to be exactly reversed. While running forward might not be a problem, left and right will be opposite what the player expects until they release the stick and the control mapping re-calibrates. 180 degree cuts are also disorienting; when moving down a hallway or path, a full half-circle rotation of the camera can confuse the player about which way he should be heading. If you have an open space that is covered by several different cameras, ensuring that this sort of 180-degree cut never happens can be a challenge (depending on the layout of the space, the player may be able to trigger cameras in a sequence that you did not expect). To combat this, games like DEVIL MAY CRY tend to place cameras in concentric circles out from the center of a space (thereby

forcing all nearby potential camera angles to be within 180 degrees of the current angle). Other games simply avoid open spaces that require more than one camera angle to represent; RESIDENT EVIL tends to box the player into small, rectangular spaces, while GOD OF WAR often pulls the camera back and lets it pan or truck to cover a space.

Another thing to consider is that most hand-crafted camera games do not allow the player to manipulate the camera at all. This is perhaps used to break the feeling that the camera is an extension of the avatar's eyes, but there is another, more practical reason as well. One major benefit to hand-crafted cameras is that it is possible to know exactly which parts of the game world will be visible. This information can be used to dramatically improve the visual quality of the scene; for example, geometry and lights in the level can be sorted and culled with respect to each camera angle offline, giving the CPU time to do other things at runtime. Games like RESIDENT EVIL: CODE VERONICA, GOD OF WAR, and SILENT HILL 3 are among the best-looking games for their host platforms because they take advantage of all kinds of performance tricks that hand-crafted cameras enable.

Finally, it is worth noting that hand-placed cameras give the player a very different sense of space than does a procedural camera. The boundaries of the current camera shot define the edges of the player's vision, and since hand-crafted cameras adjust their view direction much less frequently than procedural cameras, it is very easy to obscure important locations in space. Many of the games mentioned in this article go out of their way to build secondary and tertiary methods for communicating the existence of objects in the world that may be close by but not visible from the current shot.



**GDC**  
09  
**China**

**Game Developers Conference® China**  
**October 11–13, 2009**

Shanghai International Convention Center | Shanghai, China

Reach a unique vertical market of Chinese industry professionals in online games, game outsourcing, and mobile games at the Game Developers Conference® China's second developer event.

**Register with code: GDCCHINA09**  
and receive 10% off Attendee and VIP Passes\*

\*Discounts cannot be combined with other GDC China discounts or other promotions including group registrations and alumni discounts. Discount may be applied to Early Bird rates. GDC China reserves the right to remove and end this promotion before the end of online registration.

Supported by 

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

**THINK SERVICES**

## the state of the art

It is useful to consider a few examples from existing titles as reference when discussing the nuts and bolts of cinematography in games. Here are a few examples of complicated and interesting shots that have occurred in games over the last decade or so.



Tecmo's FATAL FRAME 2.

In the introductory sequence to Konami's *SILENT HILL*, the protagonist follows his daughter into a narrow, L-shaped alley. Initially the camera is positioned in the corner of the alley's bend and is looking at the character head on. As the character moves forward toward the camera, it lifts and twists in tandem with the player's motion so that the opposite side of the corner is revealed. By the time the character reaches the bend, the camera has moved upward and is now looking straight down at the top of his head. As the player

turns and heads down the corridor, the camera reverses its previous movement and keeps the player's back in the center of the frame. This motion is complex and unsettling, and it is a great lead-in to the next shot, in which suddenly everything has become unnaturally dark and *SILENT HILL*'s central fear mechanic is revealed.

In one of the first scenes in *DEVIL MAY CRY*, the protagonist runs up a hill and enters a castle though a hole in its outer wall. The shot for this is simple: the camera is pulled away

from the landscape and rotates in place to follow the character as he approaches the castle wall. What is nice about it is that once he enters, the camera does not immediately cut to show the inside of the castle. Instead, it automatically pans up to show the castle's wall and spire, establishing the location in which the player will spend most of the game.

Early in Capcom's *RESIDENT EVIL 2*, Leon, the protagonist, enters a deserted police station. As he travels through one of the outer rooms we are treated to a shot of a large picture window looking out into the dark. As he passes the window we see something—just some arms and legs and maybe a tail—scuttle past the window and out of sight. Leon doesn't notice the movement, but we do. He opens the door directly in front of the window and enters a long hall. Immediately

upon passing into the hall the shot changes to one on the outside of the building, looking in at Leon through a large window. The implication is clear: whatever it is that we saw moving outside is now looking in. After taking a few steps the camera cuts again to give a better view of the hallway, and a few steps after that the creature in question—a new monster called the Licker—is introduced.

The *GOD OF WAR* series is chock-full of dramatic action sequences that rely on complex maneuvering of a real-time camera. A good example is the early colossus fight in *GOD OF WAR 2*, in which the camera deftly swings from behind the protagonist to a perpendicular angle, revealing a pane of windows with the colossus lurking outside. The camera then leads the player through a window and down into the main

combat area. In the course of fighting the colossus the protagonist launches himself at the giant statue and the camera follows a pre-determined path that highlights the action.

The introductory sequence to *FATAL FRAME 2* consists of the protagonist running through a forest toward a deserted village. This is a simple sequence—the player can only follow a single, narrow path and there is no interactivity on the way—but it is made interesting by the way that the sequence is shot. At one particularly nice point, the camera is placed very low to the ground and pointed up at the protagonist, and as she rounds a curve it hugs the inner wall of the path. This kind of shot conveys a sense of oppressiveness; we can see sun through the top of the forest canopy high above, but down at the ground level things are dark, claustrophobic, and foreboding.


*SILENT HILL* causes its protagonists' heads to turn toward interesting objects, *FATAL FRAME* provides a first-person mode for looking around and combat, and in *RESIDENT EVIL* the characters automatically aim at the closest enemy. Ranged weapons are particularly problematic, as it is almost impossible (and not very fun) to shoot things that are off the side of the screen.

There is a high concentration of horror games that employ hand-crafted cameras; perhaps this is because shot composition is particularly valuable for building tension. The other game genres that often use hand-crafted cameras are the adventure genre (*THE LONGEST JOURNEY*, *BROKEN SWORD*, etc), upon which *RESIDENT EVIL* and its progenitor *ALONE IN THE DARK* are based, and the modern brawler genre (*GOD OF WAR*, *NINJA GAIDEN*, etc) as defined by *DEVIL MAY CRY*, which itself began life as a *RESIDENT EVIL* game.

### HAND TO MOUTH

» In this article I have focused almost exclusively on cameras that are set up and controlled by hand, as the best cinematography in games today comes from titles that were built that way. But there are other ways to achieve similar effects. Cing's *TRACE MEMORY* uses the DS's top screen to show dramatically-

composed stills of the area that the player is exploring while the bottom screen renders the same environment in real time using a basic top-down perspective camera. *ICO* uses a combination of procedural cameras and hand-crafted cameras that switch based on the player's motion to produce interesting shots without cutting. And the most recent *ALONE IN THE DARK* uses procedural and hand-crafted cameras interchangeably to separate areas that are mainly traversal-oriented and areas that need to be dramatically shot.

Making a cinematic game is not a simple task; it requires pulling together every discipline in game development. That said, one way to inject drama into a game is to employ an interesting camera system. The behavior of the camera is so deeply tied to the way fundamental game mechanics work that any camera system will bring with it a variety of important trade-offs. But whatever type of camera system you use, think about how shot composition can help you communicate with the player at an emotional level. 

**CHRIS PRUETT** is by day a mild-mannered game engineer working on Google's Android platform. By night he runs *Chris' Survival Horror Quest* ([www.dreamdawn.com/sh](http://www.dreamdawn.com/sh)), a blog dedicated to the analysis of horror games. Email him at [cpruett@gdmag.com](mailto:cpruett@gdmag.com).





# THE MAKERS OF EVE ONLINE ARE ACTIVELY RECRUITING LEADING TALENT

SENIOR PROGRAMMERS • 3D ARTISTS • TECHNICAL DIRECTOR • ANIMATORS  
QA ENGINEERS • SENIOR ANIMATION PROGRAMMERS • UI DEVELOPERS

» ATLANTA • REYKJAVIK • SHANGHAI «

VISIT US ONLINE AT [WWW.CCPGAMES.COM/JOBS](http://WWW.CCPGAMES.COM/JOBS)  
TO LEARN MORE AND APPLY TODAY

# the conduit

**EVOLVING FROM ONE BUSINESS MODEL TO ANOTHER CAN BE TRICKY. HIGH VOLTAGE SOFTWARE HAS A 16-YEAR HISTORY OF**

developing over 75 licensed titles across nearly every platform and genre, and THE CONDUIT marks our first original intellectual property outside of a few smaller-scope WiiWare titles. As a company, we've been successful in managing and sustaining our publisher relations as a development partner—but how would we fare as our own masters? Could we apply what we learned as work-for-hire developers to our own creations? Developing original IP, we learned, has many applicable parallels but is also quite a bit different from crafting licensed titles. THE CONDUIT has served as a starting point and valuable platform for evaluating and improving our process as we continue to move further into original IP creation.

## WHAT WENT RIGHT

**1) SELF-FUNDING.** We were in the rare and fortunate position to independently fund all of the product development of THE CONDUIT, which freed us to wholly define the feature set and core gameplay experience on our own terms. It was tremendously exciting and liberating—it's the brass ring every independent developer dreams about—and it added a high level of company-wide project pride and ownership to the title and resulted in increased involvement and dedication from the team.

Developing much of the title without a publisher attached also really put our name out there into the community, and it has paid big dividends for us. High Voltage Software became more known in the industry, and while we were largely unproven as a developer of original IP, our chosen genre and console (An exclusive FPS on the Wii? They must be mad!) and our mostly unregulated product development allowed us to interact directly and openly with the gaming community. Gamers are a tough crowd—they're brutally honest and allegiances can be fickle—and we believe we took the right approach in actively incorporating them into the process as much as possible throughout development. We solicited media and fan feedback and asked them what they wanted in the title early and often, while we had plenty of time to integrate that feedback. We were as direct and honest as possible with them regarding our goals for the game. Company management and development team members responded to emails and posted on forums. We listened to comments and took notes during industry events and private showings and incorporated elements and functionality from the feedback whenever appropriate to our core vision and goals. A vocal and rabid fan community grew up around THE CONDUIT as a result and the internet buzz went a long way toward creating goodwill within the community, which in turn attracted the attention of publishers interested in the game. We embraced the fact that we're an independent

developer—sadly, we're becoming a rarer and rarer breed—and we think that the community responded to it and appreciated our level of involvement with them.

Independently developing the game allowed us to focus on quality and sort out how best to focus on quality within our process and culture. We defined our own milestones internally, which afforded us the opportunity to go back and rework areas of the game that we felt needed some extra attention. We were able to pursue new and innovative techniques and technologies to include different shader and lighting treatments in the environments. We reworked characters to take better advantage of the normal maps as we became more familiar with them. We redesigned weapon functionality and allocated extra time to polishing our control scheme and customization options. We prototyped and tested functionality in ways we were never afforded with our time-to-market properties. This freedom truly gave us a higher level of iteration and commitment to quality—it was a flexibility that is difficult to achieve in licensed titles that typically need to be shipped day and date with a movie release or season launch—and we feel that this









additional attention has considerably improved the overall quality of THE CONDUIT and our quality studio-wide.



**2) WII CONTROLS.** When the Wii was originally announced, we believe that most everybody had candy cane visions of first-person shooter action dancing around in their heads. This unfortunately hasn't been the case. Conceptually, the Wii Remote and Nunchuk should by all rights represent a superior interface to dual analog controls ... but the Wii games released, with scant exceptions, aren't really bearing this out, and FPS titles aren't gaining noteworthy traction on the console. Tiny incremental improvements were certainly being made in the genre, but there still wasn't the widespread acceptance of the Wii's input devices as viable options for first-person shooters that we had imagined and seemed entirely feasible. When preliminary development of THE CONDUIT began, we set out to ensure that we took full advantage of this unique differentiator—indeed, it was the main reason we decided to develop the game on the

platform—as we felt the Remote and Nunchuk absolutely belonged in the conversation alongside the mouse and keyboard and dual analog sticks.

We didn't completely appreciate the magnitude of the challenges of making the controls feel great and intuitive, but remained committed throughout development to our "Wii can be a superior FPS interface" postulation, as we felt it would make or break THE CONDUIT. We assigned dedicated resources for that reason to ensure that we established a new bar for FPS controls on the system. There's an direct analog between firing a weapon and aiming the pointer, between tossing a grenade and flicking the wrist—that you just can't get on another console. The Wii's controls, when done "right" and given proper consideration, are marvelously intuitive and accessible, and we find getting a "Wii kill" a notably more satisfying experience than moving your thumb or sliding a mouse—it's a virtually black-magic-intangible sort of goal that we think we eventually achieved, and that even our harshest critics have been largely unanimous in pointing out as one of the title's greatest strengths.

