



United Business Media

TOOL REVIEW AUTODESK MUDBOX 2009

MARCH 2009

# gamedeveloper

THE LEADING GAME

POSTMORTEM

UBISOFT MONTREAL'S

# FAR CRY 2

**CODERS GONE WILD**  
PROGRAMMING TRICKS  
FROM THE DARK SIDE

**LESS IS MORE**  
CUT YOUR TEETH ON  
SUBTRACTIVE DESIGN

**SMOKY MOUNTAINS**  
BUILDING TERRAIN  
USING PARTICLES





# EYETRONICS

## 3D



## SCANNING SOLUTIONS

**CUSTOMERS INCLUDE:**  
EA, MIDWAY, SONY, 2K,  
I/O INT, BLUE CASTLE,  
ROCKSTAR, UBISOFT



**GDC**  
**BOOTH**  
**6301NH**

EYETRONICS  
[www.eyetronics.com](http://www.eyetronics.com)



800.205.9808 EU +32.16.298309



30

## POSTMORTEM

### 30 UBISOFT MONTREAL'S FAR CRY 2

FAR CRY 2 had extremely lofty goals. The aim was to create a first-person shooter with an engaging and truly dynamic narrative, in a vibrant persistent living world. After three and a half years of development, creative director Clint Hocking shares the hits and misses of this fascinating and rather under-discussed franchise reboot.

*By Clint Hocking*

## DEPARTMENTS

**2 GAME PLAN** *By Brandon Sheffield*  
Spiele Über Alles

**4 HEADS UP DISPLAY**  
Global Game Jam's debut, and top 10 freeware iPhone games

**47 TOOL BOX** *By Mike de la Flor*  
Autodesk Mudbox 2009, and product news

**70 GAMESCAPE** *By Jeffrey Fleming*  
San Francisco Bay Area, California

**87 CAREER** *By Jill Duffy*  
The Career-Seeker's Guide to GDC

COVER ART: UBISOFT MONTREAL ART TEAM

## FEATURES

### 7 DIRTY CODING TRICKS

In this tell-all article, eight programmers share harrowing tales of last-minute hacks performed to ship or save game projects. These often humorous anecdotes contain a lot of lessons for programmers, but should also be quite amusing for anyone who's ever worked with one.

*By Mark Cooke, Noel Llopis, Austin McGee, David Dyrnerman, Mick West, Nick Waanders, and anonymous contributors—compiled by Brandon Sheffield*

### 15 GROUND FROM SMOKE

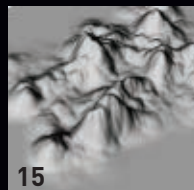
Why build general terrain the hard way, when you could have a particle system generate it for you? Korneliusz Warszawski proposes that particle-based terrain generation can save both coders and artists plenty of time.

*By Korneliusz Warszawski*

### 23 SUBTRACTIVE DESIGN

The idea behind subtractive design is to start with the core of what a game should be about, and then cut away everything that doesn't fit as the game progresses. This method, or something like it, is what led the ICO and PORTAL teams to such critical success.

*By David Sirlin*



## COLUMNS

**53 THE INNER PRODUCT** *By Noel Llopis*  
Writing Reusable Code

[PROGRAMMING]

**59 PIXEL PUSHER** *By Steve Theodore*  
The Boneyard

[ART]

**67 AURAL FIXATION** *By Jesse Harlin*  
Tools of the Trade

[SOUND]

**64 DESIGN OF THE TIMES** *By Soren Johnson*  
Asynchronous Design

[DESIGN]

**96 ARRESTED DEVELOPMENT** *By Matthew Wasteland*  
The Voice of Experience

[HUMOR]





## SPIELE ÜBER ALLES

**IN A NUMBER OF RECENT COLUMNS I'VE WRITTEN** about the potential of games to expand, and the need for the medium to evolve and mature. I've talked about how dialog and story should be more integrated into the development process, and how many lessons games could still learn from more traditional entertainment mediums. I do believe in all of that—but not at the expense of making good games. Before tackling any high falutin' artistic ideals, games have to first be good at being games.

### FASTER THAN A SPEEDING BULLET POINT

Consider **SONIC THE HEDGEHOG 2** on the Genesis as a very simple example. There's no denying that it's "just" a game—all you do is run, jump, and collect rings and powerups. And yet, this game affected a number of people. From the iconic music, to the slightly animated backgrounds, to the little birds you free when you defeat enemies, Sonic's world felt alive. These little flourishes help the game to really reach the player, but only because they're laid on top of such a solid structure. If the action of running and jumping weren't so smooth and fun, the extra graphical touches and music wouldn't have carried it through (search YouTube for "SONIC 2 early prototype" if you don't believe me). Then consider the modern **SONIC** games for current-gen consoles, which try to add combat, open worlds, sweeping story, and multiple characters, all in the name of filling bullet points, and turning **SONIC** into an "experience" rather than just a game.

A lot of modern games seem to want to be something they're not. A game shouldn't aspire to be a movie, or a novel, or a comic book, it should be a game, unless the aim is pure experimentation. That we are still using cutscenes, and not letting the player tell the story as they play is very frustrating to me, as it takes away the interactivity, which is the ultimate potential of games. (I should note that almost every game I've worked on has done the same, so I'm not innocent—it is very difficult to get out of the traditional structures in a time crunch.)

There is a lot that games can learn from movies, in terms of lighting, pacing, and editing. But these lessons should be applied to making better games, not to making games more like movies. The recent **TOMB RAIDER UNDERWORLD**, for example, has some lovely moments in it, from impressive set pieces to

clever puzzles. But it also wrests control from the player at regular intervals, to tell a story (through cutscenes) that I couldn't begin to parse, and which I eventually tried to find ways to skip. As Cliff Bleszinski said when I interviewed him about the first **GEARS OF WAR**, "I'm of the mind that you play games because you want to play, not because you want to watch."

### UNREALISTIC EXPECTATIONS?

As the game industry grows, and is lauded as a multi-billion dollar entertainment powerhouse, it can be easy—especially for publishers—to try to make more of games than what they are. At the core, are video games not meant to entertain interactively above all? **EARTH DEFENSE FORCE 2017** is an example of a game that ignores any sort of pretension and goes straight for the gameplay. It won't win any awards for its story, lack of bugs, or its camera use, but if you want to just blow up some giant monsters, you couldn't do much better.

I like games that are fun to play above all else, and I don't think I'm alone in that. If a game can provide an engaging narrative or smooth, bug-free play on top of this, then that will pretty much knock me out, icing-on-the-cake-wise. My old standbys here are **PORTAL** and **CALL OF DUTY 4**. They get the gameplay right first and foremost, and add a compelling narrative into the mix. Without that precision and visceral fun of play, nobody but the academics would be talking about a game like **PORTAL**, **ICO**, **FLOWER**, or any other game with an unconventional narrative or play style.

When I mentioned the subject of my editorial to production editor Jeffrey Fleming, he said it sounded like the anti-"games as art" manifesto, but that's not quite my intention. I believe that games can have artistry and be enjoyable and entertaining both. Subtractive design (see the article on page 23) may be one method of achieving this, and certainly iterative playtest cycles from an early stage have a tendency to polish games to a chrome finish.

My point is simply this: The enormous potential of games can only be fully realized when we are layering narrative, artistry, and thoughtful worldviews on top of games that are already fun without any of those things. If you have any comments about this article, come find me at GDC. I'll be the guy who looks like me.

—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 [bshffield@gdmag.com](mailto:bshffield@gdmag.com)

#### PRODUCTION EDITOR

Jeffrey Fleming [jfleming@gdmag.com](mailto:jfleming@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 Llopis [nllopis@gdmag.com](mailto:nllopis@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

#### MEDIA ACCOUNT MANAGER

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

#### GLOBAL SALES DIRECTOR

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

#### 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

### ADVERTISING PRODUCTION

#### PRODUCTION MANAGER Robert Steigleider

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

#### REPRINTS

#### WRIGHT'S REPRINTS

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

### THINK SERVICES

#### CEO THINK SERVICES Philip Chapnick

#### GROUP DIRECTOR Kathy Schoback

#### CREATIVE DIRECTOR Cliff Scorso

#### AUDIENCE DEVELOPMENT

#### GROUP DIRECTOR Kathy Henry [khenry@techinsights.com](mailto:khenry@techinsights.com)

#### DIRECTOR Kristi Cunningham

[kcunningham@techinsights.com](mailto:kcunningham@techinsights.com)

#### LIST RENTAL Merit Direct LLC t: 914.368.1000

#### MARKETING

#### SERVICES MARKETING COORDINATOR Laura Robison

[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



United Business Media



A close-up, back view of a person's head. On the back of the head, there is a glowing, rectangular circuit board with various components and lights. A black cable is plugged into the board, loops around the neck, and connects to a blue game controller. The background is a soft, light blue gradient.

From brain to game.

**The new 3DVIA Virtools 5 and 3DVIA MP will help you quickly create mind-blowing games.**  
Go from prototype to publisher-ready faster than ever before. You get all that speed without sacrificing the jaw-dropping, lifelike 3D graphics you expect from 3DVIA.

*FIND YOUR FAST @  
↳ [3DVIA.COM/FINDFAST](http://3DVIA.COM/FINDFAST)*

3dvia 

## GLOBAL GAME JAM'S STELLAR DEBUT

**O**n January 30th through February 1st, the first annual Global Game Jam made its successful debut. Game jams are usually local experiments in game development, in which a group of people get together and create games within 48 hours. Sometimes this leads to commercial products, sometimes to freeware, and sometimes the game will vanish into the ether, but participants generally come away feeling they've accomplished something.

The Global Game Jam was something new—a globally local

experience, in which each team worked locally, but shared progress via blogs, webcams (occasionally with kittens), and Flickr photosets for a sense of global community. With over 50 locations around the globe, over 1,600 attendees, and over 300 games created, the event was rather fabulously successful. Olivier Lejade of SOUL BUBBLES fame was the organizer of the Paris location, which was held in his company offices at Mekensleep.

Regarding the idea of having everyone work at the same time around the globe, Lejade shared his impressions:

"Having everyone do it on the same day created a nice sense of planetary event," he said, also adding that it helped get the attention of the media. "I also found it interesting to look for cultural artifacts in the resulting games. But mainly it solved lots of logistics problems to have everyone start around the same time and end around the same time. If the event had happened over a month with de-synchronized jams, we would have needed much stronger organization."

As far as the games released, "I think some of them will be fleshed out into proper games



Costa Rica's Global Game Jam.

PHOTO BY LAURA PARDO

(deservedly so) and out of these some will be released for free and some for profit," says Lejade. "If I were a publisher right now, I would think long and hard about what just happened. I think the GGJ gives a good glimpse of how stage gate publishing (see [\[cars-and-gardens-visualizing.html\]\(http://globalgamejam.org/games\)\) could work on a global scale. I believe there's an enormous potential waiting to be tapped here."](http://lostgarden.com/2007/02/rockets-</a></p></div>
<div data-bbox=)

It's important to note that all games created during the GGJ are owned by their respective creators, and are archived on the GGJ site at

<http://globalgamejam.org/games>. To commemorate everyone's hard work, and to perhaps inspire more developers to participate in the future, we present an abridged list of factoids from around the GGJ sphere, compiled by Ian Schreiber, one of the event's organizers. —Brandon Sheffield

## GGJ 2009 STATISTICS

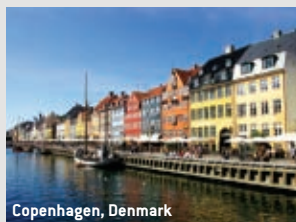
**Largest site:** Copenhagen, Denmark with 135 participants

**Smallest site:** Kyoto, Japan with 3 participants (including the organizer)

**Most dangerous site:** Charlotte, NC, which had a radioactive fire break out in the same building during the first night of the Jam. They were evacuated from the building into the freezing cold outside. Charlotte was also the most overbooked, with 21 advance signups and 48 attendees.

**Most hardcore site:** Tel Aviv, Israel, which had a participant whose wife was due during the Jam; he was developing until 4 a.m. His wife was helping out during the first day, and presumably would have participated through the whole thing had she not been due.

**Best prepared site:** Ankara, Turkey, where the organizer went around every few minutes reminding people to save their work in case of power failure. During



Copenhagen, Denmark

the Jam, someone plugged in a broken kettle and blew the power of the entire building, bringing things to a halt for 15 minutes. No work was lost in the process.

**Most distracted site:** Pittsburgh, PA, with five false fire alarms during the Jam.

**Site most likely to fail a constitution check:** Erlangen, Germany, which reportedly had no caffeinated beverages available on site, and also no windows.

**Most academically-oriented site:** Paris, France, which had a graduate student [non-participant] studying the group dynamics of the Jam for her thesis.

**Worst participant-to-signup ratio:** Ottawa, Canada, which had its registration numbers reduced by almost half due to the combination of inclement weather and a bus strike.