**3) CUSTOMIZATION.** Similar to our controls, we also felt that it was necessary to offer the player a level of configuration and customization that isn't found in many games, especially on consoles. We always knew we had to include solid presets for players who just wanted to hop in and play. Moreover, we wanted to give the player the ability to play the game as he or she sees fit, rather than try to force our "best" scheme and settings onto players and force them to learn to play it our way.

This "key feature" comes with a bit of a secret. It's important to note that this design philosophy evolved during development—we weren't originally planning to offer as robust a customization set to the player in the retail game. The functionality was developed primarily as tools for the design team to find the optimal scheme. We quickly discovered that there simply wasn't one "right" scheme for every player. Everyone played THE CONDUIT a little differently and had dissimilar preferences. And we felt this was likely indicative of the intricate and subtle disparity of the larger Wii gamer audience,

GAME DATA	
	
	
<b>PUBLISHER</b>	Sega
<b>DEVELOPER</b>	High Voltage Software
<b>PLATFORM</b>	Nintendo Wii
<b>RELEASE DATE</b>	June 23, 2009
<b>NUMBER OF FULL-TIME DEVELOPERS</b>	30
<b>LENGTH OF DEVELOPMENT</b>	22 months



so we ultimately decided to expose it all in the end product. This level of customization has been particularly well received by reviewers and fans, and is a perfect example of a positive opportunity emerging during the adaptive course of product development that wasn't a consideration in the initial design.

The pointer and motion controls are relatively new input devices, and as such we felt that there would inevitably be a certain level of suspicion or resistance to them among a segment of "core" gamers. Just as our parents will likely never see video games as anything beyond PAC-MAN or ASTEROIDS, some people will still feel the Wii controls are inherently inferior to the established mouse-and-keyboard and dual analog standards.

Unquestionably, we gamers love novelty and innovation, but regrettably we are reluctant to change when something works. We must come clean that many of us on the team were at the outset in that suspiciously willful group. Reflecting on game history, however, it wasn't that long ago that FPSes and consoles didn't mix at all. GOLDENEYE (and HALO, more appreciably) changed that and opened up those newfangled game genre opportunities to a diverse new type of gamer. We hope to make the same argument for the Wii by not only offering great controls, but the customization options as well to ease the transition a bit for those that may need an extra nudge toward picking up the controls.

**4) MULTIPLAYER.** Multiplayer became a significant facet of THE CONDUIT, and it almost didn't happen. Initially, THE CONDUIT was a single-player-only experience. But it became immensely clear extremely quickly that a multiplayer component was of colossal interest to the community, and we went straight to work securing the considerable additional funding to amply support it. Thankfully, our investor thought it was an especially relevant and worthwhile addition to THE CONDUIT as well. It's been very highly regarded on the Wii and is the complementary element that really gives the title "legs," as the marketing spinsters say.

Our initial design was pretty bare bones, but the more time we spent on it, the more we recognized that it had genuine potential to truly define our game on the Wii. Subsequently, we devoted loads of additional resources to flesh it out and offer a feature set that's competitive to what gamers expect in an online FPS title. While it was a nerve-wracking decision to add multiplayer to THE CONDUIT, it was the right one and we have a much better game because of it.

**5) FOCUS.** This is tough to adhere to consistently, but by and large we did a good job of defining a solid and compelling feature set and didn't waste a lot of cycles on a system or feature that we felt was not essential to the final game. It's always a tough decision saying no to something, or to pull the plug on functionality that just isn't coming along, or even worse, to stop working on something that has big potential but just isn't supported by the budget or schedule.

A good example of this commitment to quality is our "girlfriend" mode. We got a ton of feedback during development that people wanted an "in the same room" experience, so we spent a little time adding a second reticule that would allow a second player to play the game with a friend along the lines of SUPER MARIO GALAXY. On paper, it sounded like a good idea and perhaps even a nice solution to allow a buddy to join in the action, but after quickly prototyping it, we discovered it wasn't very much fun at all. Our primarily run-and-gun experience encourages a lot of camera movement—something the 2nd player couldn't control—and the end result was ultimately frustrating for both players. We did recognize that there was potentially something to the concept of the mode, but it would have required significant additional work to make it fun and polished. Magnifying the risk, it would have required work from all of our disciplines, so we cut it.

## WHAT WENT WRONG

**1) FIRST ORIGINAL IP TITLE.** As the company's first full original IP, everyone was incredibly excited about the potential of the project and wanted to make his or her mark on the game. We had often dreamed



of developing our own IP, but learned during the initial preproduction phase of development that we weren't entirely prepared for the enormity of this daunting endeavor. It would have been easy to get lost at sea without a licensing Sherpa providing us our typically detailed property style guides, scripts, and all manner of assets to draw inspiration from. Even more terrifying, we were missing another layer of project management—our external publishing partner—vigorously overseeing the development process, wrangling the scope and direction, and supplying actionable feedback. Would we blow our budget and not reach our goals? It was a completely new development environment for us and one that took some adjustment on all levels of the company.

The early going was unbelievably challenging as we worked to define a solid vision for the game. We knew we wanted a first-person shooter on the Wii. We knew we wanted to play up the conspiracy elements in a near-future Washington D.C. We knew we had great tech and could offer one of the best-looking games on the console. The rest was much, much vaguer. We were accustomed to having a publisher help define our art style, help establish the story or mission beats, and give input toward the desired style of gameplay ("We want GTA4 in space!"). As a result, roles weren't initially as well-defined and quite a few signals got crossed, with some directions being interpreted as Holy Writ, while at other times they were consigned to the suggestion box. The early days on THE CONDUIT at times resembled The Tower of Babel, with copious frustrations, disappointments, gnashing of teeth, and pulling of hair at all levels of High Voltage.

The story has a happy ending, nonetheless, and the ship was righted. As the communication and management problems became clearly identified, an internal framework for feedback and discussion was implemented and the process became a great deal more formalized. In many ways, we modeled our system on something we knew very well—the traditional publisher-developer model—with "official" feedback being given, estimates made, and a single executive decision-maker giving the go-ahead for a particular feature, functionality change, or gameplay adjustment. Executive



# postmortem

management became the “external publishing partner” and license sign off. This fundamentally eliminated the majority of the confusion and aggravation, and allowed for a much clearer direction to take shape.

**2) SCHEDULE/TEAM COORDINATION.** Early product development was focused on establishing our new Wii-specific tech pipeline to find out what we could and couldn't do with it, as well as creating demo levels to show off at industry events and brag about to media outlets. We knew we were trying to accomplish something that few other third party developers were trying to achieve on the Wii—we earnestly believed we were filling a void on the console and were treading into largely uncharted waters by treating the Wii as an under utilized system appropriate for “core” gamers—and we were excited and proud to show off all of our hard work. We were wildly successful in this regard, as we generated immense buzz throughout product development, but it did come at a gargantuan cost in terms of managing a coherent and well-organized schedule. We hadn't really scheduled technology familiarization/iteration or demos into the schedule (even the most casual reader of these *Game Developer* postmortems should have heeded those obvious and sensible warnings!), which suffered brutally at the hands of a delighted public as a result.

To that end, team composition was equally problematic as we struggled to define the early schedule. High Voltage always has several teams working in parallel on a variety of projects—we don't keep all our eggs in a single basket. Honestly, that's the key to how we have stayed in the business of making video games for 16 years—and as a result, we have developed an exceptionally flexible and fluid system of managing personnel as we transition from project to project, ramp up and down throughout development cycles, and experience the ordinary ebb and flow of our young industry's reactionary and intransigent behavior. Resource coordination is among our principal strengths as a company, but it also needs to be managed especially cautiously. And that dog bit us at the project onset.

THE CONDUIT's team was simply too large during early development to effectively manage. Pre-production was clearly too short (isn't it always?) as content creators were newly assigned to the team with not enough clear direction or, even worse, not enough work to go around. The resultant pressure to show forward project momentum and keep these people gainfully employed resulted in starting level work and character and object creation before it was properly conceptualized, as designers played catch-up to provide meaningful work for artists and programmers. Our level design suffered as a result and some of this content ended up being team-demoralizing “throw-

away” work. The cart was before the horse in this respect, and art and dev drove design in many cases, forcing them to make levels work around the layout and find alternate solutions for issues based on already-created assets. Much can be learned from the old adage: measure twice, cut once.

**3) TECHNOLOGY VS. DESIGN.** We had some fantastic technology before kicking off the project, but we weren't entirely prepared to use it in full production. The toolsets weren't robust enough and the team wasn't sufficiently practiced with the pioneering techniques and procedures essential for implementing and taking full advantage of our new technology. The product development team wasn't comprehensively prepared to hit the ground running once the production bell rang. Inevitably, a cyclical struggle began. There was a continuous back-and-forth between our technology team and our product development team throughout the creation of THE CONDUIT. In retrospect, there's undeniably a sense now of “if only we'd known this earlier.” While this process is an inevitable part of creating innovation and we'd like to have quite a few “do-overs” (certainly not unique to our project), it was perhaps compounded in our case as we worked diligently to discover all the new shaders, lighting systems, and post-processing effects that we now had available to us.

Our early focus on the technology (largely aimed at creating great visuals and giving a current-gen look on the Wii) did deeply impact the fundamental game design on many levels. In a few cases during early development, creating a dazzling game trumped creating a fun game as we learned the shiny tech and pipeline. It was certainly not intentional, but the sexy and easily distracting realities of new engine creation caused us to occasionally lose focus on the “fun” as preliminary product development began and instead focus on the technical and more bling-oriented aspects of the game.

**4) MULTIPLAYER.** The decision to add multiplayer (while absolutely the right decision and a giant positive in the end product) came at a profound cost to the team. Work on multiplayer began months into full development, well after most of the core systems (weapons, character states, animations, and so forth) were already planned and built. Consequently, without having taken appropriate multiplayer considerations into account, an extensive amount of additional time and resources was invested in making these single-player systems work in a multiplayer addendum, and in several cases, comprehensive workarounds were required for systems that in retrospect should have been rewritten entirely.

Furthermore, we weren't fully prepared as a company to handle the vast internal multiplayer testing that was required for an online FPS.

Historically, we had no need for a dedicated multiplayer lab, or the indispensable complement of QA personnel to support it. While Sega was an enormous resource once we signed them as our publishing partner, early going was a cruel challenge. Ever-resourceful MacGyver-like DIY Midwest developers that we are, we adopted a less than ideal approach for multiplayer testing: a good chunk of the product development team brought kits home three to four nights a week for hours of multiplayer testing for months on end. It was an exhausting schedule for the team—far from perfect conditions—but a regrettable necessity.

Relatively late in the production schedule, we didn't have a suitably stable networking experience. The legitimate fun of multiplayer was found even later than that, making it difficult to tweak and tune as much as we would have preferred. In the end product, we feel we delivered a really solid online experience, but in hindsight there are several areas that didn't get the level of attention or polish that we felt they deserved, as the bulk of the multiplayer development resources were depleted just trying to make it work at all. It's a resounding testament to the solid design and resolute efforts of a passionate team that THE







CONDUIT's multiplayer turned out so well, but it was a very bumpy ride by all accounts.

**5) OUTSIDE FACTORS.** Creating an original IP placed sizeable demands on the team and company that were new to us as a studio; demands that we didn't quite adequately appreciate when beginning production. While the level of community and industry excitement for *THE CONDUIT* was enormously appreciated, we didn't initially have the infrastructure in place to effectively respond to the nearly absurd volume and intensity of attention we received.

We were continually preparing for industry events and generating demo builds to show off the latest and greatest. The art and video support for marketing and PR needs was unprecedented for us as a studio. Pre-release, we received a humongous outpouring of encouragement, suggestions, and questions that fueled us but was equally overwhelming.

We pride ourselves on our level of involvement with the community, but this also added more stress on the company. Interview requests were constant. Fan emails and phone calls (and even unannounced visits to the building!) arrived in legions. As much as was possible, we've responded—there was a level of welcome exhilaration and interest for doing these things—with requests being fielded by everyone from our CEO and founder down to our animators and character modelers, although a side effect is

that it did pull some people away from our main focus—making a great game.

Post-release, we received congratulations, suggestions for the future, and also general support emails to troubleshoot connectivity or networking problems. As a result, we assigned a meticulous associate producer to help manage a good number of these emergent requests and correspondences, and tried to limit impact on the team progress as much as possible. We had patched the dam nicely with our thumb, but this welcome success and outside enthusiasm-driven extra stress was something that we could have better anticipated and prepared for. Nintendo fans are some of the greatest and most vocal fans in the world. And the internet, as it turns out, is a very big place. We aren't deterred by this; quite the opposite in fact. Next time we'll bring a bigger boat.

#### A WELCOME DISCLAIMER

» *THE CONDUIT* has proven itself a somewhat polarizing title, with the caveat "for the Wii" a constant companion as gamers and reviewers attempt to qualify and categorize the game. It's a distinction that we've come to embrace, though we of course recognize that the term isn't always meant as a compliment. *THE CONDUIT* is a decidedly "core" experience on a console not known for wholeheartedly embracing them. First-person shooters are staples on the PC, Xbox 360, and Playstation 3, yet largely ignored on the Wii.

Graphical chops are enormously important to most FPS titles, yet *THE CONDUIT* is exclusively on a system celebrated more for its unique controls and diverse audience than its graphical fidelity. The Wii's online component is not as ubiquitous or established as on other current generation systems, yet we opted to focus heavily on our multiplayer offerings. By outward appearances, *THE CONDUIT* and the Wii are strange bedfellows indeed.

Where we ultimately find our place in the original intellectual property games market is still very much to be determined. But we do know that we love making games for the Wii, and while there were certainly a lot of bumps along the road, we believe we've demonstrated that first-person shooters do indeed belong on the console. We sincerely hope gamers agree as well, but regardless, we've earned a fabulous education during the development of *THE CONDUIT* and are excited to apply these lessons to even more ambitious titles moving forward. The best is definitely still to come for High Voltage Software, and the lessons learned here during our introductory foray into creating original intellectual property will prove germane as we continue to nurture our abilities as game creators and assemble fun and extraordinary worlds across all genres and platforms. 🎮

**JOSH OLSON** was a producer on *THE CONDUIT*. **ERIC NOFSINGER** is High Voltage Software's chief creative officer. Email them at [jolson@gdmag.com](mailto:jolson@gdmag.com).





# COREL PAINTER 11

BY ANDREW JONES



## THE DIGITAL PAINTING REVOLUTION IS UPON

us, and we all stand on the edge of infinite possibilities. At no time in recorded history have there been so many opportunities for creative artists, and we have never had more sophisticated tools with which to render the visions of our imaginations than we do now. I can say with confidence that the new Corel Painter 11 is on the very edge of digital painting software.

I will admit that I am a bit biased. I have been a Painter user and evangelist for more than 11 years, and I can still remember when they sold Painter in a tin paint can. That said, I promise to be as objective as I can when reviewing its latest incarnation.

Before we get into the specifics of the new Painter let's get something out of the way. Whenever you bring up the topic of digital painting software the conversation inevitably turns to a debate about Painter versus Photoshop. In my opinion this is a dialog that serves very little purpose. My answer has always been to use both. There is a slight overlap in their functions and a major overlap in the outcome of the finished process, but they are both amazing programs that do very different things.

I hear a lot of people's comments and grief about Painter's transforming functions, and I've come up with a very good solution for making complicated transformations to a 2D image. I open Photoshop. Painter is a painting program. It does a crap job of doing my taxes too. Life is a multi-layered cascade of applications, and I think that great digital art reflects the collaboration of many programs. When a digital artist can combine the strength of these tools they will truly see limitless potential.

## THE MOST ACCURATE DIGITAL COLOR SELECTION TOOL KNOWN TO MANKIND!

» As one of the most robust painting and brush creation engines available in the modern world, Painter has taken the artist's ability to manufacture beauty to a new level with its new color picker.

I create many of my digital paintings from real life observation—whether it is a landscape or portrait, I'm always observing the color of the natural world and attempting to communicate that into the 2D bit map world of pixels. With all previous color pickers, selecting the right color always felt like shooting a dart at a dartboard, and the slightest movement of the Wacom pen could result in a drastically different color selection. Those days are behind us.

Painter's remodeled color picker can now scale to half the size of the screen. With the accuracy of the new color picker, millions of colors are at your whim. Once your color is selected you can use the up, down, left, and right arrows keys to explore the subtle variations. This is a huge help to any classically-trained traditional artist making the move to digital. It's important



ILLUSTRATION BY ANDREW JONES

for anyone who knows the value of slight value changes in light and form.

Painter also allows you to totally customize your pallet interface and brushes and then share that information with other users. This means that by exporting my workstation I can share an exact digital replica of my working environment with any digital artist around the world.

## REAL MEDIA

» The distinguishing characteristic of Corel Painter is its dedication to synthesizing the experience of working with natural media in the digital realm. Now the software has moved one step closer to completing this illusion by enabling the width of its brushes to dynamically change based on the angle of the Wacom pen in the artist's hand.

For example, if you are sketching with Painter's conté brush and you tilt the brush to the side it will achieve a thicker stroke, mimicking the same real life action. The result of this feature is a more natural sketching experience. Lines and strokes have more variation and can communicate more information, building a more believable and engaging final image. Combine a real media chalk brush with a simulated paper texture and

you may just be fooled into believing that you are actually sketching.

The pattern chalk is one of the most revolutionary brushes in the new Painter. It gives users the ability to take any black and white shape and assign that as the brush. To think that traditional painters have been locked into the paradigm of painting pictures with dead animal hair! Now there are no limitations to what brushes can be constructed from.

With so many options, brushes and custom variables, it's easy to become intimidated. This is really more of a user negative than a Painter negative, but sometimes less can be more. I think Painter would still be a standout product with half the options that it gives the user. With the overwhelming amount of variables you can set in your brush preferences, the combinations and infinite possibilities are staggering.

## CRACKED PAINT

» Still, there are some quibbles with the current evolution of Painter. Large file support could be better. This only applies when you work on images that will be displayed the size of billboards, but it's still a concern. For me, a big con is anything that limits my creativity. The ability to work on extremely large files with out any lag or slowdown would be great.

While we're on the subject of files: what's a .rif file? Lets take the cutting edge that Painter presents and trim off the fat. Over the past decade I have never come up with a reason to save in the priority Painter .rif format.

Painter also needs a larger user base. More people need to be taking advantage of Painter and giving Corel their feedback. It needs to be in



## Corel Corporation PAINTER 11

★★★★

### STATS

1600 Carling Ave.  
Ottawa, ON  
Canada K1Z 8R7  
www.corel.com

### PRICE

\$199.00

### SYSTEM REQUIREMENTS

Windows version:  
Windows Vista or Windows XP (with latest Service Pack)  
Pentium IV, 700 MHz or greater  
1 GB of RAM  
500 MB of hard disk space  
24-bit color display  
1024 x 768 screen resolution  
Mouse or tablet

Mac OS version:  
Mac OS X 10.4 or 10.5 (with latest revision)  
Power Mac G5, 700 MHz or greater  
1 GB of RAM  
500 MB of hard disk space  
24-bit color display  
1024 x 768 screen resolution  
Mouse or tablet

### PROS

- 1 The color selector is now the most advanced color selection technology available to digital artists.
- 2 The real media features give many of Painter's classic brushes a new dimension. It may seem like a novelty but any improvement to my digital painting experience is a welcome addition.
- 3 Painter unlocks your creativity. Corel has gone to amazing lengths to put more creative combinations in the artist's hands.

### CONS

- 1 Complexity. The amount of control that Painter puts into your hands can be intimidating.
- 2 Sluggish when working with large files.
- 3 Needs more users. Painter has the potential to stand next to Photoshop as the industry standard.

every digital art college and curriculum. The more the Painter user community grows, the more Corel will strive to evolve the tool in response to that.

### GOING WILD

» I tell my students that Painter is like a wild animal, or a beautiful woman. It can sometimes be amazingly unpredictable in its behavior. It is out of this type of chaos that great art is born. I can open a new canvas and have absolutely no idea what my final result will look like. With access to millions of colors and thousands of tools and variations of tools, I could paint the same subject a thousand times with Painter and it would never look the same—and I'll never run out of paint. It's this propensity for novelty and change that I appreciate most about the tool. It's important not to get obsessed with what Painter is but what it makes possible.

My greatest moments in digital painting are the times where I become so consumed with the work I am creating that I forget I'm using a computer. My hands disappear, the Wacom dissolves, the menus blur and vanish, and I'm at a place of pure creative expression. This is a moment that I have only been able to achieve with Painter. Like the proverbial finger pointing at the Moon, don't get fixated on the finger but cast your gaze at the Moon. The Moon is the full bandwidth of creative possibilities, and thanks to Corel we as a tribe of creative individuals are one step closer to the Moon than we were yesterday.

**ANDREW JONES** has contributed concept art to *ICEWIND DALE: HEART OF WINTER*, the *METROID PRIME* series, and *HELLGATE: LONDON*. You can find him navigating airport security lines, exotic locations, and venues that feature the latest in underground electronic music, art, and community. Email him at [ajones@gdmag.com](mailto:ajones@gdmag.com).

## ART TRAINING REVIEWS

BY TOM CARROLL

### LYNDA.COM

MAYA 2009 ESSENTIAL TRAINING BY GEORGE MAESTRI

I've already gone on record in *Game Developer* as saying that software developers introduce way too many versions of their software way too quickly for the average person to absorb. That said, thank god for companies like Lynda.com that make it their job to absorb new software revs about as quickly as Bounty soaks up grape juice. Specifically, Lynda now offers various levels of training in the new Maya 2009, and trainer George Maestri breaks it down into small chunks that are easy to process. Maestri's voice and delivery are also very easy to process. In particular I felt that the polygonal modeling section was well done, especially considering that most of the videogame industry is still utilizing relatively simple models, even if they're loaded with sophisticated normal maps delivered through Zbrush or Mudbox. Maya's Paint Effects capabilities, extremely under utilized by many, if not most, are also well explained.

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

size matters

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

RTPatch and Pocket Soft are registered trademarks of Pocket Soft, Inc.



## GNOMON WORKSHOP

DESIGNING CREATURES IN ADOBE PHOTOSHOP WITH AARON SIMS

Veteran film designer Aaron Sims is behind one of the latest instructional DVDs from the Gnomon Workshop. Sims, who designed the look for such films as *AI*, *Terminator 3*, *Constantine*, and *Fantastic 4*, shares his secrets on how to use Photoshop to lay down a character design. The DVD is excellent in all phases, including



showing how to use Warp, Liquefy, and various layer blend modes to build up the character sketch quickly and efficiently—but he reaches the highest heights by demonstrating how to add appropriate surface details. Being a 3D character artist is one of the top jobs in videogames because the available slots in the industry are eagerly sought after and jealously guarded. One of the best ways to attract attention to your own work is to have a selection of well thought out and executed 2D designs. Borrowing from Aaron Sims' repertoire of skills will help to put your own work over the top.

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

## DIGITAL TUTORS

INTRODUCTION TO MUDBOX 2009

Mudbox has furthered the revolution in computer graphics software by making it possible to sculpt and

texture quickly and efficiently in 3D, but you'd be amazed at how little training is available for the package. Digital Tutors' *Introduction to Mudbox 2009* provides nearly seven hours of "production-focused" training to help beginners get up to speed on the techniques currently being used in film, games, and design. The list of topics covered is long and varied, and it includes everything that experienced pros and newbies alike want to know, such as making custom stamps and projection painting (that is most often used to add selective details to models that are fairly far along). The ten minute segment that helps to explain how to refine facial details was especially informative, but a slightly shorter tutorial about painting eye textures was not to be outdone. Pick up *Introduction to Mudbox 2009*. Think of it as comfort food for Mudbox fans.

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

The main textures you create are concrete, wood, cloth, grass, ground, and metal.

<http://eat3d.com>

## ESCAPE STUDIOS

ONLINE 3DS MAX COURSE

Billed as the essential course for anyone who wants to learn more about the power of 3ds Max for architectural visualization and design, Escape Studios' *Online Visualization Course* contains more than 10 hours of HD video content. It takes the student from modeling and texturing various assets within an office environment to creating shaders to enhance their look and using Mental Ray to make the final render eye-poppingly brilliant. This course is a must for anyone who is just getting into Max, or for users of other 3D software who need to brush up on the subject.

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

## BALLISTIC PUBLISHING

EXOTIQUE4, MASSIVE BLACK, AND D'ARTISTE (DIGITAL ARTISTS MASTER CLASS)

While it may seem strange to have a book blurb within an art DVD column, publishers like Ballistic Publishing have really addressed the industry's need for analog collections of the best digital work. Especially in the *d'Artiste* series, the books themselves provide a blow-by-blow account of how some of these pieces were created. My favorite of the three books listed here is *Massive Black*, mostly because it shows the project-based sketches, comps, finished artwork and animation tests, but also some of the team's personal artwork, too. While *d'Artiste* provides great instruction, I can't help but feel that sometimes they're skipping a little too much of the in-between, making the jumps more like leaps of faith. And *Exotique4* simply lives up to its name: exotic to the extreme.

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

## EAT 3D

NEXT-GEN TEXTURING

Do you ever feel like you're fading out of a career that is constantly on the move? Training content like Eat 3D's *Next-Gen Texturing* will help you



hold onto your edge. The reason? It's a topic that will never become irrelevant as long as there are various 3D models that need to look good, dammit! This block of training utilizes Photoshop, 3D Studio Max 2009, and UnrealEd to step through what it takes to model and texture a complex environment scene, and then to check that scene to make sure it's working according to plan. Because you're focusing on creating great textures, this downloadable training is relevant regardless of whether you're using 3ds Max.