**Coollest space:** Raleigh, NC, which took place (in part) inside a mocap studio. Unfortunately, the mocap equipment had to be removed to make room for the Jam.

**Most likely site to have boosted attendance next year:** Copenhagen, Denmark, which boasted an open bar on site. (Honorable mention: Costa Rica, which has nearby beaches and sunshine in January.)

**Most gender-balanced site:** Savannah, GA and London, England tied with 20% female participation.

**Greatest percentage of left-handed people:** Santa Cruz, CA with 44%. (Most sites reported equal or greater numbers of southpaws when compared to females.)

**Best multitasking effort:** Hamar, Norway, where a single musician made music for all teams on site.

**Most hardcore site organizer:** Foaad, who not only organized the Santa Cruz, CA site but also provided tech support to the other 53 sites.

**Most common site organizer name:** Michael.

**Most common question during the Jam:** "Hey, can I ask a question?"

**Most common point of confusion during the Jam:** Whether to count organizers and disgruntled security guards when counting total numbers of participants.

**Second most common point of confusion:** Not knowing exactly how to re-gruntle a security guard.

**Most popular webcam:** Kittens, which generally kept at about 2500 views. We're not sure which site or which project team they worked on.





# TOP 10 FREEWARE IPHONE GAMES

Are you curious to test out the gaming capabilities of your iPhone or iPod Touch, but daunted by the thousands of choices in Apple's ever-growing iTunes App Store? Here's a list of games to check out as a starting point. They're fun, they're odd, and best of all, they're free. —Danny Cowan



## CUBE RUNNER

Developer: Andy Qua

CUBE RUNNER's massive online fanbase can testify to the fact that this classic app features one of the best uses of the iPhone's accelerometer functions. By tilting your iPhone left and right, you pilot a low-flying ship through an unending field of deadly cubes. Multiple difficulty levels, downloadable stages, and an engrossing soundtrack will keep you coming back for more.



## MAGNETIC JOE

Developer: Most Wanted Entertainment

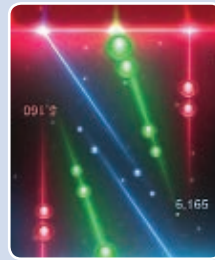
Originally a hit on mobile phone platforms, MAGNETIC JOE's entire 40 levels of addictive single-switch gameplay are available for free on the iPhone and iPod Touch. By touching the screen, the smiling spheroid Joe gets magnetized, attracting him to magnet blocks found in each level. Controlling Joe can be a harrowing task—carefully-timed presses are required in each danger-fraught stage.



## TOPPLE

Publisher: ngmoco

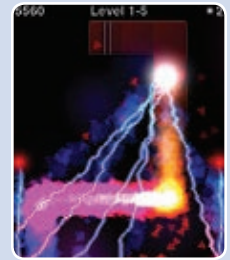
Have you ever tried to throw a game of TETRIS by dropping every block as fast as possible? TOPPLE is sort of like that. Battling against a strict time limit, you'll frantically stack ill-fitting puzzle pieces skyward in the hopes that your hastily-built structure will remain standing by the time you reach each level's required height.



## TAP TAP REVENGE

Developer: Tapulous

Fans of music simulation titles like BEATMANIA and FREQUENCY will find a lot to like in TAP TAP REVENGE, a rhythm-based game that challenges players to tap and shake their iPhones in time with a varied selection of music. Every week brings a new selection of downloadable tracks, with featured artists including the likes of Weezer, The Offspring, and Daft Punk, among many others.



## MAZEFINGER

Publisher: ngmoco

This one will push your hand-eye coordination to its limits. MAZEFINGER's mazes must be completed by dragging your finger from each level's starting point to its exit, in a clone of Namco's FLAMIN' FINGER arcade game. It's simple at first, but as the time limits get shorter and the mazes get more complex, your nerves will quickly fray as you race your finger across electrified floors and toward the next challenge.



## PAPIJUMP

Developer: Yohei Iwasaki

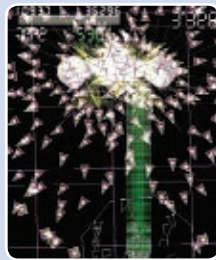
PAPIJUMP, like fellow early iPhone debut CUBE RUNNER, remains one of the platform's most popular games to date, even though its gameplay could hardly be simpler. Tilt your iPhone to guide Mr. Papi as he bounces ever upward with help from an irregular arrangement of platforms. Like CUBE RUNNER, games of PAPIJUMP often end in seconds, but you'll likely burn up more time than you'd expect in pursuit of new high scores.



## TAPDEFENSE

Developer: LL Group

The tower defense genre comes to the iPhone with TAPDEFENSE, a free title that features a surprising amount of depth for its price. You're in charge of defending the gates of heaven from the forces of hell by placing weapons in the army's path. Multiple tower upgrades and enemy types are featured in the 42-level campaign mode, and a challenge mode lengthens the experience with additional scenarios.



## rROOTAGE

Developer: Kenta Cho/Lazrhog Games

Kenta Cho's classic freeware shooter RROOTAGE goes mobile in an official iPhone port that manages to be entirely enjoyable, even without the precise controls of the original PC release. RROOTAGE is a "boss rush"-styled shooter that challenges players with tough enemies and bullet patterns, and multiple gameplay modes which recall classics like GIGA WING and IKARUGA.

## JELLY CAR

Developer: Timothy FitzRandolph

This puzzler places you in control of a squishy, jelly-like car which you must drive through a series of twisty, labyrinthine worlds. Your iPhone's accelerometer will get a workout here, as successful navigation involves rotating your car across shifting terrain and bending the in-game physics engine to your will.



## SPACE DEADBEEF

Developer: Yuji Yasuhara

This challenging horizontally scrolling shoot-em-up tasks you with simultaneously moving your ship and locking missiles onto approaching enemy craft...using only one finger. It's a difficult mechanic to grasp at first, but it's immensely satisfying to dodge a hailstorm of bullets and destroy entire enemy squadrons with a single swipe of your finger.



# BLADE

Enabling everyone to  
create, share & play!

Blade3D is a full featured  
development platform enabling  
the creation of games with a  
cost effective subscription model.

BladeMarketplace is integrated  
with Blade3D, providing the  
ability for users to buy, sell, and  
share assets.

BladeServices provides creative  
development resources for  
game studios and publishers  
around the world.

VISIT US AT GDC...

**BOOTH #6306**  
**North Hall**

Try Blade3D today!

GAME CREATION PLATFORM

## BLADE3D



BUY, SELL, SHARE ASSETS

## MARKETPLACE



CREATIVE DEVELOPMENT

## SERVICES



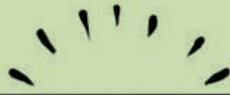
[www.BladeGamesWorld.com](http://www.BladeGamesWorld.com)



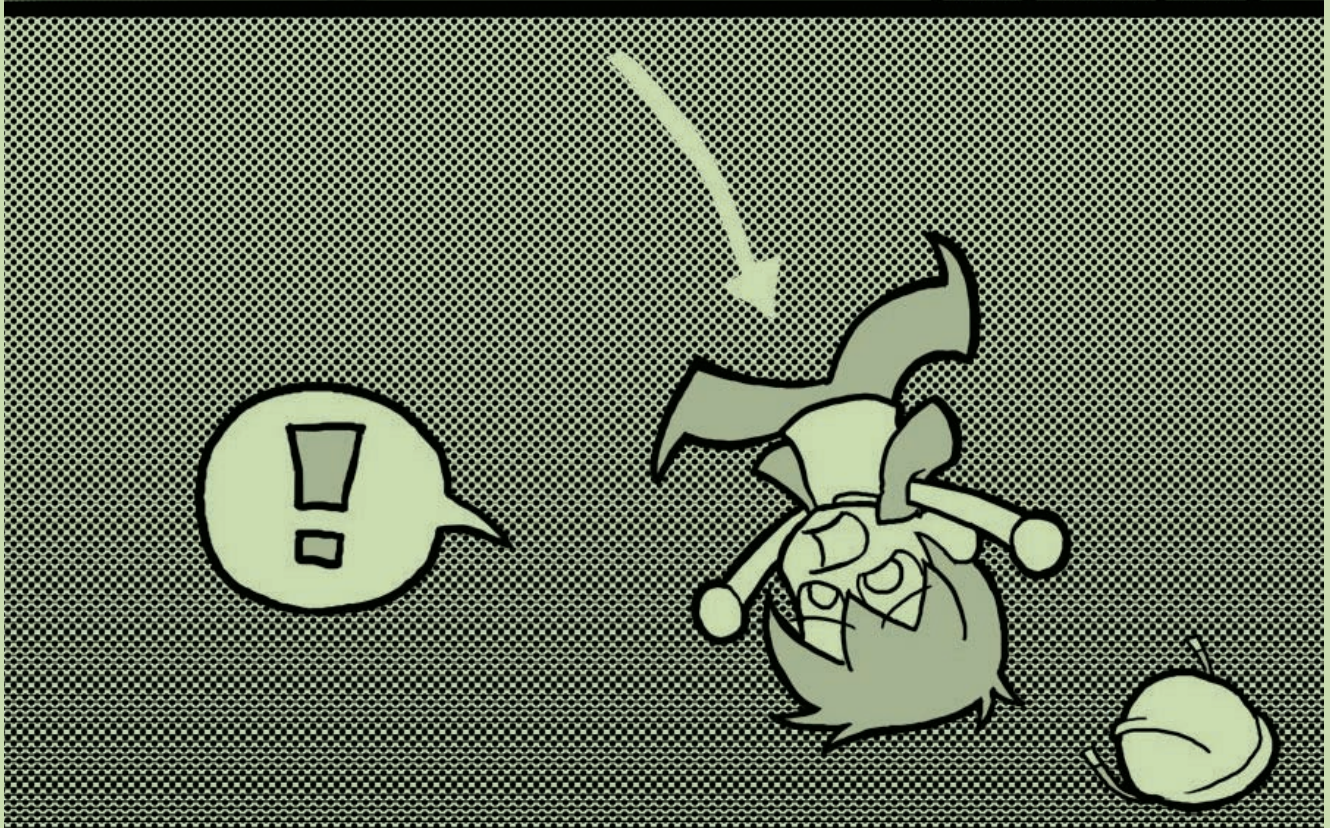
# DIRTY

## CODING TRICKS\*

////////// Programmers are often methodical and precise beasts who do their utmost to keep their code clean and pretty. But when the chips are down, the perfectly-planned schedule is shot, and the game needs to ship, “getting it done” can win out over elegance. In a case like this, a frazzled and overworked programmer is far more likely to ignore best practices, and hack in a less desirable solution to get the game out the door. We have here compiled nine testimonials from working developers, which chronicle times when they weren’t quite able to follow the script and had to pull some tricks to save a project. —Brandon Sheffield



CONTINUED ON PG 8



ILLUSTRATIONS BY JON "PERSONA" KIM

**\*THIS IS WHAT HAPPENS WHEN CODERS STOP BEING POLITE...AND START GETTING REAL**



CONTINUED FROM PG 7

## SHOOT ME FROM MY GOOD SIDE

**A**round four years ago I was working as a programmer on a multiplatform PlayStation 2, Xbox, and GameCube release. As development was coming to a close it should come as no surprise that some code hacks started creeping into the game to get the title shipped. The PS2 version in particular had a late, extremely hard to track down issue: A long soak test of the player character standing still in the first level of the game would cause it to periodically crash. Unfortunately, the crash was limited to disc-only retail builds that included no debugging information. With fears of a Sony technical requirement check (TRC) failure looming we worked hard to find a solution.

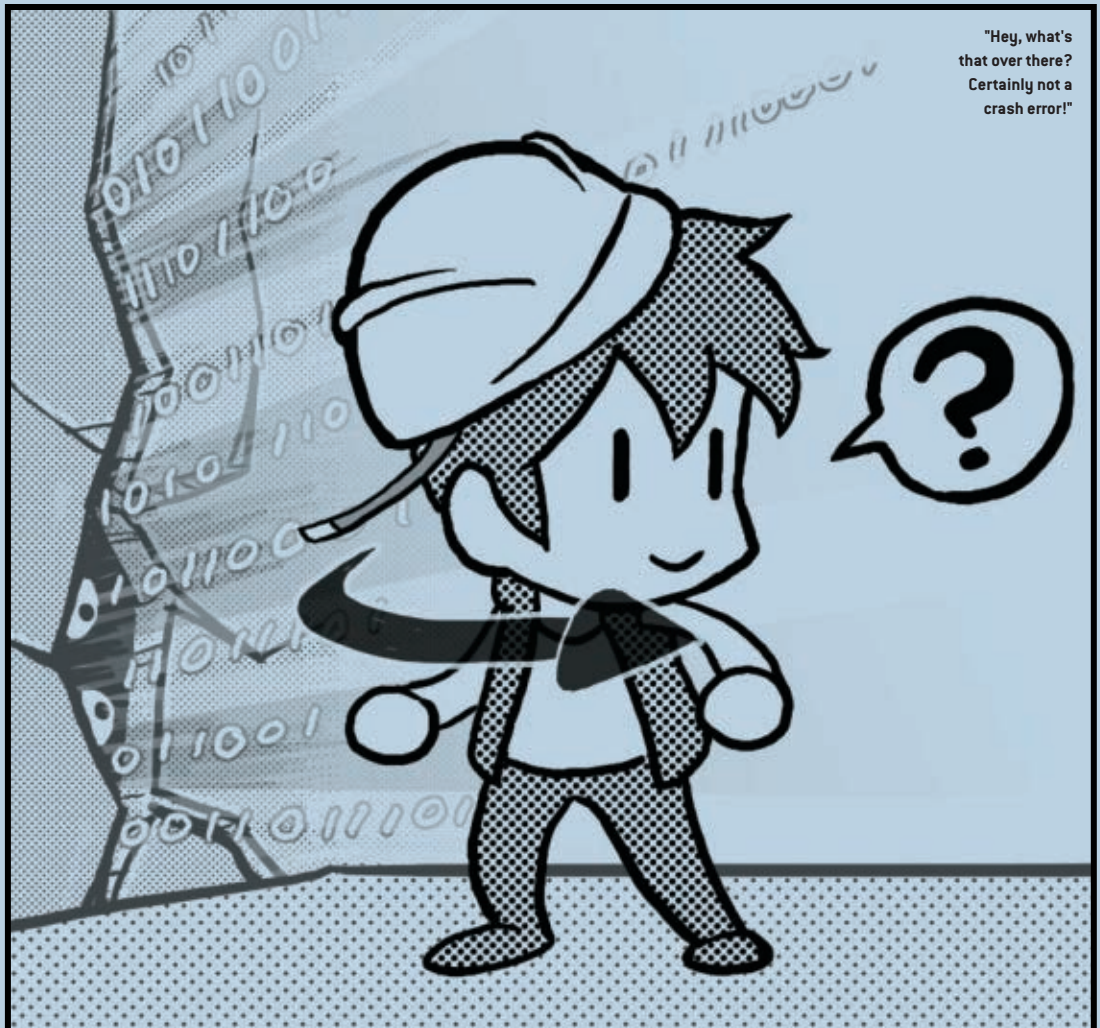
To narrow down the issue we continuously rendered a border around the edge of the screen with a different color for different sections of the code. For example, blue represented render setup, and green was the player control update. Because the crash was intermittent, the lead engineer on the title and I would manually burn a bunch of discs and start them up on an array of PS2s. (Keep in mind this was an independent developer, with

no IT team or fancy disc burning machines.) The test cycle—between making a code change, burning discs, deploying, and waiting to whittle down some approximation of where the crash was—took hours. It was the eleventh hour of development and there was a fixed number of these test cycles we could conduct.

Between stretches of passing the time over the weekend in the office with *WORLD OF WARCRAFT* while waiting for a crash, someone noticed that the game didn't crash if you rotated the camera 90 degrees to the right. Initially, the programming team correctly waved this off as some kind of fluke that was masking the underlying bug. This isn't the type of "fix" that would repair the root cause of a crash. That didn't stop us from shipping it though! As we ran out of time, we created the final PS2 disc with a rotated camera at level start—the other two platforms were already on track—and submitted everything. The game passed all the platform holders' TRC tests and was shipped to market on time.

I wouldn't call this a "coding trick," but it was definitely "dirty." The mysticism of the fix is disconcerting to this day, but thankfully I never heard of any reported user issues.

—Mark Cooke







## IDENTITY CRISIS

This scene is familiar to all game developers: It's the day we're sending out the gold candidate for our Xbox 1 game. The whole team is playtesting the game all day long, making sure everything looks good. It's fun, it's solid, it's definitely a go in our minds.

In the afternoon, we make the last build with the last few game-balancing tweaks, and do one last playthrough session when disaster strikes: The game crashes hard! We all run to our workstations, fire up the debugger, and try to figure out what's going on. It's not something trivial, like an assert, or even something moderately hard to track down, like a divide by zero. It looks like memory is garbage in a few places, but the memory reporting comes out clean. What's going on?

One dinner and many hours later, our dreams of getting out on time shattered, we manage to track it down to one data file being loaded in with the wrong data. The wrong data? How's that possible? Our resource system boiled down every asset to a 64-bit identifier made out of the CRC32 of the full filename and the CRC32 of all the data contents. That was also our way of collapsing identical resource files into a single one in the game. With tens of thousands of files, and two years of development, we never had a conflict. Never.

Until now, that is.

It turns out that one of the innocent tweaks the designers had checked in that afternoon made it so a text file had the exact same filename and data CRC as another resource file, even though they were completely different!

Our hearts sank to our feet when we recognized the problem. There's no way we could change the resource indexing system in such a short period of time. Even if we pulled an all-nighter, there was no way to know for sure that everything would be stable in the morning.

Then, as quickly as despair swept over us, we realized how we could fix this on time for the gold candidate release. We opened up the text file responsible for the conflict, added a space at the end, and saved it. We looked at each other with huge grins on our faces and said:

"Ship it!"

—Noel Llopis

## DRIVING UNDER THE INFLUENCE

I've been tasked with taking care of a system dealing with vehicles, which is a fairly major part of our current game. Fortunately, we were able to grab the majority of the code from another studio. Unfortunately, the code isn't always perfect or well written, as is the perception with most code you don't write personally. I happened to find this nice bit of code that was simply trying to get a variable from the engine loop, but was going about it in the most backward way possible (See Listing 1).

The funniest part about this chunk of incredibly gross code is that there was a simple function on that object to get the frame count. Even if there weren't, the person who wrote this code could have easily added one himself! Needless to say, this code was not added to my game, nor to the game this code was taken from, as soon as I pointed it out. If this isn't an example of why code reviews are good, I don't know what is.

—Austin McGee

### LISTING 01

#### DRIVING UNDER THE INFLUENCE

```
//*****
//*****
// Function: AGameVehicle::Debug_GetFrameCount
//
//! A very hacky method to get the current frame
// count; the variable is protected.
//!
//! \return The current frame number.
//*****
//*****
UINT AGameVehicle::Debug_GetFrameCount()
{
    BYTE* pEngineLoop = (BYTE*)&GEngineLoop;
    pEngineLoop += sizeof( Array<FLOAT> ) + sizeof(
    DOUBLE );
    INT iFrameCount = *((INT*)pEngineLoop);
    return iFrameCount;
}
```

## 10-TATIVE CODE

Back at [company X], I think it was near the end of [the project], we had an object in one of the levels that needed to be hidden. We didn't want to re-export the level and we did not use checksum names. So right smack in the middle of the engine code we had something like the following. The game shipped with this in.

```
if( level == 10 && object == 56 )
{
    HideObject();
}
```

Maybe a year later, an artist using our engine came to us very frustrated about why an object in their level was not showing up after exporting to what resolved to level 10. I wonder why?

—Anonymous

## ALL SIGNS POINT TO "NO-NO"

This issue came up during Raven Software's new WOLFENSTEIN, as I was setting up controller support for the 360. It turned out that for the Live integration, we had to know which controller was sending input events. The DOOM 3 input code we were using was derived almost straight from QUAKE 3, so it was a pretty simple system.

In the existing event system, each event was passed around with two integer arguments and a void pointer in case you needed extra parameters. So the goal was to associate a controller ID with each input event. No problem—just pack the controller ID into one of the event's integer arguments, right? Of course, they were both already in use.

Then came the next idea: Let's just use our fast no-fragmentation per-frame allocator to reserve some memory, plop the controller id in there, and then pass a pointer to it in the event's pointer slot.

It turns out that the event system would take it upon itself to free() the event's void pointer after processing the event. Of

course this was totally incompatible with the multi-heap approach that we were using. Moreover, there were a few legacy DOOM 3 chunks that actually relied on the event code calling `free()`, so we couldn't just remove the call without non-trivial changes across the codebase. What about adding a third integer parameter? Well, that would involve changing several hundred function calls.

The deadline is looming, I can't spend much more time on this. So, I did the unthinkable—I packed the controller id into the pointer parameter. I marked it as a horrible hack in a 4-line all-caps comment, and checked it in. This worked fine for a while, and lasted until the whole input system was replaced with something a little better down the line.

—David Dynerman

## VELCRO SNEAKS

Here's a tale from a project in the early PS2 days. We had a ton of collision/bounding issues that were largely solved at the last minute by a rewrite of character collision to use a "collider" model—a stack of spheres that were way better at resolving bumps vs. boxes than our hierarchical tree of oriented boxes ever could be (ah, the Land Before Havok). However, we had a rare bug for ages that was pretty maddening—we called it Velcro-ing—where every now and then the player character would be up against a wall and sliding along it, but then a sphere of the collider would suddenly decide it was on the wrong side of the wall and wouldn't let you get away from it (i.e. you'd be stuck to the wall's surface).

This was one of those god-awful "sounds easy" bugs—just figure out why the sphere thinks it's on the wrong side, and fix it. Problem is, you had to catch what happened in all the collision resolutions before it happened to figure this one out, and when you'd drop in things like handy conditional breakpoints to look for a weird response, or a push that made it get confused where it was on a skinny box or something similar, it'd bog the frame rate down and the issue simply wouldn't happen. (The real fix to this problem took blood, sweat, tears, and I think other fluids not usually required, but that is another story.)

Since we were trying to submit, and regularly would get this issue as a blocking bug, one "solution" was to buff out the bounding on whatever was causing the problem, which was guaranteed to make it not happen, but was clearly not the right fix. We'd forward the bug to our art team, they'd fix up that case, and in the ensuing turn-around time, we bought ourselves precious days to keep trying to track down the actual deep, subtle problem. Testers would typically send back a "player hit invisible wall, player must not hit invisible wall" bug of roughly equal severity (bounding the wall tightly again fixed that, as Velcro-ing was really rare and hard to reproduce anyway), but after a couple cycles of this as we cleaned up the rest of the game, the blood, sweat, and tears fix was eventually found and off we went to duplication and shelves.

This wasn't the proudest cycle of bug-turn-around-stalling, but it got the upper-ups off our backs, because we could say "oh, no, we released that to testing yesterday," and keep scrambling to figure out that damned flipping issue.

—Anonymous



## MEET MY DOG, PATCHES

There's an old joke that goes something like this:

Patient: "Doctor, it hurts when I do this."

Doctor: "Then stop doing it."

Funny, but are these also wise words when applied to the right situation? Consider the load of pain I found myself in when working on a conversion of a 3D third person shooter from the PC to the original PlayStation.

Now, the PS1 has no support for floating point numbers, so we were doing the conversion by basically recompiling the PC code and overloading all floats with fixed point. That actually worked fairly well, but where it fell apart was in collision detection. The level geometry that was supplied to us worked reasonably well in the PC version of the game, but when converted to fixed point, all kinds of seams, T-Junctions and other problems were nudged into existence by the microscopic differences in values between fixed and floats. This problem would manifest itself by the main character (called "Damp") simply falling through those tiny holes, into the abyss below the level.

We patched the holes we found, tweaking the geometry until Damp no longer fell through. But then the game went into test at the publisher, and suddenly a flood of "falling off the world" bugs were delivered to us. Every day a fresh batch of locations were found with these little holes that Damp could slip through. We would fix that bit of geometry, then the next day they would send us ten more. This went on for several days. The publisher's test department employed one guy whose only job was to jump around the world for ten hours a day, looking for places he could fall through.

The problem here was that the geometry was bad. It was not tight and seamless geometry. It worked on the PC, but not on the PS1, where the fixed point math vastly magnified the problems. The ideal solution was to fix the geometry to make it seamless.





# 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 was inducted into the Hall of Fame this year.

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:

### Game Developers Conference

San Francisco, CA  
March 23-27, 2009

### Triangle Game Conference

Raleigh, NC  
April 29-30, 2009

### Electronic Entertainment Expo

Los Angeles, CA  
June 2-4, 2009

Please email:  
mrein@epicgames.com  
for appointments.



## MOD AUTHORS SHINE IN \$1 MILLION INTEL MAKE SOMETHING UNREAL CONTEST

Epic Games recently announced Phase 2 winners of the \$1 Million Intel Make Something Unreal Contest, the ongoing mod competition for *Unreal Tournament 3*.

Categories comprising Phase 2 include Best Game Mod, Best New Character or Customization Pack, Best New Vehicle, Best Warfare Level, Best Deathmatch or CTF Level, Best Use of Physics, Best vCTF Level, Best Graphics in Map, and Best Machinima.

The first place winner in the Best Game Mod category is Prometheus, a single-player first-person puzzle mod. Prometheus has a unique rewind feature that enables the player to reverse events and spawn a new version of himself as ghost characters repeat actions that have already been performed. The goal is to cooperate with one's past identities while anticipating what future selves will need to do in order to complete objectives.