**TOM CARROLL** is a video game artist currently with a prominent game studio. He is a contributor to [myIPD.com](http://myIPD.com), an intellectual property portal. Email him at [tcarroll@gdmag.com](mailto:tcarroll@gdmag.com).



**GAME DEVELOPMENT PARTNERS**  
COST EFFECTIVE | HIGH QUALITY | TIMELY DELIVERY

SPECIALIST IPHONE DEVELOPERS

- 2D Game Development
- 3D Game Development (Unity 3D)
- Application Development
- Published Several Titles

Approved Developers for:

● iPhone / iPod Touch    ● Android G1    ● PSP

DEVELOPMENT / PORTING SOLUTIONS FOR  
MOBILE & ONLINE GAMES

2D/3D Art Development | Handheld & Flash Game Development

Special Attractive & Economical Rates    Partner with Us, Today!

Developers of Award-winning games.

E-mail: [info@spiel-s.com](mailto:info@spiel-s.com)    Website: [www.spiel-s.com](http://www.spiel-s.com)





IS HIRING



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

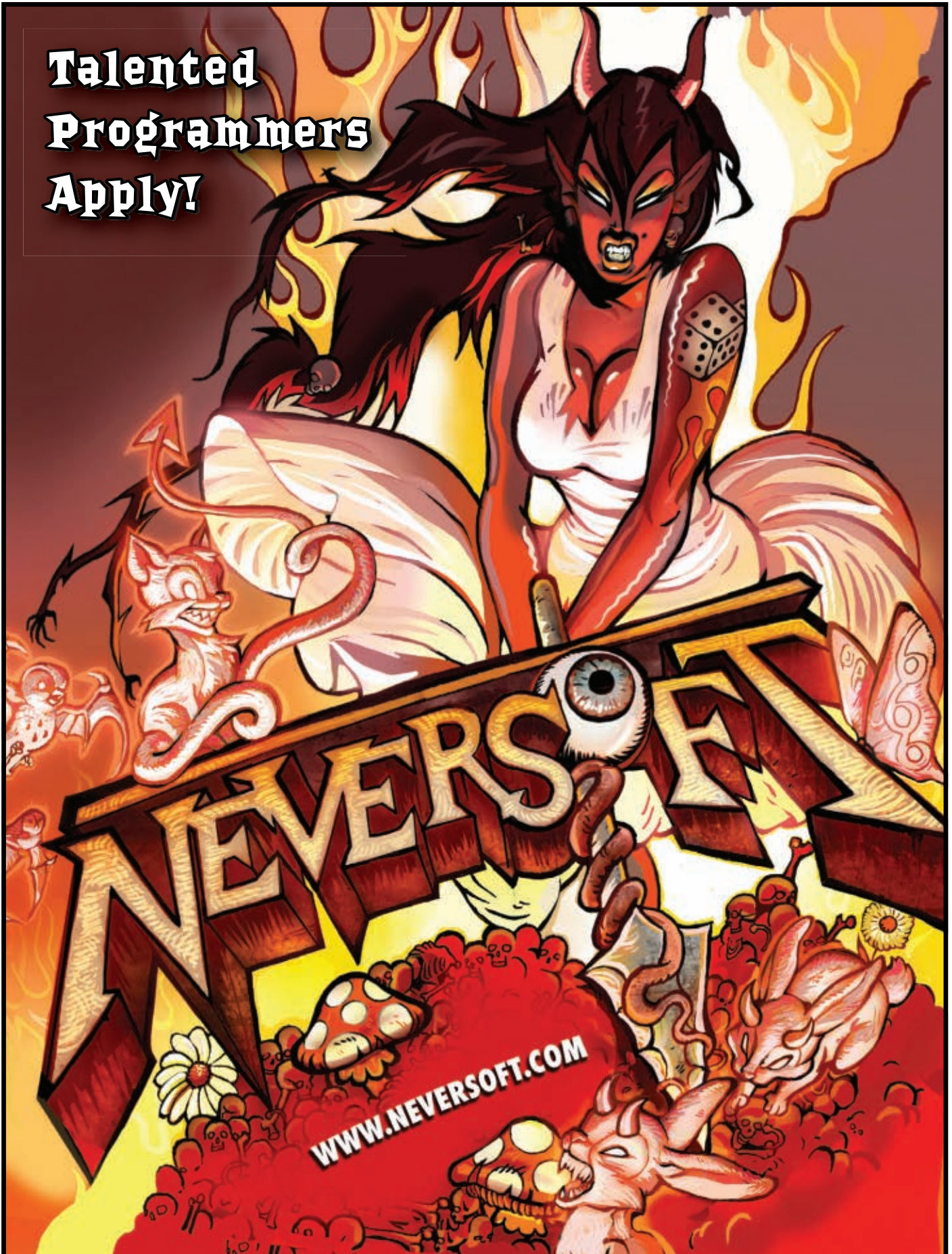
Irvine, California | Austin, Texas | Velizy, France | Cork, Ireland

Seoul, South Korea | Shanghai, China | Taipei, Taiwan

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



**Talented  
Programmers  
Apply!**







# PROCEDURAL CONTENT CREATION

## MAKING SOMETHING OUT OF NOTHING

**PROCEDURAL CONTENT CREATION IS NOTHING NEW. IT'S A CONCEPT THAT'S BEEN AROUND SINCE** the beginning of video games. Even though nowadays hand-crafted content in games is king, we still rely on procedural content quite a bit for heightfield terrains, water surfaces, or particle effects. And, of course, games like *SPORE* rejuvenated the concept and took it a step further by procedurally creating animations and textures.

This month's column delves into the use of procedural content creation in games, and specifically how it was used and what I learned during the development of my game *FLOWER GARDEN* for the iPhone, which uses procedurally-created flowers in pots as its central focus.

### WHY PROCEDURAL CONTENT

» The main reason to create content procedurally is that it can be cheap. Really cheap. Generating new content is a matter of turning knobs and moving sliders. At that point, our imagination is the limit of what we can do, and there's very little work involved in creating each asset.

Before any content can be generated though, the code and tools to create it need to be in place. This means that procedural content creation has a steep initial cost, but it flattens out to almost nothing once that technology is in place. In contrast, hand-created content has a much lower initial cost (in the form of exporters and asset pipeline) but a higher-per asset creation cost (see Figure 1). This creates an economy of scale, where procedural content creation is much cheaper for large amounts of content.

Procedural content can also be a very effective form of compression. Just a few bytes or Kilobytes of information are enough to generate intricate, high-resolution textures, or a full galaxy of alien worlds. This is particularly beneficial for downloadable games that need to keep asset size to a minimum. The flip side to this is that generating procedural content on the fly can be a performance-intensive step, so it's a tradeoff between asset size and CPU time. Fortunately, these days we have plenty of idle cores around that can be put to good use. As a plus, procedural generation can usually be parallelized easily, or at least done concurrently with level loading.

Games that use procedural content can also generate nearly-infinite variations of content on the fly to avoid repeating the same asset multiple times. A game with procedural vegetation can generate thousands of variations of the same plants, which will make the landscape look much more natural. Variation can also be used to adapt to the particular situation in the game: time of day, player's past actions, etc. That would all be much more difficult with pre-generated content.

In *FLOWER GARDEN*, the main reason I decided to create flowers procedurally was to keep content creation costs to a minimum. The game centers around growing flowers from seeds, and the final game shipped with 20 different seed types. I later added another 10 bonus downloadable seeds (with more coming). Modeling 30–50 different flowers, plus all the stages of growth would have been very expensive and difficult to iterate with an artist. Creating a totally new seed type from scratch procedurally takes about 10–5 minutes, and is literally a matter of moving some sliders around tweaking DNA parameters.

The other reason I decided to use procedural content creation was to allow for cross-pollination of flowers to create hybrids. This is something that would be almost impossible with pre-generated flowers, but would have been very easy to implement with procedural generation. Unfortunately, this was a feature I put on hold half-way through development, so it was not implemented in the released version. It still remains as an option for a future update though.

CONTINUED ON PAGE 42

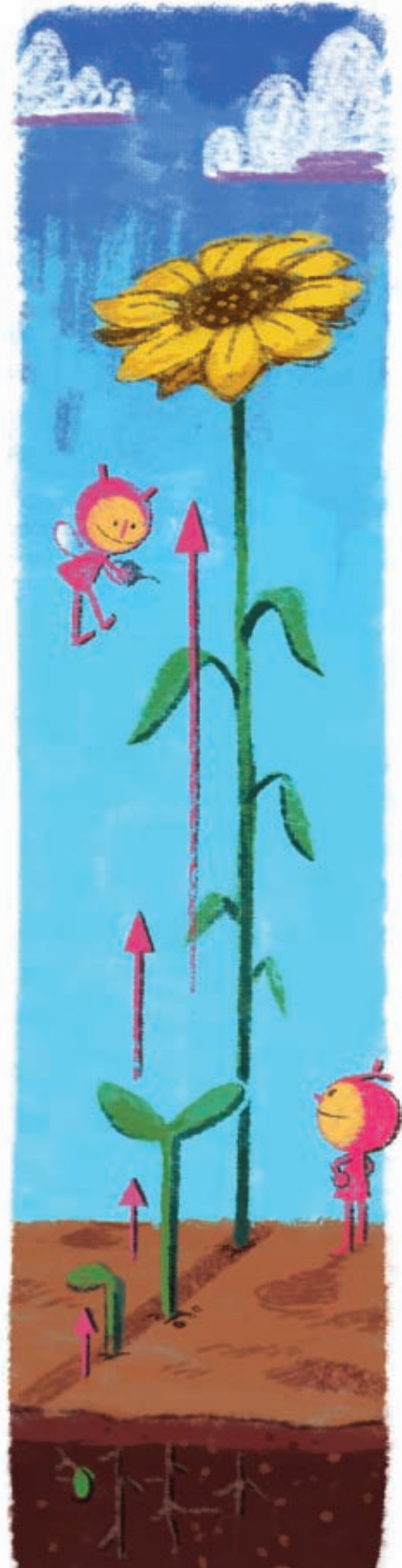


ILLUSTRATION BY MATT BRALY



CONTINUED FROM PAGE 41

Even so, each flower is slightly different than all the others. When a seed is generated, its parameters are tweaked slightly from the reference DNA, so no two flowers are exactly alike.

Content size wasn't a deciding factor, but it helps that the data necessary to fully describe a flower and how it grows weighs in at under 3KB. And that's just the raw size of the binary structure, without any attempts to compress it further.

### TYPES OF PROCEDURAL CONTENT

» Almost any type of content in a game can be generated procedurally. We usually think of meshes, because that's one of the easiest types of content to create—for example, creating a road mesh from a spline laid out in the tool. But it can really be applied to any type of content: textures, animations, and even sounds or speech.

Another important distinction of procedural content is when it happens. On one extreme, content can be generated on the fly, during the game runtime, maybe as frequently as every frame. At the opposite extreme, content can be generated offline by an artist and used as a starting point for a piece of hand-created content, which is then saved and loaded through the static asset pipeline. That distinction is a continuous spectrum, so it's also possible to generate content at load time, or on demand at runtime, but cache it while it's being displayed.

The only type of content that's procedurally generated in FLOWER GARDEN is geometry. Even though the patterns on the petals might look different at first glance, they're just a small set of pre-created texture masks that are used to interpolate between two colors.

What is perhaps surprising is that the full mesh for each flower is re-generated each frame. That's because flowers grow in real time, so they can change appearance from frame to frame. Also, since there's no vertex shader support on the iPhone 3G, all the animation for the stems and petals is done on the CPU, and is based on simple spring systems. It was easy to fold the animation into the last step of the procedural mesh creation, so it happens every frame to achieve a smooth animation.

### THINKING PROCEDURALLY

» It seems that every other word so far has been "procedural," but we haven't defined yet what exactly we mean by "procedural content creation." One possible definition is "content that is created in code," as opposed to hand-created content like I mentioned earlier. That's a bit vague, so let's step back and look at what it really means to generate content procedurally. What is going on when we do that?

It turns out that most forms of procedural content creation are interpolations along some axes. There are some types of procedural content creation that are more purely mathematical, such as Perlin noise or fractals. But the vast majority of procedural creations are based around the idea of interpolating between two extremes. Another way to think about procedural content creation is as a mapping in n-dimensional space that associates input values with a huge space of content. It is by choosing the right axes

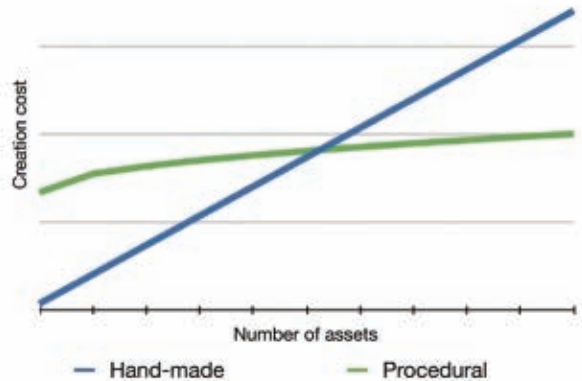


FIGURE 1 Hand created content vs. procedural content costs.

that we're parameterizing all the possible content that can be created.