Second place honors in the Best Game Mod category go to The Haunted, a multiplayer third-person shooter in which one person controls a "haunted" character whose goal is to fend off onslaughts of undead creatures and earn points by staying alive.

Other players can control the undead, navigate the map unseen by the haunted character, and spawn different enemies or place traps in the level. Each type of threat varies in value, and a meter regulates what can be deployed, so it's important to be selective with attacks.



The Ball: Oztoc mod based on *Unreal Tournament 3*

The Ball: Oztoc takes home third place for Best Game Mod. The Ball is a first-person, single-player puzzle game in which the player manipulates a large steel ball with an Impact Gun to solve puzzles and overcome obstacles in an Aztec-themed temple.

The Ball: Oztoc also placed first in Phase 2 for Best Use of Physics, and its predecessor, The Ball: Pehua, won

Best Use of Physics and Best Gametype in Phase 1 of the contest.

Fourth place for Best Game Mod goes to Angels Fall First: Planetstorm, a team-based planetary assault mod in which battles rage in space and on land. Maps are linked in pairs so players can easily take the fight from the space theater to ground locations and vice versa.



Angels Fall First: Planetstorm *UT3* mod

Snowreal, the fifth place winner in the Best Game Mod category, is a third-person snowboarding mod in which players have the option to race against each other over a variety of maps or compete in trick competitions. Snowreal also features a remote camera that lets players record stunts and create trick videos.

Historically, Epic's mod contest has provided benefits not only for its entrants but also for the game industry as a whole. Past winners of the contest have gone on to ship commercial games using the Unreal Engine, and several former members of the Unreal mod community have found jobs either with Epic or with companies licensing the Unreal Engine.

Epic will begin accepting Phase 3 entries in April, and the submission window will close on May 15, 2009.

Anyone interested in checking out the hundreds of *UT3* mods submitted thus far or participating in the contest can visit [www.makesomethingunreal.com](http://www.makesomethingunreal.com).

Many top mod teams have openings, so it's not too late to get involved and make something Unreal!



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 many faces of Nick Waanders.



However, this was a vast task, impossible to do in the time available with our limited resources, so we were relying on the test department to find the problem areas for us.

The problem with that approach was that they never stopped finding them. Every day bought more pain. Every day new variants of the same bug. It seemed like it would never end.

Eventually the penny dropped. The *real* problem was not that the geometry had small holes in it. The problem was that Damp fell through those holes. With that in mind, I was able to code a very quick and simple fix that looked something like:

```
IF (Damp will fall through a hole()) THEN  
Don't do it
```

#### LISTING 01

### MEET MY DOG, PATCHES

```
damp_old = damp_loc;  
move_damp();  
if (NoCollision())  
{  
damp_loc = damp_old;  
}
```

#### LISTING 03

### MEET MY DOG, PATCHES

```
if (damp_aliencoll != old_aliencoll &&  
strcmpi("X4DOOR",damp_aliencoll->enemy->ename)==0 &&  
StartArena == 6 && damp_loc.y<13370)  
{  
damp_loc.y = damp_old.y; // don't let damp ever  
touch the door.. (move away in the x and y)  
damp_loc.x = damp_old.x;  
damp_aliencoll = NULL; // and say thusly!!!  
}
```

The actual code was not really much more complex than that (see Listing 2).

In one swoop a thousand bugs were fixed. Now instead of falling off the level, Damp would just shudder a bit when he walked over the holes. We found what was hurting us, and we stopped doing it. The publisher laid off their “jump around” tester, and the game shipped.

Well, it shipped eventually. Spurred on by the success of “if A=bad then NOT A”, I used this tool to patch several more bugs—which nearly all had to do with the collision code. Near the end of development the bugs became more and more specific, and the fixes became more and more “Don’t do thispreciseandexacting” (see Listing 3, actual shipped code).

What does that code do? Well, basically there was a problem with Damp touching a particular type of door in a particular level in a particular location, rather than fix the root cause of the problem, I simply made it so that if Damp ever touched the door, then I’d move him away, and pretend it never happened. Problem solved.

Looking back I find this code quite horrifying. It was patching bugs and not fixing them. Unfortunately the real fix would have been to go and rework the entire game’s geometry and collision system specifically with the PS1 fixed point limitations in mind. The schedule was initially aggressive, so we always seemed close to finishing, so the quick patch always won over the comprehensive, expensive fix. But it did not go well. Hundreds of patches were needed, and then the patches themselves started causing problems, so more patches were added to turn off the patches in hyper-specific circumstances. The bugs kept coming, and I kept beating them back with patches. Eventually I won, but at a cost of shipping several months behind schedule, and working 14 hour days for all of those several months.

That experience soured me against “the patch.” Now I always try to dig right down to the root cause of a bug, even if a simple, and seemingly safe, patch is available. I want my code to be healthy. If you go to the doctor and tell him “it hurts when I do this,” then you expect him to find out *why* it hurts, and to fix that. Your pain and your code’s bugs might be symptoms of something far more serious. The moral: Treat your code like you would want a doctor to treat you.

—Mick West





## YOU WOULDN'T LIKE ME WHEN I'M ANGRY

I once worked at THQ studio Relic Entertainment on *THE OUTFIT*, which some may remember as one of the earlier games for the Xbox 360. We started with a PC engine [single-threaded], and we had to convert it to a complete game on a next-gen multi-core console in about 18 months. About three months before shipping, we were still running at about 5 FPS on the 360. Obviously this game needed some severe optimization.

When I did some performance measurements, it became clear that as much as the code was slow and very “PC,” there were also lots of problems on the content side as well. Some models were too detailed, some shaders were too expensive, and some missions simply had too many guys running around. It’s hard to convince a team of 100 people that the programmers can’t simply “fix” the performance of the engine, and that some of the ways people had gotten used to working needed to be changed. People needed to understand that the performance of the game was everybody’s problem, and I figured the best way to do this is with a bit of humor that had a bit of hidden truth behind it.

The solution took maybe an hour. A fellow programmer took four pictures of my face—one really happy, one normal, one a bit angry, and one where I am pulling my hair out. I put this image in the corner of the screen, and it was linked to the frame rate. If the game ran at over 30fps, I was really happy, if it ran below 20, I was angry. After this change, the whole FPS issue transformed from, “Ah, the programmers will fix it.” to, “Hmm, if I put this model in, Nick is going to be angry! I’d better optimize this a little first.” People could instantly see if a change they made had an impact on the frame rate, and we ended up shipping the game at 30fps.

—Nick Waanders

## THE PROGRAMMING ANTIHERO

I was fresh out of college, still wet behind the ears, and about to enter the beta phase of my first professional game project—a late-90s PC title. It had been an exciting rollercoaster ride, as projects often are. All the content was in and the game was looking good. There was one problem

though: We were way over our memory budget.

Since most memory was taken up by models and textures, we worked with the artists to reduce the memory footprint of the game as much as possible. We scaled down images, decimated models, and compressed textures. Sometimes we did this with the support of the artists, and sometimes over their dead bodies. We cut megabyte after megabyte, and after a few days of frantic activity, we reached a point where we felt there was nothing else we could do. Unless we cut some major content, there was no way we could free up any more memory. Exhausted, we evaluated our current memory usage. We were still 1.5 MB over the memory limit!

At this point one of the most experienced programmers in the team, one who had survived many years of development in the “good old days,” decided to take matters into his own hands. He called me into his office, and we set out upon what I imagined would be another exhausting session of freeing up memory.

Instead, he brought up a source file and pointed to this line:

```
static char buffer[1024*1024*2];
```

“See this?” he said. And then deleted it with a single keystroke. Done!

He probably saw the horror in my eyes, so he explained to me that he had put aside those two megabytes of memory early in the development cycle. He knew from experience that it was always impossible to cut content down to memory budgets, and that many projects had come close to failing because of it. So now, as a regular practice, he always put aside a nice block of memory to free up when it’s really needed.


He walked out of the office and announced he had reduced the memory footprint to within budget constraints—he was toasted as the hero of the project.

As horrified as I was back then about such a “barbaric” practice, I have to admit that I’m warming up to it. I haven’t gotten into the frame of mind where I can put it to use yet, but I can see how sometimes, when you’re up against the wall, having a bit of memory tucked away for a rainy day can really make a difference. Funny how time and experience changes everything. ❖

—Noel Llopis

Experience a New Kind of Gameplay !

# Bot Colony™



"Hello!  
How can I help you?"  
"I'd like to rent a room,  
please."  
"How long are you  
staying with us?"  
"Four days."  
"Sea view or mountain  
view?"

**AI that talks!  
Just ask...**

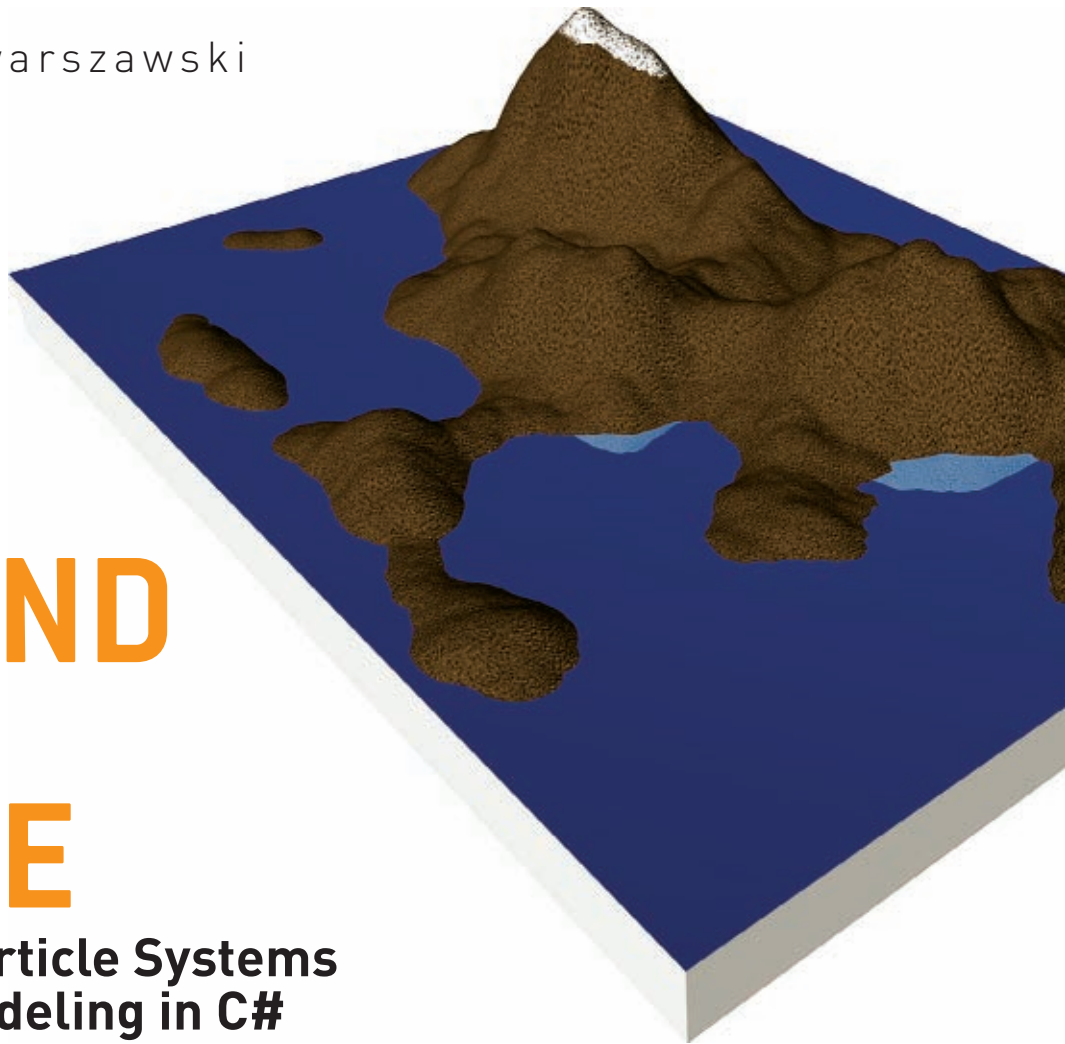
**Bot Colony™** is the first game ever where you can converse freely with the characters. You simply speak in free-form, unrestricted English to the characters, and they reply using speech. As you speak, the characters learn new concepts and facts from you, and use them in the game.

See unrestricted English conversation between player and characters at GDC, booth 5016 NH

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







# GROUND FROM SMOKE

## //// Using Particle Systems for Terrain Modeling in C#

Generating artificial terrain is a big part of modeling a virtual environment. Expanses of artificial terrain are needed in computer simulations for military training, virtual reality systems, and video games, particularly open-world games.