By that definition, picking a random color for the armor of an enemy would be considered procedural generation. That is example is extreme because we're only interpolating parameters along a single axis. If in addition to color we can change size, armor rating, and decals, we're much closer to what we mean when we refer to procedural content creation.

A more common example is character creation. Many games can change character skin, hair, facial expression, build, clothes, and many other characteristics. In this case we have about 30–40 axes along which to vary our choices. The main limitation is that some of those axes are discrete, with only some limited values in each of them (hair style or clothing).

The flowers in FLOWER GARDEN are created from a DNA-like structure containing information about both their characteristics in full bloom (color, size, number of petals, number of leaves, shapes, and so forth) and the parameters controlling how it grows over time (rate of growth, germination time, or when different parts start growing). Each DNA structure has about 350 parameters that can be tweaked separately. All those parameters—except for the few referring to the texture mask used on the petals and leaves—are analog, and the space of possibilities created by all those axes is staggering.

Before you can write any code that creates content, you need to know really well how to create that content. After all, you can only write a program to do something you know how to do by hand. In my case, I checked out every library book I could get my hands on dealing with flower and plant morphology and learned all I could about the topic. This allowed me to learn new aspects (like the phyllotaxis arrangement of petals), and break the problem down into manageable chunks.

This research phase also allowed me to decide what things not to tackle. For example, I decided to keep plant structure as simple as possible, so I ignored different types of inflorescences and stuck with a single stem with multiple leaves and one flower.

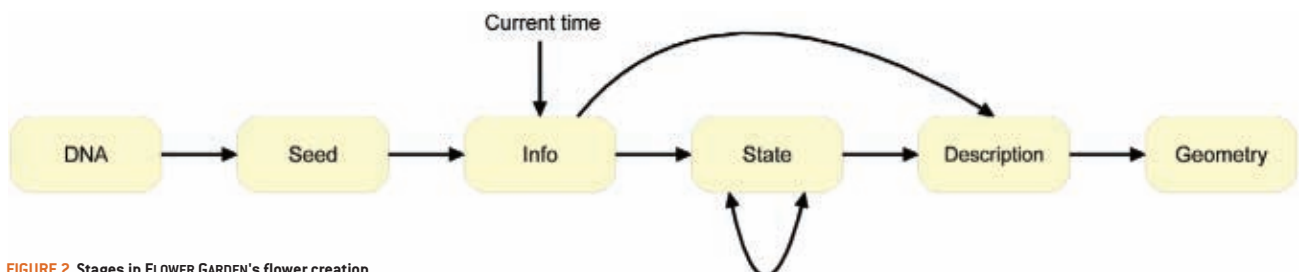


FIGURE 2 Stages in FLOWER GARDEN's flower creation.



Once you're familiar with the vocabulary of the space you want to generate content in, and you've decided which axes you will be using to spawn the space, it's time to start coding.

## MAKING THINGS OUT OF THIN AIR

» Now that we completely understand the problem space, we're ready to start writing some code to generate that content based on some inputs. Except that it's not as simple as it sounds. How do we go about creating a plant that is swaying in the wind, half wilted, and about to start blooming? Or how about a large, residential building, a bit run down, and with the roof caved in? As usual, the answer is to break the problem into manageable steps.

I already mentioned the first type of breakdown, which is to tackle each part separately. The plant can be broken down into stem, leaves, petals, sepals, head, body, pistils, and so forth. Finding hierarchical relationships also helps to simplify the problem. For example, a building can be broken down into floors, each of which is made up of different rooms. Each part is more manageable in isolation and they can be assembled together as a final step.

The more interesting type of breakdown is at the creation level. Even once we've narrowed down the problem to creating a single petal, there's still too much data and complexity to go from a simple description all the way to the final geometry format (where is this petal, how bent is it, how shiny is it, what state of fluttering is it in, how grown is it?). To tackle this problem, it helps to identify several stages in the creation process, each of them more detailed and refined than the previous one. Each stage is used to generate the next one, and the final one is the actual geometry to render (or whatever type of asset you're creating).

In FLOWER GARDEN there are six distinct stages (see Figure 2).

**DNA.** Everything starts with a DNA structure, containing about 350 parameters. These parameters are very high level, and describe both the final look of the plant and how it will grow over time: size, color, shapes, arrangement, etc. As an example, in this stage, the stem is described simply in terms of length, radius, and color. Each parameter contains both a starting and an ending value to change as the plant grows.

**Seed.** From this ideal DNA structure, every time a seed is planted, I create a seed structure, which is identical to the DNA but has some randomness added in some of the parameters (height varies a bit between each plant, but not the number of petals for example).

**Current info.** Given this seed, and the amount of time passed, a structure containing a high-level description of the plant given its current growth is created. The stem info structure contains the same information as the DNA one, but without any ranges, just the current parameters of the stem at this point in the life of the plant.

**Plant state.** From the plant information and the previous frame state, a new state is computed. This state contains the simulation results for the spring

systems, as well as the watering information for the plant. The stem state includes information such as current curvature and velocity.

**Description.** The description is created from both the plant information and the plant state, and it's a more detailed description of the plant, without getting down to the polygon level yet. In this case, the stem description includes an array with joints, each of them with the correct length and orientation.

**Geometry.** Finally, from the description, we can generate the final geometry that will be rendered.

One advantage of having clearly-defined stages is that we can choose when to update each one. For example, it's clear that we only need to create the seed stage from the DNA once. But we might choose to only update the current info state once every few frames, or not at all even if the plant is not actively growing.

It's important to realize that this is just a conceptual way of thinking about content creation. In practice, you probably will want to keep some context as you're creating each part of the asset instead of treating them in complete isolation. By doing that, you'll be able to reuse potentially expensive calculations you just computed in a previous step (for example, by reusing the transformation of the neighboring petal and just adding an extra rotation to it). It will also allow you to create more efficient meshes by having a broader view of the content you're generating (for example, by combining all petals in a single mesh, or even in a single triangle strip).

## LESSONS LEARNED

» This is not news, but it's worth reiterating: Procedural content can be great, but make sure it can be tweaked to the satisfaction of artists and designers. Otherwise you'll end up with worlds full of generic-looking landscapes, forests, or even flowers.

By the time you're done writing the code that generates the content, you're pretty close to an expert in that area. It's important to realize that you know all about the different axes along which your content will be generated, but your brain can't visualize every combination that results from it. Allow yourself time to explore the ranges of what's possible. Crank some of the sliders to the maximum (or beyond the maximum you had initially set), or add a feature to create truly random content by picking random values along each of the axes. I guarantee you'll surprise yourself and will discover things you never thought were possible. I ended up creating some unique flowers by cranking up the curvature of the petals to the point that they would interpenetrate with other petals, forming a very special bell shape.

It all comes down to three things: Learn the problem space, choose good axes to parameterize content, and implement it in small steps. With that, there's nothing you can't create. 🍀



**NOEL LLOPIS** has been making games for just about every major platform in the last ten years. He's now going retro and spends his days doing iPhone development from local coffee shops. Email him at [nllolis@gdmag.com](mailto:nllolis@gdmag.com).



# DEADLY SKILLS?



PlayStation®

**We are HIRING across all disciplines:  
programmers, artists, producers, designers, etc.**

Apply online: <http://www.us.playstation.com/Jobs>

SONY



COMPUTER ENTERTAINMENT®





# CHECK OUT THAT ASSET!

## A STRATEGY FOR MANAGING ART FILES

**THE LAST EPISODE OF PIXEL PUSHER WAS A BRAVE ATTEMPT TO DO THE** impossible—namely, to get artists interested in file management. Bo-o-o-o-ring! Seriously though, the paradoxical point we tried to stress was this: File organization is important precisely because it's so dull and so annoying. If you try to ignore the boring, bureaucratic business of file naming and storage you're doomed to lost files, impossible debug sessions, and lots of grousing when assets get passed among teammates. On the other hand, not many of us can really get enthusiastic about clerical work. It's a case of "bored if you do, damned if you don't."

In the time-honored Pixel Pusher fashion we offered a way out of this unfortunate dilemma. The technological fix we proposed relies on the same kind of techniques that music and video applications use for managing big libraries of songs or movies. Just as your MP3 library can be searched and sorted by artist, genre, year, or album no matter how the folders on your iPod are arranged, it should be possible for you to find and work with game assets quickly and easily using search and sort keys that make sense for your project. By integrating this kind of information (the fancy buzzword is "metadata") into common operations like opening, saving, and exporting files you can save time, keep better track of your data, and waste less energy on manual file mongering. At any rate, that's the hope—so this month, we're going to look at some of the practicalities of implementing such a system.

### ASSERTION

» Before you can set out to clean up your file management mess, it's worth asking what you're really interested in managing. A large modern game contains an unimaginable wealth of data. Trawling through that huge, trackless mass of information is a serious proposition—hardly the sort of thing you'd ordinarily want to tackle with a scripting language like MaxScript, Python, or Mel.

Fortunately though, this enormous cloud of files condenses down to a more comprehensible size when you remember that individual files aren't really what you're managing. When an artist gets tasked with fixing a hitched run-cycle or adding a new object to a particular scene, the conversation always starts with a game asset or level rather than a file. "The golden dragon's run is a couple of frames too long," or "design needs some more trees to screen out the terrace in the temple level," are natural ways to talk about the problem. It's also what we should be thinking of when we search for files.

The list of characters, vehicles, props, and levels in a game can still be quite large, but it's a tiny fraction of that daunting total file count. Hence, it's also a big step down in complexity both for the users and the tools team. It makes sense, for this reason, to build a system that recognizes that all those files are grouped together to form game assets rather than living in a vacuum. On the human side, this helps because assets have memorable identities, which makes them ideal landmarks. On the computer side, this reduces the problem from directly managing tens of thousands of files to the much simpler job of managing dozens or hundreds of assets.

Focusing on assets, rather than files, is the right way to help both users and tools deal with the scary complexity of a modern game project. An asset-based system speeds up finding things by asking two easy questions:

"What am I working on?" and "What part of it do I care about?" instead of the one very hard question: "Which one of the 13,745 files in this folder hierarchy do I want?" This may sound daunting, but the scripting needed to make this work is pretty simple. Modern machines are fast enough that even script-based tools can speed up file finding enormously over the traditional hunt through the file dialog box. You can achieve quite a lot with fairly simple techniques that should be within the capabilities of a typical technical artist.

### ASSEMBLY

» There are lots of ways to build an asset-based set of tools, but we'll sketch out a very basic one. The easiest way to build an asset toolkit is simply to represent every asset with a single file—for the sake of simplicity we'll call it an "asset file." These asset files act like a database index for your entire project—anything that ends up in the game is part of some asset file or another (or possibly is shared between assets). Your tools can use the asset files to find the information they need without bugging the user.

The asset file doesn't have to be complicated. It's basically a text file containing all the metadata tags that describe the asset. The set of tags you support can be as simple or as complex as your project requires. Most tags are pretty obvious: what faction or team does this character belong to? Is this an outdoor or an indoor prop? Is this a single or multiplayer level? You can also add tags to help with production management: Has this asset been approved? Is it placeable in the game? You could even use tags to assign assets to teams or individual artists. In all of this you'll be helped by the fact that tags—unlike folders!—aren't mutually exclusive; tagging an asset for Sally the modeler doesn't mean you can't also tag it for Joe the animator.

To get value from these asset files you'll need to offer some UI to make sure users don't have to laboriously type in lots of tags (with the attendant typos). The easiest way to handle this is to script the creation of new asset files—asking users to create them by hand will never be popular! Luckily

**A fast but slightly over-generous search that updates while you type will make finding files a breeze.**

most projects don't need an enormous volume of tagging, and either Mel or MaxScript can provide an adequate list of "approved" tags. A handy trick is to treat the disk path of the asset files themselves as if they were tags, so everything in the "levels" folder appears as if it's tagged with "levels" and so on. Since even a tag-based project will still have files and folders, this gives you a lot of useful tagging information for free.

### ASSUMPTIONS

» To use these asset files, you'll need a script to catalogue them all so the user is presented with a quick, painless search through what will probably be a long list. Disk access is pretty fast these days (on a modern machine, a Max or MEL script can catalog 10,000 files in 3–4 seconds). However, you don't want to go hunting across the entire disk every time the user asks



for an asset file; it's probably best to scan the file tree for assets once at the beginning of your Max or Maya session, keeping the results around in memory for speedy access.

Once you've got this list of files and their associated tags in memory, the next step is to winnow it down to the items that match any keys your user types in. The "search" consists of simply looping through all the collected assets, collecting any assets whose name, path, or tags match what the user typed. (See Figure 1.) Just as when searching for music, the key here is to be quick, rather than precise. You might think you'd need a complex database style query of the "this AND that but NOT this other" variety, or complex pattern matching. Don't bother. This is an artist's tool, and a complex database style UI will rarely repay the time it takes to build. A fast but slightly over-generous search that updates while you type, on the other hand, will make finding files a breeze. The real key is to simplify that original list of hundreds of assets down to an easily managed list of half a dozen or so, allowing the user to finish with a mouse click or by tabbing through the list. Combined with a well-chosen set of tags, the strategy is much more efficient than hunting through a huge file tree.

ASPIRATION

» Once the user has selected the asset, there's a second step: selecting the aspect of that asset you want to edit. Selecting an asset isn't the same thing as selecting a single file, it's almost like navigating to a directory full of files—although, as we've said before, with a metadata-based pipeline you don't care as much whether the files that "belong" to an asset are physically located in one particular folder. Instead, the asset file acts as a registry of all the pieces that go together to make up the asset. If the asset is a character, its animation files could all be listed in the asset file. For a level, individual encounter spaces or stages might be listed instead. A shader might include paths to the bitmaps it needs. The principle is always the same: anything a user or a script might need to know to work with the asset should be

recorded in the asset file explicitly. Moreover, you want store them in a way that makes the job of each file clear.

Naturally, you won't be scoring high on the "artists hate clerical work" index if you expect artists to maintain asset files by hand. You'll want to provide a dialog that makes it easy to add new files and assign their pipeline functions easily. Many teams already have tools for setting up files with approved naming conventions and file locations; adapting these to update asset files is fairly straightforward. The key here is asking users to clearly tell you what they're doing, rather than trying to guess their intentions by parsing file names or locations. For example, "I am adding

Focusing on assets, rather than files, is the right way to help both users and tools deal with the scary complexity of a modern game project.

a walk animation" is easy for a computer to deal with, but "I happen to be creating an animation with 'walk' in the name" is not. Fortunately new files are added to an asset only rarely, so it's not a terrible burden to ask the user for important information as part of the creation step. Once the new animation, part, or what-have-you is added the system will forever know the role it plays and where to find it, and the user won't have to be pestered for help in digging it up.

The idea of keeping this list of all the bits and pieces of the final game character, level, or prop stored in your asset file has three big advantages beyond just bypassing the clunky "file open" dialog:

First, it's the key to getting users to their work efficiently. To refer to an earlier example: it's natural for a human to think "go fix the golden dragon's run cycle"—but it's hard to express this to a computer unless you have something like an asset file to help your scripts find the files you need. The

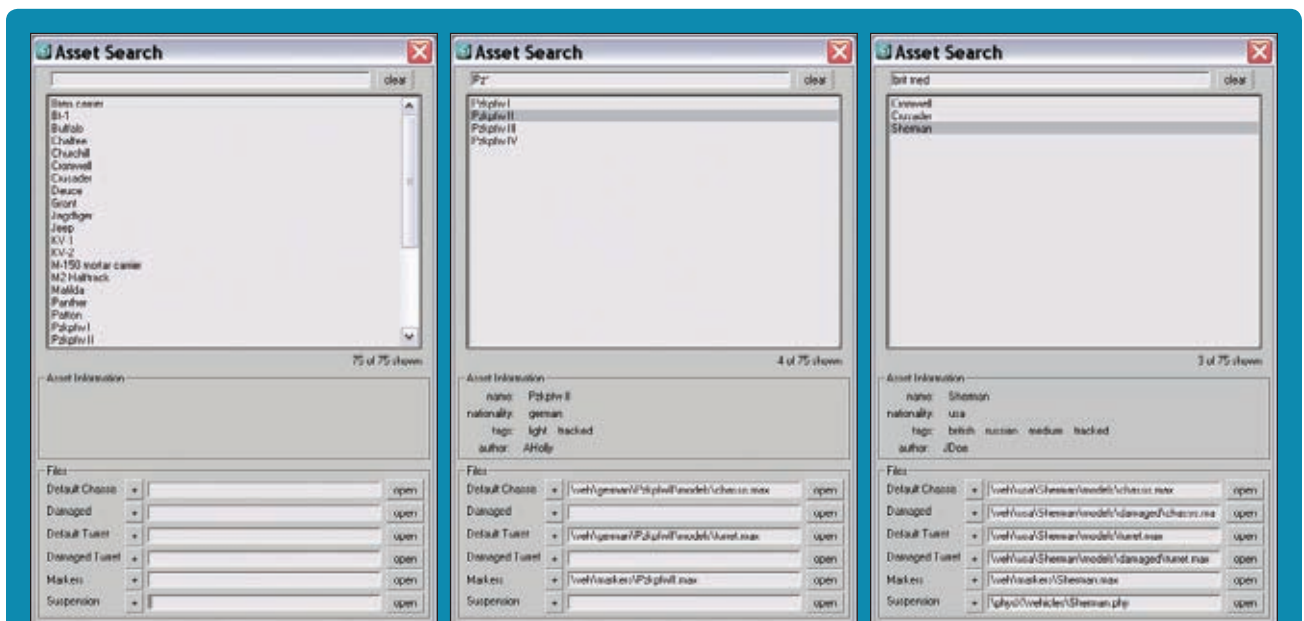
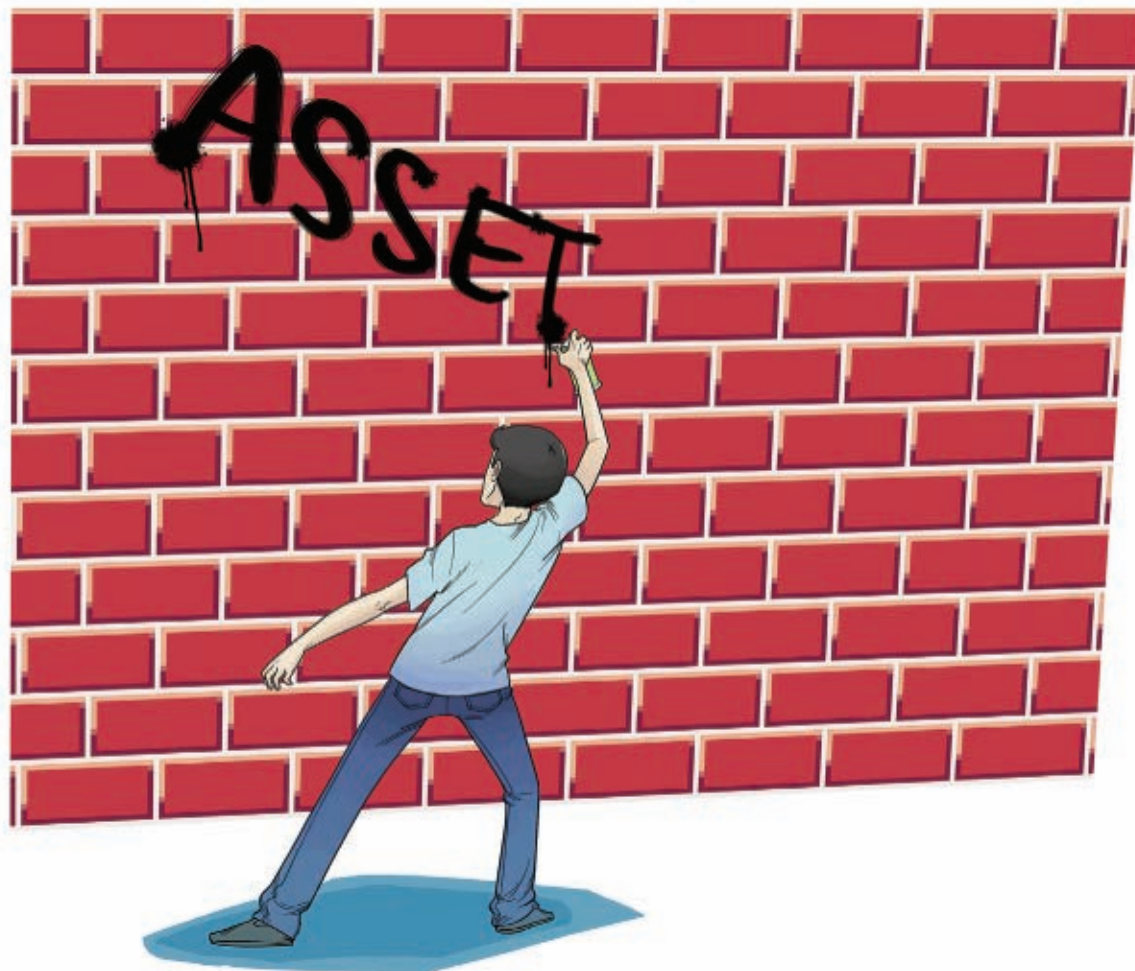


FIGURE 1 Asset searching in action. Here, tags link to various assets by context.





Asset tagging the hard way.

asset file provides all the relevant bits and pieces in one place, no matter how they may be spread out on disk. This speeds up the otherwise tedious business of finding or checking out files enormously. Plus it's a natural complement to automatic source management scripts that handle check-ins and check-outs and no-dialogs-needed exports.

Secondly, recording all this stuff in the asset file is a huge boost to the power of other scripts and tools. Rules like "go up two folders then look for a folder called 'animations' and check for <assetName>\_run\_cycle.ma" are awkward to maintain in script, and are vulnerable to all sorts of human errors. Contrarily, if the various roles a file can perform are simply recorded in the asset file, matching the intention of a script with the location of a file is trivial. Need the high res source model so you can recast your normals? The normal mapper tool can check the asset file to see if you've got one. Looking for the max file that makes the palm tree you reference all over the jungle level? The jungle's asset file should have a pointer to it, even if it's in a shared library folder rather than in the jungle folder.

That last point brings up the third handy feature of an asset-based toolset. Storing paths in the asset file lets you do file sharing on the cheap—if two vehicles both reference the same model of a tire, you don't have to tie yourself in knots to decide the torturous folder hierarchy which lets them both see it. Rather, simply including its location in both

asset files allows modelers working on either vehicle to quickly get to the file they need. Of course, this won't magically resolve some of the other issues involved in asset sharing (always a touchy subject) but it can certainly make simple sharing easier, safer, and less of a pain for overstressed artists.

### ASTOUNDING!

» To sum up, there's a lot to be said in favor of using a system like the one described here as a key component of your art tools setup. An asset-based system can be much faster than a traditional file tree for common tasks, and the gatekeeper scripts that make it work also cut down on the tedious, error-prone clerical work that drives artists crazy. It's true that neat files and faster checkouts aren't as sexy as cool new shaders or whiz-bang animation rigs, but biting the bullet and putting together a good file handling toolkit will more than pay for itself. Plus, once it's done, putting together the cool stuff will be a bit easier as well. 🎧

**STEVE THEODORE** has been pushing pixels for more than a dozen years. His credits include *MECH COMMANDER*, *HALF-LIFE*, *TEAM FORTRESS*, and *COUNTER-STRIKE*. He's been a modeler, animator, and technical artist, as well as a frequent speaker at industry conferences. He's currently content-side technical director at Bungie Studios. Email him at [stheodore@gdmag.com](mailto:stheodore@gdmag.com).

# ACTIVISION®

## GREAT GAMES

START WITH

## GREAT PEOPLE



ARE YOU READY TO EXPLORE THE MOST SOUGHT AFTER JOBS IN ENTERTAINMENT?



Visit our site: [activision.com](http://activision.com)





# TURN-BASED VS. REAL TIME

WHICH WAY SHOULD YOUR NEXT GAME ROLL?



Although it plays in real time, *PLANTS VS. ZOMBIES* has the deterministic mechanics of a turn-based game.

**ONE OF THE MOST IMPORTANT CHOICES** A designer makes at the start of a project is whether to make a turn-based game or a real time one. Each type of base game mechanic provides potential benefits and drawbacks. While turn-based games favor more strategic and transparent play, they can feel a little stodgy to players used to action-oriented titles. Real-time games, on the other hand, are more immersive and multiplayer-friendly but can also easily overwhelm new players if not well-paced.

Turn-based games, of course, descend directly from the board game tradition, predating video games. Indeed, the fanbase for turn-based games still overlaps significantly with the fanbase for board and card games. Real-time games (excluding sports), however, were only truly possible with the advent of computers. Quite a few games—*SUPER MARIO BROS.*, *TEAM FORTRESS*, *FIFA*, *PAC-MAN*—could only ever conceivably be developed as real-time games.

There are quite a few games that could go either way though, with an understanding that each path comes with its own set of trade-offs. Rogue-like dungeon-crawlers, for example, have been made as both turn-based and real-

time games. Early versions, such as *NETHACK*, were purely turn-based; the game's clock only moves forward each time the player takes an action. However, Blizzard's *DIABLO* put the same explore-and-loot formula into a real-time environment and created an experience that was less strategic but more visceral and potentially addictive. Furthermore, without the waiting inherent in a turn-based system, the designers could develop a viable multiplayer mode.