Usually in order to get an ideal virtual landscape, developers have to spend a lot of time manually working on polygon-net deformations. But it is possible to achieve an equally good outcome much faster, using an automated technique such as midpoint displacement, fault formation, or in the case of this article, a particle systems-based approach, shown using C# in the sample code.

### PARTICLE SYSTEMS

Originally, particle systems were developed as an efficient alternative for modeling in real time various kinds of natural objects that would be described as irregular, dynamic, or hard to define, such as clouds, smoke, explosions, fog, dust, and fields of grass or wheat.

Just as in a more traditional approach, the key elements in any particle system used for terrain modeling are: an emitter or set of emitters, the defined algorithm workspace, and a collection of parameterized particles.

Before generating a landscape, it is important to define the bounds of the system workspace. Any particles outside this space must be automatically removed to prevent them from endlessly flying around the virtual space.

**KORNELIUSZ WARSZAWSKI** is part of the faculty of Electrical Engineering, Computer Science and Telecommunications department at University of Zielona Gora, Poland. Email him at [kwaszawski@gdmag.com](mailto:kwaszawski@gdmag.com).

### LISTING 01

#### SAMPLE OF THE EMITTERSTRUCTURE

```
// used properties:
// -----
// Position - emitter
// position in 3D space
//
// (structure of three
// double variables X, Y
// and Z)
// Height - height of
// the emission window
// Width - width of
// the emission window

struct TEmitter
{
    private TCoords
    Position;

    private double
    Height;

    private double
    Width;

    /// <summary>
    THE EMITTER MAIN
    CONSTRUCTOR </summary>
    /// <param
    name="position">
    emitter position in 3D
    space </param>
    /// <param
    name="height"> height
    of the emission window
    </param>
    /// <param
    name="width"> width of
    the emission window
    </param>
    public
    TEmitter(TCoords
    position, double
    height, double width)
    {
        Position =
        position;
        Height = height;
        Width = width;
    }
    /* rest of the code
    (if needed) */
}
```

Next, we need to place the emitter or set of emitters. In the code and figures, I'm using just one. The emitter should be positioned above, and some distance from, the terrain surface. I recommend putting it above the central point of the landscape.

After setting up the emitter, we have to define the emission window size, which dictates how new particles will be ejected into the environment. Each

## LISTING.02

SAMPLE OF  
THE PARTICLE  
STRUCTURE

```

// used properties:
// -----
// Position - particle position
// in 3D space
// (structure of three
// double variables: X, Y and Z)
// Angle - current directional
// angles
// (structure of two
// double variables: Alpha and Beta)
// Size - size of the particle
// Speed - speed of the particle

struct TParticle
{
    private TCoords Position;
    private TAngle Angle;
    private double Size;
    private double Speed;

    /// <summary> THE PARTICLE
    MAIN CONSTRUCTOR </summary>
    /// <param name="position">
    current particle position in 3D
    space </param>
    /// <param name="angle">
    current directional angle </
    param>
    /// <param name="size"> size
    of the particle </param>
    /// <param name="speed"> speed
    of the particle </param>
    public TParticle(TCoords
    position, TAngle angle, double
    size, double speed)
    {
        Position = position;
        Angle = angle;
        Size = size;
        Speed = speed;
    }

    /* rest of the code (if
    needed) */
}

```

## LISTING.03

## PARTICLES DOWNFALL WITH PSEUDORANDOM DIRECTION CALCULATIONS

```

/* rest of the code (if needed) */

// used variables:
// -----
// range - range for the particle displacement
// Xmove - chance for particle displacement in X axis
// Ymove - chance for particle displacement in Y axis
// Zmove - chance for particle displacement in Z axis
// random - pseudo-random number generator
// pCol - collection of the particles
// PARTICLES - number of particles in a collection

for (int i = 0; i < PARTICLES; i++)
{
    // next line was inserted only for code readability
    double range = pCol[i].Speed; // we used here the Speed
    particle property

    int Xmove = random.Next(1000);
    int Ymove = random.Next(1000);
    int Zmove = random.Next(1000);

    // we simulate 25% chance for particle displacement for
    both direction
    // and 50% for no displacement in X axis
    if (Xmove > 250)
    {
        if (Xmove < 500)
        {
            // increase position in X axis
            pCol[i].Position.X += 1.0 * range;
        }
    }
    else
    {
        // decrease position in X axis
        pCol[i].Position.X -= 1.0 * range;
    }

    // we simulate 20% chance for particle displacement up
    // 30% for no displacement
    // and 50% for down displacement in Y axis (gravity
    factor)
    if (Ymove > 200)
    {
        if (Ymove < 300)
        {
            // increase position in Y axis
            pCol[i].Position.Y += 1.0 * range;
        }
    }
    else
    {
        // decrease position in Y axis
        pCol[i].Position.Y -= 1.0 * range;
    }

    // we simulate 25% chance for particle displacement for
    both direction
    // and 50% for no displacement in Z axis
    if (Zmove > 250)
    {
        if (Zmove < 500)
        {
            // increase position in Z axis
            pCol[i].Position.Z += 1.0 * range;
        }
    }
    else
    {
        // decrease position in Z axis
        pCol[i].Position.Z -= 1.0 * range;
    }
}

/* rest of the code (if needed) */

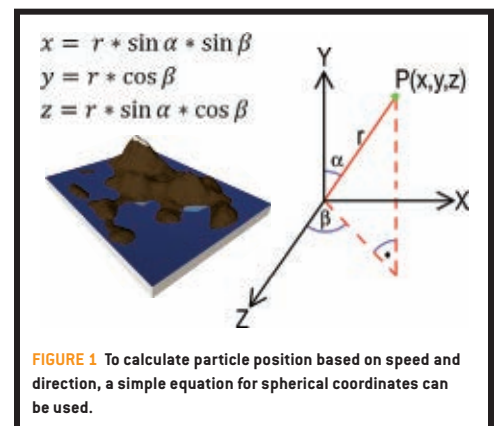
```

particle's starting position must be inside the emission window's borders. A sample of the emitter structure is shown in Listing 1.

Once the emitter is defined, the next step is to select some attributes for particle collection. The attributes will affect the characteristics of the final terrain structure that we generate. For simple surfaces, the list of attributes would be limited to a few. I prefer, as is shown in Listing 2, to use a current particle's position in virtual space, its direction, current speed (meaning how fast a particle approaches the terrain surface), and size, which determines a particle's effect on the heightfield modifications.

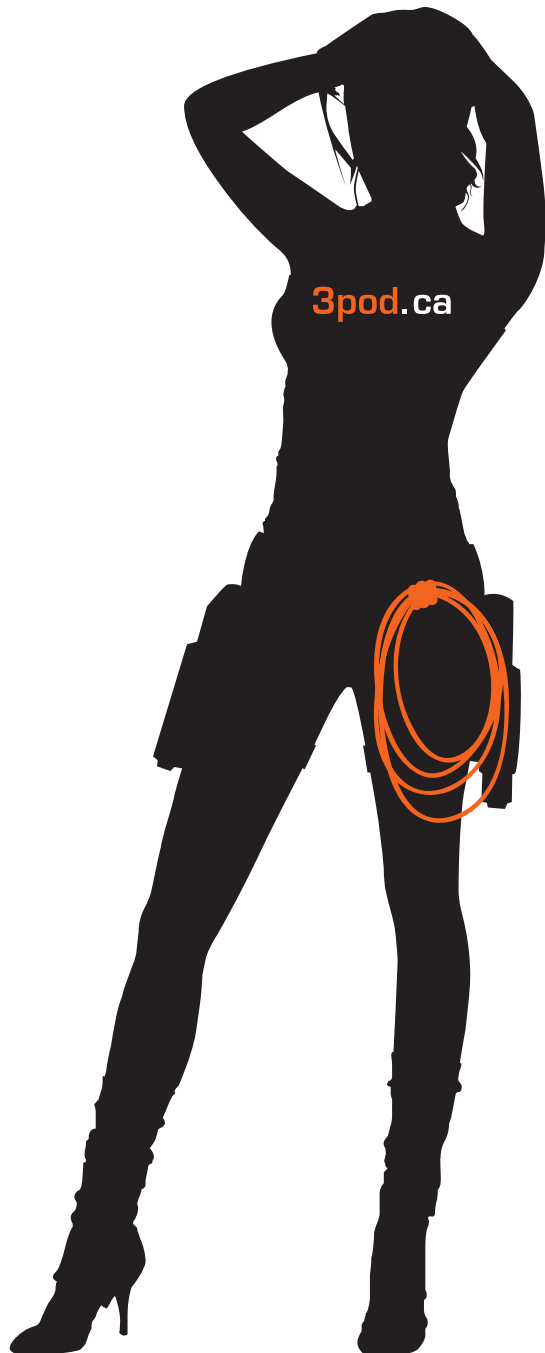
The trajectory calculations of moving particles will depend on their attributes, such as speed, velocity, and orientation, as well as outside forces, such as gravity, wind, and collision response. Further, the outside or global forces can modify a particle's attributes, despite its current position, speed, and orientation.

Here, I present two alternative methods for doing particle position calculations. The first method uses



pseudo-random generation, and for that we must force a particle to “choose” a direction toward the landscape location. In other words, we need to simulate a “gravity”





The premiere recruiting agency. **Committed to talent, dedicated to gaming.**

## LISTING 04

### PARTICLES DOWNFALL WITH SPHERICAL COORDINATES CALCULATIONS

```

/* rest of the code (if needed) */

// used variables:
// -----
// range - range for the particle displacement
// pCol - collection of the particles
// PARTICLES - number of particles in a collection

for (int i=0; i < PARTICLES; i++)
{
    // next three lines was inserted only for code readability
    double range pCol[i].Speed;
    //we used here the Speed particle property
    double Alpha pCol[i].Angle.Alpha;
    double Beta pCol[i].Angle.Beta;

    pCol[i].Position.X += range*Math.Sin(Alpha)*Math.Sin(Beta);
    pCol[i].Position.Y += range*Math.Cos(Beta);
    pCol[i].Position.Z += range*Math.Sin(Alpha)*Math.Cos(Beta);
}

/* rest of the code (if needed) */

```

## LISTING 05

### SIMPLE HEIGHTFIELD CLASS

```

class THeightfield
{
    // number of columns of the heightfield
    // with read-only property
    private int __Cols;
    public int Cols
    {
        get { return __Cols; }
    }

    // number of rows of the heightfield
    // with read-only property
    private int __Rows;
    public int Rows
    {
        get { return __Rows; }
    }

    // the heightfield value
    public double [] Cell;

    /// <summary> THE HEIGHTFIELD BASIC CONSTRUCTOR </summary>
    public THeightfield()
    {
        __Cols=0;
        __Rows=0;
        Cell = null;
    }

    /// <summary> THE HEIGHTFIELD MAIN CONSTRUCTOR </summary>
    public THeightfield(int setCols, int setRows)
    {
        // <param name="setCols"> number of columns of the heightfield </param>
        // <param name="setRows"> number of rows of the heightfield </param>
        __Cols = setCols;
        __Rows = setRows;

        // let's try to allocate memory
        Try
        {
            Cell=new double [ __Cols * __Rows];
        }

        // and catch an exception if allocation fail
        catch (OutOfMemoryException)
        {
            MessageBox.Show(
                "Not enough memory to store the Heightfield",
                "Allocation error",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error
            );
            Application.Exit();
        }
    }

    /* rest of the code (if needed) */
}

```

factor (see Listing 3) and simply set the most probable direction of the particle to decrease its altitude.

The second method is to calculate particle position based on speed and direction (see Listing 4). To do that, we can use simple equations for spherical coordinates. See Figure 1 for an example of that kind of equation.

### TERRAIN DATA STRUCTURE AND MODELING

Once we have defined and organized our particle system, we can proceed to build our virtual landscape surface. For the terrain data structure, we can use a typical heightfield organized as a standard two-dimensional array (see Listing 5). Each cell represents the height value of a three-dimensional point defined by rows and columns of this array. As you can see in Figure 2, this type of data structure can be easily converted to a polygons-net and drawn by any graphic engine.

To model the virtual terrain surface, we could use a variety of techniques and mathematical equations—but to make it fast, we need a simple one. We can use an equation from the circle algorithm and convert it to the particle method, as shown in Listing 6.

When a particle collides with the terrain surface, it sets the central point of its influence zone over a related heightfield. At this point, a height modification factor will be the strongest, and all height values in the vicinity of that point will be affected. The particle's effects will be weaker

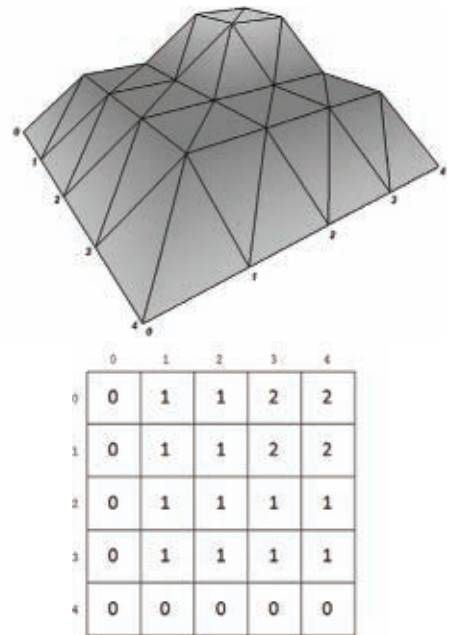


FIGURE 2 The data structure shown here has been converted to a polygons-net.



**KILLER  
TRACKS™**



**GDC ATTENDEES!**  
Visit Killer Tracks  
at GDC in booth  
5345 (North Hall)  
to receive a  
complimentary  
copy of our new  
Video Game  
Music Sampler  
and more!



**KILLER**  
TRACKS

# Video Game Music

**Perfect For:**

- Action
- Fantasy Adventure
- Shooter
- Strategy / Military
- Sports
- Music Games
- Trailers & Cinematics

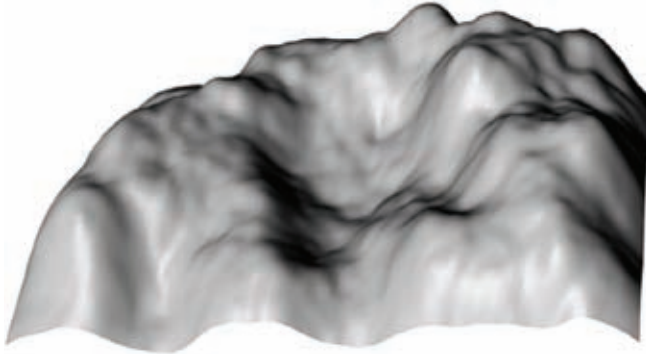
**ASK ABOUT OUR NEW GAME TRAILER CUES FROM AAA GAME  
COMPOSERS BILL BROWN, LENNIE MOORE & ALEX KHARLAMOV!**

**Web: [www.killertracks.com](http://www.killertracks.com) • Email: [sales@killertracks.com](mailto:sales@killertracks.com) • Phone: 1.800.4.KILLER**

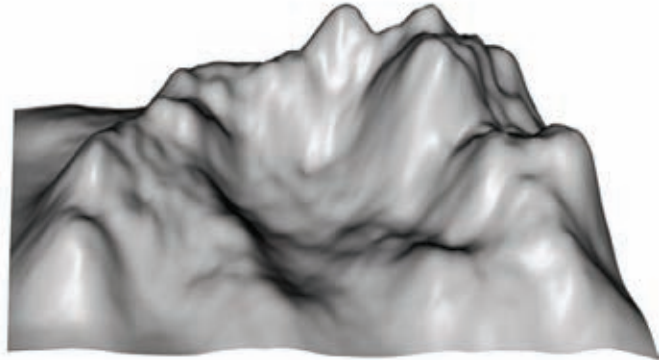


# GROUND FROM SMOKE

Particle size = 15

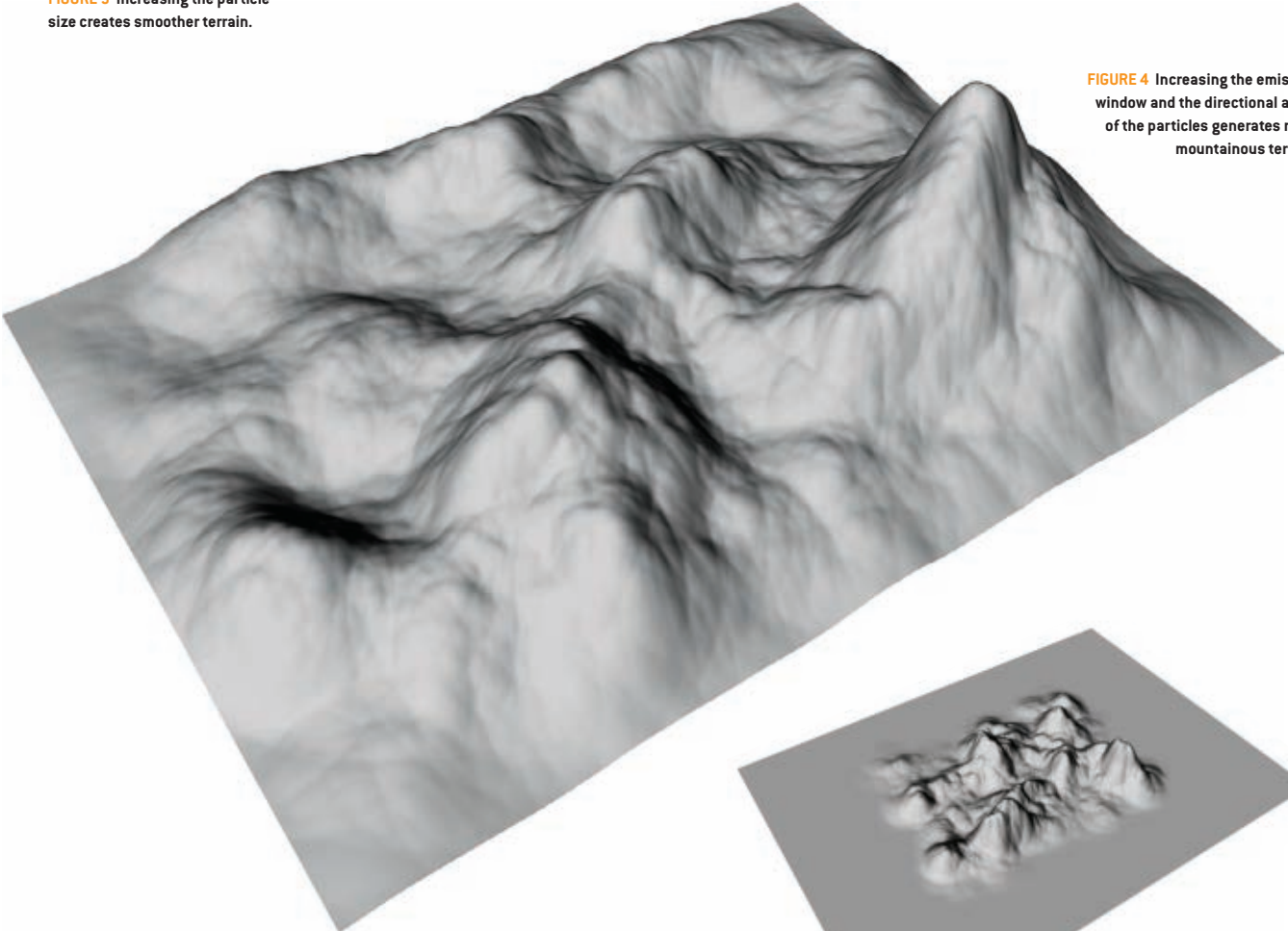


Particle size = 10

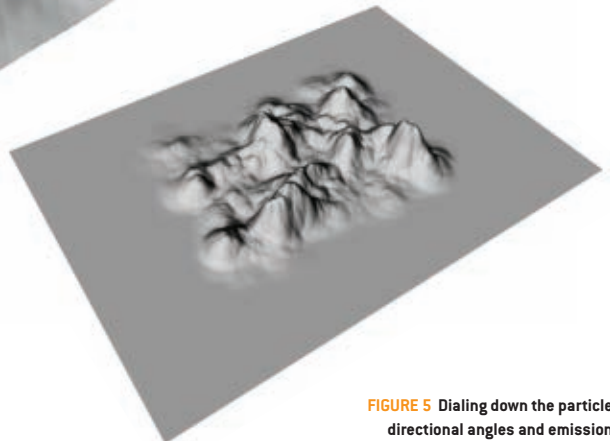


**FIGURE 3** Increasing the particle size creates smoother terrain.

**FIGURE 4** Increasing the emission window and the directional angle of the particles generates more mountainous terrain.



**FIGURE 5** Dialing down the particle directional angles and emission window produces islands.





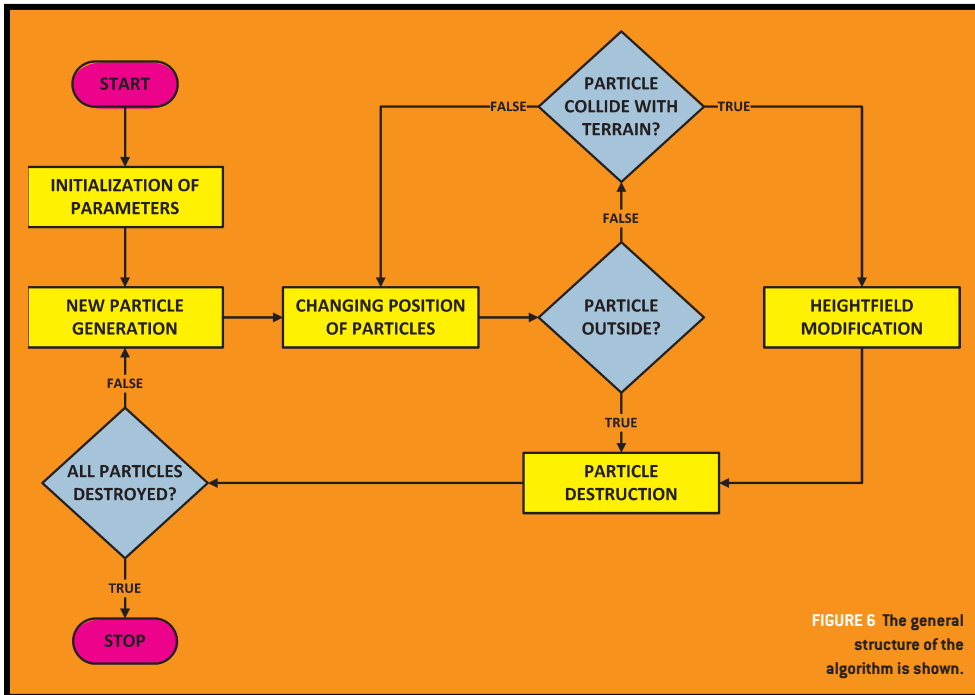


FIGURE 6 The general structure of the algorithm is shown.

## LISTING 0.6

### SURFACE MODIFICATIONS

```

/* rest of the code (if needed) */

// used variables:
// -----
// HF - heightfield data object
// CP - collision point coordinates
// (structure of three double
variables X, Y and Z)
// range - maximum distance of
influence zone
// from particle size
property
// hFactor - height modification
factor

for (int i=0; i < HF.Cols *
HF.Rows; i++)
{
    double hFactor=Math.Pow(range, 2)
    - (Math.Pow((i %
HF.Cols) - CP.X, 2)
+ Math.Pow((i /
HF.Cols) - CP.Y, 2));

    if (hFactor > 0)
    {
        HF.Cell[i] += hFactor;
    }
}

/* rest of the code (if needed) */

```

## RESOURCES

- >> Lander, J. "The Ocean Spray in Your Face," *Game Developer*, July 1998.
- >> Reeves, W.T. "Particle Systems: A Technique for Modelling a Class of Fuzzy Objects," *Computer Graphics*, Vol. 17, No 3, July 1983, pp 359–375.
- >> Van Der Burg, J. "Building an Advanced Particle System," *Game Developer*, March 2000.
- >> M.A. DeLoura, ed. *Game Programming Gems*, Vol. 1, Charles River Media, 2000.
- >> M.A. DeLoura, ed. *Game Programming Gems*, Vol. 2, Charles River Media, 2001.
- >> Dante Treglia, ed. *Game Programming Gems*, Vol. 3, Charles River Media, 2002.

### TERRAIN TUTORIAL

[www.lighthouse3d.com/opengl/terrain](http://www.lighthouse3d.com/opengl/terrain)

as distance increases. The range of the influence zone is a product of several particle properties, such as size, mass, and weight, but in this article it depends only on particle size. The influence of a particle on a heightfield diminishes when the modification factor of height approaches a value of zero.

### MODELING ISSUES AND PARAMETERIZATION

Using this terrain modeling technique, we can create several different types and styles of landscape surfaces. We can generate rougher terrain, for example, if we modify the particle size. Increasing it will make the terrain smoother, as shown in Figure 3.