Nonetheless, *DIABLO* has not supplanted the continuing popularity of turn-based roguelikes, such as *POKEMON MYSTERY DUNGEON* or *SHIREN THE WANDERER*, which maintain their own tactical charm. Thus, deciding between turn-based and real time is not a question of which system is better or worse, but rather a question of which set of trade-offs best fits the game the designer wants to make.

## HOW MUCH STUFF?

» One simple way to look at a game is by asking how many game systems and elements the player needs to master to feel competent. For example, a typical shooter might have ten weapons; a real-time strategy game might have fifteen units per

side; a role-playing game might have twenty spells available. New players can be intimidated by the sheer quantity of new concepts and options a game presents to them, and the time pressure of a real-time game only makes this learning experience an even greater challenge.

When first prototyping the original *CIVILIZATION*, Sid Meier originally built the game as a real-time simulation. Inspired by Will Wright's *SIMCITY*, he tried to extend the concept to a global scale. But he quickly found that players were overwhelmed by the high number of new game systems they needed to juggle at once. After all, *SIMCITY* had no diplomacy, no trade, no combat, no research, and definitely no marauding barbarians. Thus, he changed course and rebuilt his prototype as a turn-based game, and the phrase “just one more turn” entered the gaming lexicon.

Designers should always be aware that each game can only contain so much “stuff” before the center cannot hold, and the experience overpowers the senses. By removing time pressure, turn-based games allow players to adjust the learning curve to their own needs. Veterans can still play quickly, but new players can take their time poking around the interface and thinking through their moves.

This makes turn-based games generally more accessible than real-time ones. It is no surprise that many of the most popular casual games are turn-based, from staples like *SOLITAIRE* and *MINESWEEPER* to PopCap's stable of *BEJEWELED*, *BOOKWORM*, and *PEGGLE*.

## DETERMINISTIC OR CHAOTIC PLAY?

» At their core, turn-based and real-time games play to different strengths. One example is the question of whether an experience should be deterministic or chaotic. With the former, success often depends on knowing exactly what the results of one's actions will be; in *PUZZLE QUEST*, for example, the player needs to know that when a row of four skulls disappears (forming an attack), the other pieces will fall in a specific way so that a new column of consecutive red gems might form. Just because some luck elements are involved—such as the unknown new pieces which fall from the top—doesn't mean that the player isn't mapping out an exact series of events in her head. This sequential gameplay is one of the core strengths of turn-based games.

CONTINUED ON PAGE 50



CONTINUED FROM PAGE 49



The V.A.T.S. mode in *FALLOUT 3* gives users the option to switch at will to a strategic play style.

On the other hand, chaotic, unpredictable gameplay is a strength of real-time games. When players first spot a Heavy/Medic combo in *TEAM FORTRESS 2* (in which the weak Medic unit follows and heals the very strong Heavy), they know that they are probably in trouble, but the sequence of events to follow is so varied that players know it's impossible to overanalyze the situation. A sniper could kill the medic. An explosion might knock the heavy off a platform. A spy might sneak up behind them. An event on the other side of the map might encourage one side to simply abandon the area. Real-time games support chaotic gameplay best because, with the added pressure of a shared clock, players are not able to reduce each situation down to a repeatable series of moves and counter-moves.

### MULTIPLAYER OR SINGLE-PLAYER?

» Another divide which defines the different strengths of turn-based and real-time games is whether the focus of the experience is multiplayer or single-player. Generally speaking, multiplayer games work best in real-time, whereas turn-based games usually focus on single-player sessions. Turn-based games like *ADVANCE WARS* and *CIVILIZATION*, have only a tiny, hard-core multiplayer audience (though more casual games like *MARIO PARTY* or board games like *CATAN* work well this way). On the other hand, real-time games with similar themes, such as *COMMAND & CONQUER* and *AGE OF EMPIRES*, respectively, gained much of their popularity from their multi-player modes.

The reason for this divide is clear—waiting for another player to finish his turn is anathema to fun—so designers looking for a synchronous, multiplayer experience almost always prefer

real-time games. However, because no one else is waiting, designers of purely single-player games give themselves the option of using turn-based elements whenever convenient, to either add some spice or allow more strategic play. For example, the single-player game *FALLOUT 3* allows players to pause real-time combat and enter V.A.T.S. mode to strategize which enemy body parts to target, even displaying the exact probability of success for each possible choice. The game then “rolls” for the hits. Similarly, the *BALDUR'S GATE* series uses a hybrid model, with real-time combat that pauses depending on certain player-selected events, such as when a character receives damage or a new enemy becomes visible.

### BREAKING THE RULES

» These are but a few of the many games that blur the line between “pure” turn-based and real-time systems. For example, what about turn-based decisions with a time limit, such as *MADDEN*'s play-calling clock? What about *X-COM*, with its crunchy real-time strategic shell surrounding a gooey turn-based tactical core? Or the *TOTAL WAR* series, which does the exact opposite? What about *EUROPA UNIVERSALIS*, which is technically real-time but plays out so slowly that it “feels” like a classic, sprawling turn-based strategy game. How about asynchronous Web-based games like *TRAVIAN*, which play out over months instead of minutes, eliminating the time pressure but keeping the multi-player benefits of real-time play? What about *BANG! HOWDY*, which plays as a typical tile-based tactical wargame, except that each unit's turns regenerate in real-time? In reality, a vast continuum stretches from one extreme to the other, and most games find a space somewhere

in the middle.

Therefore, the most important thing to focus on is not the labels themselves but what types of gameplay they represent. For example, the tower-defense game *PLANTS VS. ZOMBIES* is ostensibly real-time, but its characteristics are more in line with traditional turn-based games. The gameplay itself is strictly deterministic, even more so than many turn-based games. The map consists of five tracks along which the zombies progress, each with exactly nine slots on which to place defensive plants. Furthermore, the zombies' behavior is entirely predictable—Pole Vaulting Zombies will always jump over blocking Wall-Nuts,

even if that means falling right into the jaws of a Chomper plant. The game may look chaotic to an observer, but—like most tower-defense games—the strategic play is built upon predictable enemy behavior. The real-time mechanics simply provide time pressure, not the other qualities usually associated with the format, such as chaotic play or a multi-player mode.

Likewise, *BOOM BLOX* is a turn-based game which eschews the usual strengths of the format. In the game, players have a discrete number of throws of a ball with which to knock down various block-based structures. Unlike most turn-based games though, *BOOM BLOX* is a very chaotic affair, with unpredictable physics-based game mechanics. Unlike *PLANTS VS. ZOMBIES*, in which players' actions take place on a precise 5-by-9 grid, players of *BOOM BLOX* use strictly analog controls to point at the screen and then “throw” the ball with the Wii Mote. Chaos theory dictates that an identical series of throws will almost never happen twice in a row. Furthermore, this unpredictable nature coupled with the very short turns (each only a single throw) makes *BOOM BLOX* an excellent multi-player game, a rare feat for turn-based video games.

Thus, in the end, deciding whether to make a game real-time or turn-based is less important than deciding which aspects of those formats are most relevant to the overall design. As they say, one needs to learn the rules to know how to break them. 🎮

**SOREN JOHNSON** is a designer/programmer at EA Maxis, working on an unannounced project. He was the lead designer of *CIVILIZATION IV* and the co-designer of *CIVILIZATION III*. Read more of his thoughts on game design at [www.designer-notes.com](http://www.designer-notes.com). Email him at [sjohnson@gdmag.com](mailto:sjohnson@gdmag.com).





# RETRO FITTING IN

OUT WITH THE NEW AND IN WITH THE OLD

## THE NEWEST TREND IN GAMING

is old games; or rather, taking classics from the halcyon days of floppy disks and 8-bit cartridges and bringing them up to date for 21st century audiences. If the opportunity to work on a remake presents itself, take it. It's a fascinating peek into a formative era of gaming on which many of today's audio professionals were weaned.

But the process of updating a classic is full of unique challenges. In these instances, companies are paying to remake a game they've paid for once before—sometimes more than once before. Knowing some of the pitfalls you'll face in advance can help save critical time and money.

## NOSE TO THE GRINDSTONE

» Step one is the easy part: become an expert on the original game's audio. Play through the game completely at least once and make detailed notes on all the sound effects, music, and voice, if any exists. Your playthroughs should result in a detailed asset list covering every audio file in the original, whether it needs to be updated, where it occurs in-game, and what kind of update it will get. Note how many sound effects seem to be capable of playing back simultaneously. Note whether the music in the original loops or

plays simply as one-shots, and whether changes to this would be a benefit. If the game has successful sequels, find out if they used recurring actors as part of their voice cast.

More likely than not, any game being remade is going to have some form of existing fan community. Tap into fan game FAQs, web sites, and playthrough videos on YouTube. Make sure that you know the complete scope of the game and have documented everything, including branching gameplay, alternate endings, and easter eggs which may affect audio.

## TINKER AND TWEAK

» After you know the scope of the project, befriend the game's engineering team. A remake involves a massive amount of reverse engineering and the engineering crew will be crucial to any successful audio retrofit. If the game is being rebuilt atop the game's original engine, audio may very well find itself limited to only as many triggers and implementation hooks as initially shipped with the title. Adding new scripts and audio triggers into old code can be like trying to add a wing onto a house of cards. It's not impossible, but expect some push-back from the engineering team and be

prepared to compromise. If a new or hybrid engine is being developed, level designers and engineering staff will be critical to any new implementation support. Additionally, you may need engineering to help with everything from extracting any text scripts intended for dialogue recording to extracting and translating old proprietary music file formats into usable MIDI data.

Audio gets updated in retro games because it sounds bad, and it sounds bad because of technical limitations from an older era of gaming. With those limitations now no longer relevant, there are basically two ways to look at the challenge of updating the game's original audio content. The first is to recreate new, higher quality versions based upon the sound of the original audio. The second is to create new, higher quality versions based upon the intent of the original audio.

The difference between when to base your changes off of original sound or original intent will only come from hearing the audio in context. For decades, games weren't able to use digital audio files and relied on tone generators and MIDI sound cards for their playback. In many of these instances, an exact translation of sounds would be completely

inappropriate for today's audiences. Just because a game from twenty years ago uses a MIDI snare drum every time a door is opened or closed doesn't mean that a beautiful 24 bit, 96kHz bank of randomized snare samples is the right choice for the new game's suite of door sounds. In this case, you have an opportunity as a sound designer to really explore the sound of this world in ways that haven't been heard before. Though working in an existing franchise, there's a fantastic amount of freedom available.

The same goes for composers. What may have once been written for two square waves and one triangle wave fifteen years ago may now find itself meriting anything from full orchestra to solo voice to a jazz or rock quintet. It may also, however, still find itself comprised mostly of square and triangle waves, though updated with 21st century production techniques and forming the core of a new glitch pop track. The question of context will again decide the direction along with the aesthetic goals of the core design team.