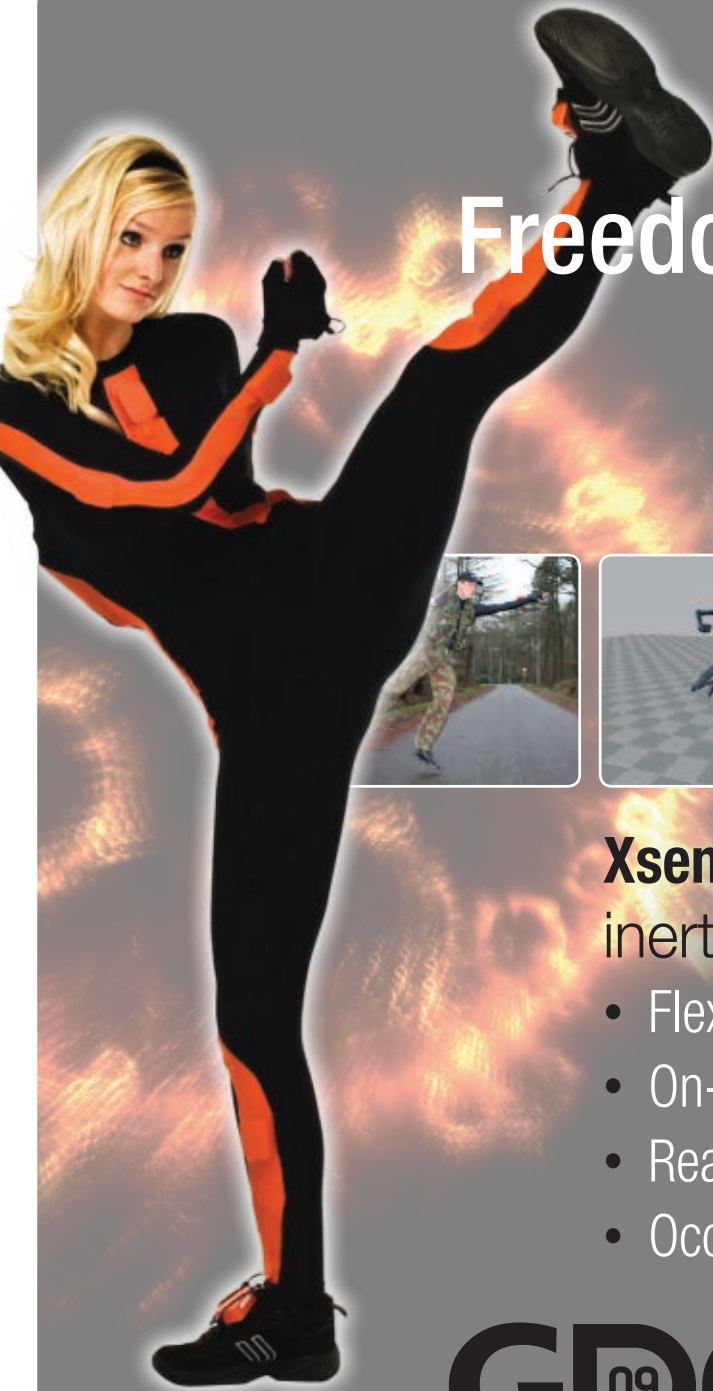
Using a wider emitter window and setting the directional angle for the particles to be broader, we can create a complex heightfield, which resembles hills or mountains (see Figure 4). Alternatively, by making the emission window narrower and tightening the particle directional angles, we can generate islands. In Figure 5, which shows an island that was created using this method, only a limited area of the

heightfield is exposed to particle influences.

Finally, just before rendering, there must be normalization of the height values on the whole heightfield. Typically, the normalization range is [0.0–1.0]. For calculations of normalized values we can use a common normalization equation as follows:  $y_n = (y - \min) / (\max - \min)$  where,  $y_n$  is the new normalized height value,  $y$  is initial unnormalized height value, and  $\min$  and  $\max$  represent the minimum and maximum height value found on the heightfield.

The basic idea of the algorithm can be summarized in just a few steps; its general functionality is shown in Figure 6.

The beauty of using particle systems to generate surface terrain is that it gives developers a fully-automated way to model a wide spectrum of virtual landscapes. It can be used either for forming complex heightfield or limited areas with different terrain surfaces. It is also fairly sensitive to parameterization, which is useful for the ways in which game developers adapt the technique for modeling their virtual worlds. ❖



# Freedom of movement



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

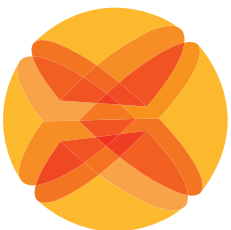
- Flexibility and short turnaround times
- On-set/outside motion capture
- Real-time visualisation
- Occlusion-free

**GDC** 09

Meet Xsens at:

GDC 09, March 25-27, Booth 5934,  
Moscone Center, San Francisco, USA

CUSTOMERS & PARTNERS INCLUDE



**xsens**

gearbox  
software

THQ

SONY

BIG  
HUGE  
GAMES







» david sirlin

# SUBTRACTIVE DESIGN

“Make everything as simple as possible, but not simpler.”—Albert Einstein

**S**ubtractive design is the process of removing imperfections and extraneous parts in order to strengthen the core elements. You can think of a design as something you build up, construct and let grow, but it's pruning away the excess that gives a design a sense of simplicity, elegance, and power.

## WHY SUBTRACTION?

Designers in many fields, not just games, often think in terms of negatives [subtracting things] rather than positives [adding things]. Design is creating a form (a game in our case) that fits a context. There isn't just one boundary we have to check between form and context though, there are infinitely many. Is our game easy enough to learn? Does it have the desired amount of strategy or depth? Does it appeal to the intended age-group? Is it cheap enough to make in both time and money? Is it aesthetically pleasing? Do the aesthetics help the player understand how to play the game?

Do the mechanics work well with each other? Do they require the desired amount of dexterity? The list goes on.

We first come up with a design that might fit all the requirements. Sometimes this comes from the intuition of a designer who has internalized all those forces and somehow spits out a new answer. More likely, we start with something pretty well established so that we know it solves many of the requirements already. That's how genres, sequels, and remakes help us make good (but not necessarily new) designs.

Once we have something, we have to evaluate how good our design is. Does our form actually fit the context? Architect Christopher Alexander had some choice words on this subject in his *Notes on the Synthesis of Form* (Harvard University Press, 1964):

“We should find it almost impossible to characterize a house which fits its context. Yet it is the easiest thing in the world to name the specific kinds of misfit which



prevent good fit. A kitchen which is hard to clean, no place to park my car, the child playing where it can be run down by someone else's car, rainwater coming in, overcrowding and lack of privacy, the eye-level grill which spits hot fat right into my eye, the gold plastic doorknob which deceives my expectations, and the front door I cannot find, are all misfits between the house and the lives and habits it's meant to fit. These misfits are the forces which must shape it, and there is no mistaking them. Because they are expressed in negative form they are specific, and tangible enough to talk about."

Alexander explains that when a misfit occurs, we are able to point at it specifically and describe it. When we instead try to explain what a good fit would be like, we're often reduced to generalities that are hard to act on.

"With this in mind," says Alexander, "I should like to recommend that we should always expect to see the process of achieving good fit between two entities as a negative process of neutralizing the incongruities, or irritants, or forces, which cause misfit."

## APPLYING SUBTRACTIVE DESIGN TO GAMES

There's no question that it's easiest to get people to agree on vague concepts, and high-level design descriptions sometimes sound like this: "The game we're making is a platformer with exploration, but also with fast action and time pressure. It has an epic story of course, but also a personal story. The game is really challenging, but it's for everyone to enjoy. It has 20 enemy types and 20 weapons including a kazoo and a kitchen sink." What's not to agree with? There's something for everyone.

A more direct approach would be to find a starting point, whether it's another game's design or something you generate yourself. How to do that is another topic entirely. But once that core concept is in place, stay on the lookout for things that get in the way. Is every button press really needed? Every menu item? Every HUD element? Are there features that you just don't have time to do justice? (Let's add multiplayer!) If the game is about testing player skill, is it testing the only the skills you want to test? If it's about story, are all the scenes contributing to that story? Do you need



all the mechanics you planned, or will a smaller set (easier for the player to remember and learn) suffice? Is there a way to make your levels smaller or shorter by removing or shrinking areas that do not have much purpose?

Getting rid of all that stuff means there are fewer things for the player to misunderstand, and makes it more likely that the vision of the game in your head actually ends up in the player's head, too.

## CASE STUDIES

**A discussion of successful subtractive designs should help illuminate why and how subtractive design is done, and also why this practice can yield greater critical success. As examples, I'll discuss ICO, BRAID, PORTAL, and TEAM FORTRESS 2, as well as the non-game designs of Google Chrome and Apple Time Machine.**

### /// ICO

In a lecture at GDC 2004, Sony's Fumito Ueda shared his design process for *Ico*. He did not start with a list of everything the game should have—instead, he started with the core idea that it should be a platform / puzzle game about a boy and a girl, and that the game should have emotional impact by creating an environment that had its own believable reality to it. Using other platform and puzzle games a point of reference, he then started subtracting away everything that was extraneous to his core idea.

Other games might use a traditional hero's journey-inspired nine act structure,

where the story starts in a village, then you go into the forest, then you find a castle, then escape back to the forest, and so on. Ueda was conscious of this, but cut everything except the castle, so that the castle could be fully realized, fully polished, and seem to be a character of its own. Other games have NPCs that stand around and give you hints, but when you see the same character say the same lines over and over, it takes you out of the fictional world. Ueda stated from the beginning that he would have no such NPCs. Other games may have an army of different enemies, but Ueda found that puzzles were enough, and only one type of enemy was needed. He also removed the health meter, inventory screen, and even background music from his design—all things that come standard in other games.

Another misfit that was on Ueda's mind was bad animation. If the core idea is to show a boy and a girl escaping a castle together, and we want a sense of immersion and reality, then nothing in the boy's or girl's animations can stand out as strange. His team spent a great deal of effort on the character animations, especially those where the girl and boy interact, because any imperfections there would have been glaring.

I learned all this from Ueda's 2004 presentation, but there was one more detail that stuck with me. He showed us a screenshot of one room in the castle and asked us what was wrong with it. I thought it looked pretty good. Ueda then pointed out that there was a chair in the screenshot that didn't look very good. He said when something like this happens,



IF YOU COULD CREATE A

# VIRTUAL WORLD

FOR YOUR BUSINESS

# GEORGIA

IS WHAT IT WOULD BE LIKE.

More than 60 video game companies—from Time Warner's GameTap and Atlanta's own Kaneva, Inc., to China's CDC Games and CCP North America, creators of EVE Online—have found Georgia is designed to help video game developers succeed. Our state's deep talent pool is fed by cutting-edge schools like the Savannah College of Art and Design, Georgia Tech, and Georgia State University. Financial support is obtainable from private investment capital and tax incentives provided by Georgia's Entertainment Industry Investment Act. Plus, Atlanta boasts the planet's premier airport and is the most heavily wired city in the U.S. with the fat pipes you need. Georgia is a world unlike any other, and we're ready to help you grow your game business. Contact the Georgia Film, Music & Digital Entertainment Office today.



Georgia®

Visit [georgia.org/gamedevelopment](http://georgia.org/gamedevelopment) or call 404.962.4052.

you have to decide whether you have the time and resources to fix it (make the chair more beautiful and believable) or cut it. In this case, he cut the chair.

For all ICD's cuts, you'd think it would be a game with nothing much left. And yet it received high critical acclaim as powerful game. The important elements are executed unusually well, and the unimportant elements—I couldn't find any.

## /// BRAID

Jonathan Blow's BRAID shows a similar reductionist design with a similarly powerful result. The core concept behind BRAID is the manipulation and rewinding of time without the need of a meter to limit the mechanic. You can rewind time as much as you like, as often as you like, and the difficulty of the game comes in puzzles that test just how clever you are with this manipulation. Because time manipulation is the core concept, BRAID explores time mechanics fully. On one level walking left reverses time while walking right moves time forward. On another, some objects are immune to the time shifting. Each level investigates a new idea.

What's remarkable about BRAID is how many things Jonathan Blow cut away. There are no "lives," because the entire



idea of lives is incompatible with having infinite time-rewind powers. There are only about five types of enemies, much fewer than is usual for a 2D platform game. There are only two action buttons: jump and rewind time (though there is a third button used later in the game to put down an object).

What's most strikingly minimal of all in BRAID is the level design. Each level is as small as it can possibly be, containing as few elements as it could realistically contain while still being interesting. This gives the game's construction the feel of a tight short story: every part is there for a reason and there's nothing extra. Even the quantity of levels follows this logic—when the game is done exploring new time mechanics, it ends. It feels no need to make us fight hundreds of blue slimes

in order to level up to fight red slimes.

By trimming the fat in action buttons, UI, enemy types, level size, and level quantity, BRAID feels vigorous in the way Strunk and White meant in *The Elements of Style, 4th Edition (Longman, 1999)*, as seen in his discussion of omitting needless words.

"Vigorous writing is concise. A sentence should contain no unnecessary words, a paragraph no unnecessary sentences, for the same reason that a drawing should have no unnecessary lines and a machine no unnecessary parts. This requires not that the writer make all his sentences short, or that he avoid all detail and treat his subjects only in outline, but that every word tell."