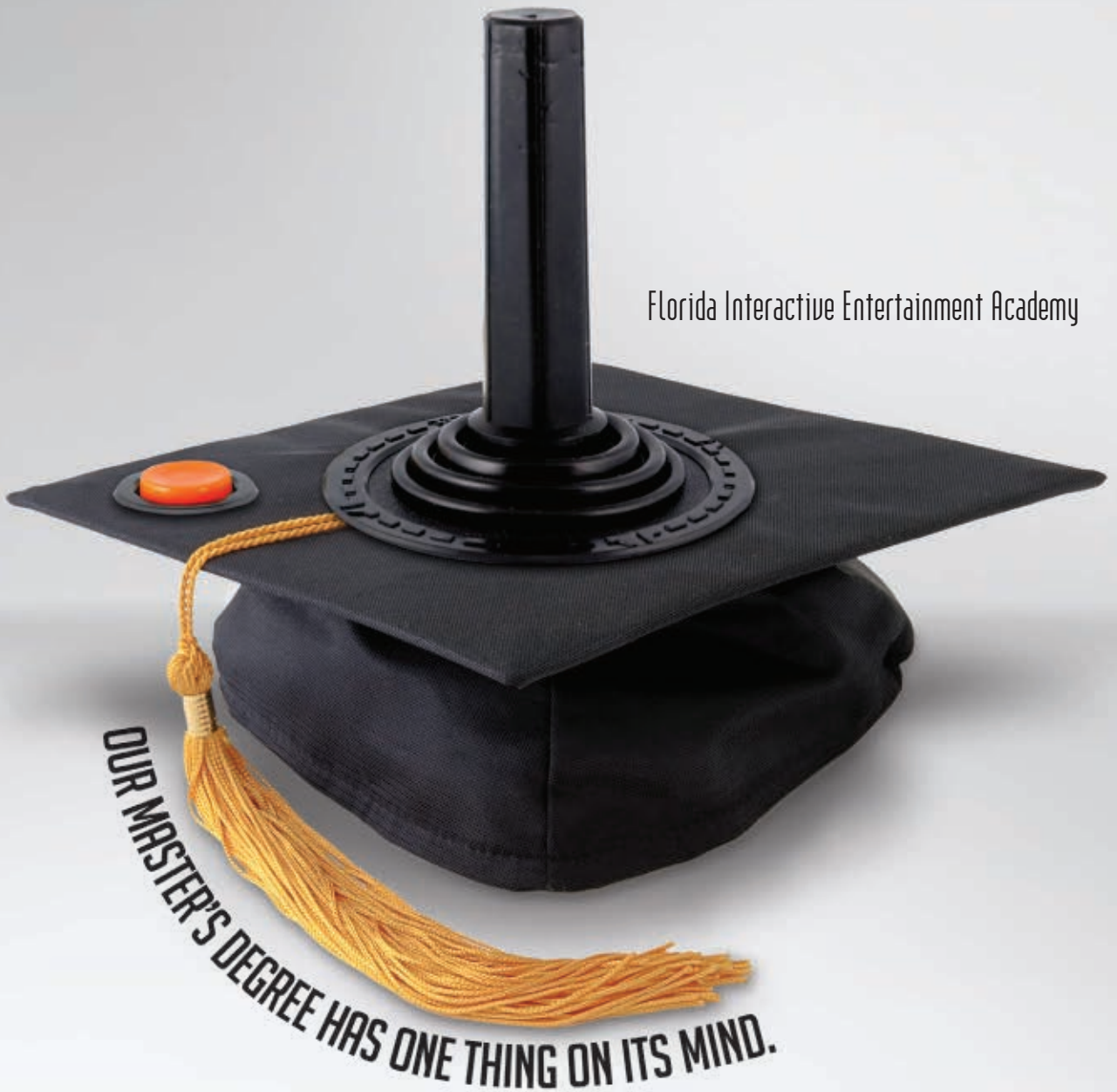
Regardless of the changes made, the primary goal is to always remember that your new work will be meticulously scrutinized alongside the original. Never lose sight of what made the original a classic in the first place. If the original was fun, don't suck the fun out in an attempt to put your creative stamp all over an already beloved game. Simply enjoy the history lesson and the chance to pay homage to the gaming greats of the not-so-distant past. 🎧

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



Capcom's original BIONIC COMMANDO (1988) for the NES and BIONIC COMMANDO REARMED (2008) for PlayStation 3, Windows, and Xbox 360.

Florida Interactive Entertainment Academy



**YOUR NEXT JOB.** At FIEA everything from our industry-based curriculum to our new MOCAP studio is geared toward teaching you what you need to know to become a successful video game producer, programmer or artist.

Earn an accredited Master's degree in 16 months while being mentored by industry veterans who have shipped more than 40 games and films. So tip your cap to our 95% placement rate and learn more at [www.fiea.ucf.edu](http://www.fiea.ucf.edu).

Producer, artist, programmer track | Fully accredited Master's degree | 95% placement rate | Financial aid available | [www.fiea.ucf.edu](http://www.fiea.ucf.edu)



Florida Interactive Entertainment Academy  
University of Central Florida  
500 West Livingston St.  
Orlando, FL 32801  
407-823-2121 • [info@fiea.ucf.edu](mailto:info@fiea.ucf.edu)





# BUILDING THE ALPHA G33K NATION

JOCKS NEED NOT APPLY

G33K



Sign them up for G33K 411, the magazine created especially for the future Alphageeks and Innovators of the world.

For every valid contact you provide, you and your friend will receive a "Building the Alpha G33K Nation" T-shirt.

And no, you can't let use your grandpa's address to get the shirt.

Go to [www.alphageeknation.com](http://www.alphageeknation.com) to get your shirt!

go to [www.alphageeknation.com](http://www.alphageeknation.com)

# Create the Game

Gaming Degree Programs for the Next Generation

## Game Art

Bachelor's Degree Program

## Game Development

Bachelor's Degree Program

## Game Design

Master's Degree Program



ANIMATION | DESIGN | ENTERTAINMENT BUSINESS | FILM | RECORDING ARTS | SHOW PRODUCTION | VIDEO GAMES | WEB

Master's | Bachelor's | Associate's Degrees

800.226.7625 • 3300 University Boulevard • Winter Park, FL 32792  
 Financial aid available to those who qualify • Career development assistance • Accredited University, ACCSCT

fullsail.edu

Online Programs Available



FULL SAIL  
UNIVERSITY

# MAKE MORE ENEMIES

Game Design at Vancouver Film School shows students how to make more enemies, better heroes, cooler levels, and tighter connections to the industry.

In just one year, you'll learn every aspect of game design. Your portfolio project is a playable video game.

VFS grads get snapped up by top companies like BioWare, Radical, Relic, and Ubisoft, and the *LA Times* named VFS a top 10 school "most favored by video game industry recruiters".



VFS

vfs.com/enemies

VFS student work by Moo Won Kim





Westchester  
Community  
College  
Center for the  
Digital Arts  
in Peekskill

Create  
Art *in the*  
Digital  
Age

914-606-7300  
www.sunywcc.edu/peekskill



Westchester  
Community College

State University of New York

CENTER FOR DIGITAL IMAGING ARTS AT BOSTON UNIVERSITY



capture  
imagination

3D ANIMATION  
+ INTERACTIVE MEDIA

GAME ART + CHARACTER ANIMATION

..... certificate programs

TWO CAMPUS LOCATIONS  
WALTHAM, MA & WASHINGTON, DC

Financial assistance and career services available.  
Now accepting applications. *Apply today!*



CDIABU.COM

## ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
Activision .....	48	Po ke tSoft.....	37
Blizzard Entertainment .....	39	R A D Ga me Tool s.....	C4
CCP.....	29	Seapine Software .....	17
Center for Dig ta ll mag i g.....	55	Sony Computer Entertainment.....	44
Crytek .....	13	Spiel Stud bs .....	38
Epic Ga mes .....	2 2 3	University of Advancing Technology.....	53
Eye tonics .....	C2	Uni versity of Centra l Florida .....	52
Full Sail University .....	54	Va ncouer Fi l m r school .....	54
Havok.....	21	Westchester Community Col ege.....	55
I m age Me trics .....	6	Xa t ment.....	3
Neversoft .....	40	XSENS Technologies.....	9
Perforce Software.....	25		

*Game Developer* (ISSN 1073-922X) is published monthly by United Business Media LLC, 600 Harrison St., 6th Fl., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as United Business Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is \$49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: \$69.95; all other countries: \$99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 4601 W. 6th St. Suite B, Lawrence, KS 66049. 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 *Game Developer* on any correspondence. All content, copyright *Game Developer* magazine/United Business Media LLC, unless otherwise indicated. Don't steal any of it.



# STAYING SANE

## YOU NEED HELP

**YOU PROBABLY THINK YOU'RE FINE. AFTER ALL, YOU DON'T RUN AROUND** the office making blib-blib-blib noises with your finger over your mouth—not in a crazy way, anyway. But let's not forget that game development can be a harrowing process, and that a long and difficult project can take an emotional toll. You're probably thinking of how some of your more unfortunate colleagues might not have survived the last product cycle with all of their marbles intact (or, more accurately, even fewer marbles than they started with). But before you get all high and mighty about how everyone else ought to be carted off to the asylum, take a moment to make sure that you aren't the one in need of the head-clearing effects a well-made straitjacket can impart.

It can often be hard to tell exactly how batty you've gone, because your perceptions of yourself will be as distorted as your impressions of anything else—but there are hints. Take a look at the symptoms below to see if you mightn't be in need of a little "alone time" to unwind from the project at hand (in a padded cell, perhaps):

### HALLUCINATIONS

» Sometimes it's hard to tell if you're experiencing them or not. You could have sworn you were in a meeting yesterday where everyone decided to cut that level. Wasn't everyone else in that meeting, too? You could have sworn we didn't decide to make an open-world game in the Unreal engine. Right? Because, I mean, that would be absurd ... right?

Other times, a hallucination will be very obvious. No, that designer's face is not made out of cottage cheese. No, the tangle of cables in back of your desk is not trying to eat you. No, your dev kit is not speaking to you in Deep Speech about opening a portal to the Far Realm and beginning a demon beast invasion of the human plane. I know—too bad.

### PARANOIA

» The way the front desk girl smiles at you when you come into work in the mornings. The way that artist said the creature you wanted would be "difficult" to animate. You can tell they're barely containing their derisive laughter at the very sight of you. And you're pretty sure they all go to big lunches together and talk about you—about how your time is coming. You're seriously considering setting up a tripwire across your office door that will send your bookshelf full of dictionary-sized tomes collapsing on the head of anyone who bursts in trying to fire you. If all you can think of is the delicious irony when the last thing that producer sees is Steve McConnell's *Rapid Development* hurtling toward his face, well, yes, that's amusing. But do seek help.

### ANGER MANAGEMENT ISSUES

» The hapless production intern ordered sandwiches again. The poor kid just wasn't thinking about it very carefully—about how those crappy, soggy six-inch subs murdered the team's morale every time they showed up. That didn't mean you had to scream bloody murder at the sight of that mushy bread and browning lettuce. You didn't need to go berserk and, in an apoplectic rage, flip that plastic platter out of his hands, his eyes as wide as saucers as sandwich pieces rained down on top of his head.

We snickered the first time. The second time it happened we started to wonder if you were okay. And the third time it wasn't even sandwiches, it



ILLUSTRATION BY JONATHAN KIM

was an enchilada tray! Man, seeing him standing there dumbfounded with the sauce and the cheese on him was super funny, but maybe also just the tiniest bit not totally right.

We would have brought this up earlier, but we were afraid that you would, you know, get angry.

### SUBSTANCE ABUSE

» It always starts innocuously enough. The bottle was for special late nights, when the team was there after nine and the build was humming along. Then it sort of became after eight. Then it sort of became after dinner, and then it sort of became with dinner. Then it sort of became the chaser to those beers or margaritas you'd order at lunch. Then it somehow replaced the hemoglobin in your blood such that it was necessary to have in you at all times to carry and distribute oxygen from your lungs. (Note: that is not actually possible.) As you should understand by now, drugs and alcohol won't fill the hole in your life ... try some happy pills from your doctor instead. The stuff they make these days is strong!

### SOBBING UNCONTROLLABLY

» Possibly you don't need me to tell you this, but I'll say it anyway. Breaking down and crying is a sign of problems. Not that I or anyone I know has ever done this, mind you. I'm just saying. On a theoretical level.

### YOU DECIDED TO MAKE VIDEO GAMES FOR A LIVING

» Well, there's my incontrovertible proof that you're crazy to begin with. Allow me to pull back the curtain, *Doctor Caligari* style, and reveal that you were already in the loony bin this whole time. Heyo! 🎮

---

**MATTHEW WASTELAND** is a pseudonymous game developer who has a fairly common first name. Email him at [mwasteland@gdmag.com](mailto:mwasteland@gdmag.com).

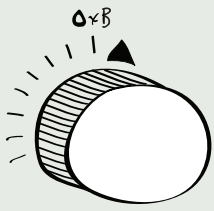


EVEN IF YOUR  
ROYALTY  
PROCESSING  
OFFICE IS  
CURRENTLY  
UNOCCUPIED,  
**your game can be Unreal.**



No matter what size your budget. No matter what type of game.  
Unreal can be your game engine. Email Mark Rein at [getunreal@epicgames.com](mailto:getunreal@epicgames.com)





TURN UP THE

# WICKED

AUDIO IN YOUR GAME WITH

# MILES 8

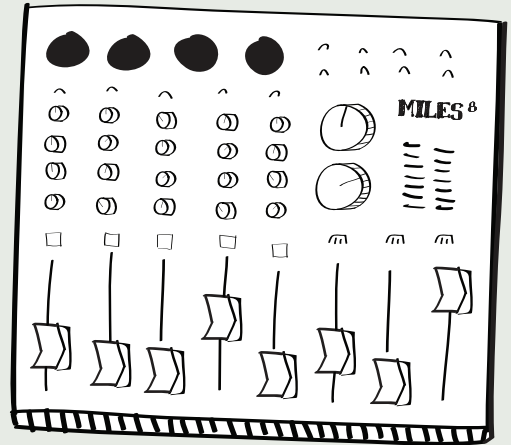
BRAND NEW!



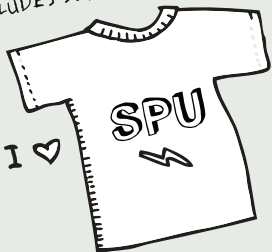
Our multichannel sound system features cool, new

high-level audio tools

+ the world's **FASTEST**



INCLUDES SUPPORT FOR THE PS3



**MP3** and Ogg DECODERS.

USING MILES 8 NOT ONLY MAKES YOU

WANT TO TURN IT UP, IT MAKES YOU rad!



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