## /// PORTAL

Valve's game PORTAL is another example of a compact, distilled design. The core idea is that you can shoot two kinds of portals (orange and blue) and then walk through one to come out of the other. You have no actual weapons, no inventory screen, no NPCs to talk to, not even any enemies aside from the occasional turret and the final boss. The controls are as simple as they can be, with action buttons only for shooting the two kinds of portals, for jump, and for using objects (open door, pick up crate, and so on).

PORTAL's environments are sparse and sterile, containing practically nothing except for elements that are part of puzzles, elements that offer you visual cues as hints about what you should do, and elements to convey the story. That there's nothing extra puts all the emphasis on the portal mechanic itself, which is incredibly fun. Like BRAID, PORTAL explores its mechanic fully, doing just about everything you can think of to do with portals, then it gracefully ends without overstaying its welcome or subjecting you to filler content.

## /// TEAM FORTRESS 2

Valve's TEAM FORTRESS 2 has a lot of things going for it, but it's specifically the approach to map design that stands out as a case for subtractive design. Most games of this type would offer as many maps as possible. More is seen as better by marketing departments, after all. Valve deliberately limited the game to only six maps when it shipped, though.

One benefit of fewer maps in a multiplayer game is less fragmentation of

the player-base. If there are hundreds of maps it can be hard to find anyone who wants to play the particular map you do. But more to the point, Valve knew that in most multiplayer games, the community settles on just a very few maps they play endlessly. If this is a known phenomenon in so many games, why make tons of maps?

By sticking to only six, an unusually small set for this type of game, Valve had time to make these the best, most polished maps they could. Fewer maps means each one received more attention from playtesters, artists, and designers. The process of playtesting a map is, itself, a subtractive design exercise. You play it as much as possible in as many different ways as possible, looking for bugs, exploits, and defects that make the gameplay less fun or less strategic. The more you limit the number of maps, the more defects you can fix in each one.

## /// GOOGLE CHROME

Google has always had a simple elegance in its products, and the Google Chrome web browser is a great example of subtracting the debris that other browsers have. Why do we need two different fields at the top of a browser (one to search the web, one for the URLs) when they could be combined into one? Google Chrome does this to save space and reduce clutter. There is no chance of confusing the two uses in one field anyway, because search terms have spaces between them while URLs have things like ".com" in them.

The core idea behind Google Chrome is "get out of the way and remove debris whenever possible." The "find" field only shows up when you press cmd-F; otherwise it's not even there to get in the way. When you mouse over a link, the status bar showing where the link points to shows up in small box in the corner, but this box fades away entirely at other times. Chrome also got rid of these things from Internet Explorer 6:

File menu, edit menu, view menu, favorites menu, tools menu, help menu, homepage button (you can turn it on in the options), search button (just type in the URL bar to search), favorites tab button, history button, mail button, print button, edit button, messenger button, the word "address" labeling the address bar, and the status bar.

It added these interface elements: menu button for options about the current page,



# YOUR ULTIMATE RESOURCE

Course Technology PTR is your ultimate game development resource. Our books provide comprehensive coverage of everything from programming, story development, character design, special effects creation, and more, for everyone from beginners to professionals. Written by industry professionals, with the techniques, tips, and tricks you need to take your games from concept to completion.

Introducing the *GameDev.net Collection* series! Each book features a collection of the best articles taken from the GameDev.net archives, updated and revised for the current technology, as well as brand new articles never before published. GameDev.net is the leading online community for game developers to network and share ideas. Start your collection today!



**Business and Production for Games**  
A GameDev.net Collection  
1-59863-809-2 • \$29.99



**Design and Content Creation**  
A GameDev.net Collection  
1-59863-808-4 • \$49.99



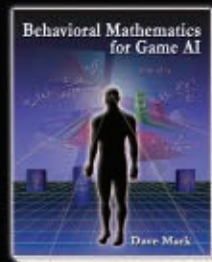
**Beginning Game Programming**  
A GameDev.net Collection  
1-59863-805-X • \$39.99



**Advanced Game Programming**  
A GameDev.net Collection  
1-59863-806-8 • \$44.99



**ShaderX7**  
Advanced Rendering Techniques  
1-58450-598-2 • \$59.99



**Behavioral Mathematics for Game AI**  
1-58450-684-9 • \$49.99



**Protecting Games**  
A Security Handbook for Game Developers and Publishers  
1-58450-670-9 • \$39.99

**Game Coding Complete**  
Third Edition  
1-58450-680-6 • \$59.99

**Beyond Game Design**  
Nine Steps Toward Creating Better Videogames  
1-58450-671-7 • \$39.99

**Figures, Characters, and Avatars: The Official Guide to Using DAZ Studio to Create Beautiful Art**  
1-59863-816-5 • \$34.99

**David Perry on Game Design**  
A Brainstorming Toolbox  
1-58450-668-7 • \$49.99

Visit us at **GDC 2009** in **Booth #5545!** Stop by to check out these new releases and save with special show discounts on our comprehensive list of game development titles.

Not coming to GDC? Our complete line of books are available at Amazon, Barnes & Noble, Borders, and other fine retailers nationwide. Visit us online at [www.courseptr.com](http://www.courseptr.com) or call **1.800.354.9706**.

menu button for options about Google Chrome in general, and a new tab button.

Does this simpler interface with all the debris removed mean that Google Chrome is less powerful or less advanced than its competitors? Quite the contrary. Under the hood, it separates each tab into a new process, meaning that one pesky website can't cause your entire browser to crash. It also does a better job of preventing memory leaks than its competitors with better memory garbage collection. The power is out-of-sight, and the browser UI is as minimal as possible so that everything is there for a purpose, everything with no purpose is gone, and the things that are there are high quality.

### /// APPLE'S TIME MACHINE

Apple also has a long history of simple and elegant products. While most would probably point to the iPod's scroll wheel as the classic example of an elegant interface, I'll use the less familiar example of Apple's Time Machine software. The core concept behind this built-in part of a Mac's operating system is "make it so easy to back up your data that you'll actually do it." The problem with data backup software is not that it doesn't do enough things, it's that the average user is too lazy to ever actually do it at all.

Apple's solution is a "zero click" interface, though maybe it's more fair to call it one click. When you plug in any drive, a

daily backups for the last month, and weekly backups forever, until the drive full.

The interface for recovering old files is slick and useful, but it's the "zero click" setup that makes the feature so practical.

### COMMON THEMES

There's something all these examples have in common. Apple's Time Machine and Google Chrome are both very sophisticated under the hood, even though they present simple interfaces to the user. TEAM FORTRESS 2 is not more shallow for its decision to launch with fewer maps; it's actually deeper because of that decision. PORTAL and BRAID did not, as Professor Strunk would say, "avoid all detail and treat their subjects only in outline." Quite the contrary. PORTAL and BRAID each fully explore their concepts—more fully than most larger games can explore theirs. And finally, ICD's sense of reality, immersion, and emotional power is not less because it subtracted all the extraneous elements; it is greater. In each case, subtracting did not leave us lacking, it enhanced the experience.

### THE CONTROVERSY

Subtractive design is not all rainbows and puppies though. By fully committing to this idea, you are more likely to encounter resistance on your game development team, with your publisher, and with your players. The reason is that when we use vague language, it's easier to get an agreement. When we use very honest, precise language, it's easier for someone to realize that they disagreed all along.

Consider this: "Some amount of collateral damage is expected in the mission." Sure, ok.

Then this: "We are going to kill innocent people on this mission." Wait, really?

When we distill a design down to the core concepts and

remove the extraneous, it forces us to admit and agree what the core concepts actually are. For example, as designer of STREET FIGHTER HD REMIX, I made the statement that performing difficult moves is not part of the core concept of the game. It's an imperfection that should be removed, so that there can be more focus

on the essence of the game: strategy. Clearly, that is troublesome statement if you believe that performing difficult moves is part of the essence of the game. I think subtracting some emphasis on that aspect enhanced the final product though.

Likewise, in STARCRAFT, the ability to play not just fast, but extremely fast is highly rewarded. When Blizzard has to decide what STARCRAFT 2 is really about at its core, it might decide that a game in the "real-time strategy" genre should focus more on strategy and less on extremely fast clicking, and design its UI to have more user-friendly features. Regardless of how STARCRAFT 2 ends up, you can see that the mere proposal to focus on the essence of the game raises deep questions about what the game is really about. Is STARCRAFT really about strategy? Or is it equally about rewarding the most actions per minute that you can enter? It's easier to agree that we like STARCRAFT overall (vague) than it is to agree on whether a new version of the game should or shouldn't remove the emphasis on certain skills.

The card game *Magic: the Gathering* has an even deeper conflict about what it's really about. When it comes to the wording on the cards, the *Magic* team has made great strides over the years to remove unnecessary words, creating as many simple, elegant cards as they can. Compare the original wording of the card Control Magic to the current wording (see Figure 1).

But on a more zoomed out level, what is *Magic* really about? Is it about delivering the most fun gameplay experience possible to its players? Or selling collectable items that have artificial scarcity? One gets in the way of the other, as it stands. I propose that the essence of customizable card games is the gameplay, and that collectability is purely a barrier between players and the game. But making such a statement naturally creates a firestorm of argument because it forces us define what the essence of a game is. That can be uncomfortable to do.

### SUBTRACT TO ADD

It takes some courage and pain to commit to a specific idea and subtract the rest away, but I think ICD, BRAID, PORTAL, TEAM FORTRESS 2, Google Chrome, and Apple Time Machine all demonstrate that doing so can lead to powerful, memorable experiences. ✨



FIGURE 1 The Control Magic card from *Magic: the Gathering*'s original edition and current edition.

pop-up asks you if you'd like this to be your Time Machine drive to back up your files (it doesn't ask if you've already set a Time Machine drive, of course). If you say yes, that's all there is for you to do. Time Machine will then back up all your computer's files and keep running backups every hour for the last 24 hours,





STOP THE ZOMBIES  
GIVE 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)

 **xaitMENT**  
INTELLIGENCE ENGINEERED



# POSTMORTEM







# UBISOFT MONTREAL'S FAR CRY 2

////// Far Cry 2 was an enormously challenging project from both a technical and creative perspective. For the team at Ubisoft Montreal, it was our vision from the beginning to deliver a seamless 50-square-kilometer open world, with no loading, that was as beautiful from both technical and artistic standpoints as any modern, top-tier, corridor shooter. We also envisioned for the game a highly dynamic and destructible environment that supported robust and realistic fire propagation through building structures and vegetation.

## OPEN WORLD AMBITION

Once in production, we retargeted from a PC-only release to a planned simultaneous launch on both PlayStation 3 and Xbox 360. And each version would support multiplayer versions and contain a level editor.

As if that weren't enough, we also aimed to innovate on numerous fronts by making the main characters of the game autonomous and unscripted, allowing the player to build relationships with them through game mechanics. We further expected these characters to be able to live or die at any time as determined by the player's actions, effectively creating a dynamic narrative that could sustain itself

---

**CLINT HOCKING** has been at Ubisoft for almost eight years where he has worked as a level designer, game designer, scriptwriter and creative director on the original *SPLINTER CELL*, on *SPLINTER CELL: CHAOS THEORY*, and on *FAR CRY 2*. He lives happily in Montreal with his wife and their dog. Email him at [chocking@gdmag.com](mailto:chocking@gdmag.com).





beyond the attrition of the major characters of the story.

As is typical, we succeeded at delivering some of these things, and failed at delivering others.

## WHAT WENT RIGHT

**1) TOOLS.** From the very beginning, technical director Dominic Guay asserted, “We have designed a game that forces us to make tools that will allow artists and designers to build and iterate content very rapidly.”

With engine development planned to happen parallel to game development, we would be working under constantly shifting budgets, and we would need to be constantly tweaking and tuning the gameplay, even as the technical constraints changed. We literally needed to be able to build a square kilometer of the game in one day, and then be prepared to throw it away the next day if things changed.

At the end of pre-production, we presented a two-minute time-lapse video of a level designer and a level artist creating a one-square-kilometer section of African jungle in four hours. The proof-of-concept included all the terrain, dynamic vegetation, roads, structures, AI, and gameplay, and demonstrated beyond any doubt that we could create our game world.

The toolset would ultimately become the foundation of the level editor that shipped with all versions of the game. It allows players to create multiplayer maps rapidly and iterate their designs

to deliver professional-quality levels to the FAR CRY 2 multiplayer community.

**2) COMMITMENT TO INNOVATION.** It is a misconception that innovation is overthrown by corporate coup and that large companies are averse to innovation. Innovation is more typically chipped away at, piece by piece, and frequently, it is dismantled from the bottom-up. As experiments in delivering new experiences fail and time ticks by, individuals tend to retreat to more conventional or established solutions, which not only wither the blossoms of innovation, but can kill it at the root.

On FAR CRY 2, we often committed to no-compromise strategies that would not accommodate partial withdrawal. For example, the dynamic narrative structure of the game does not function with fewer than 12 buddy characters. When the game flow and scripting of the narrative were massively behind schedule, and the character modeling team was badly overloaded at the same time, removal of buddies began to look very attractive.

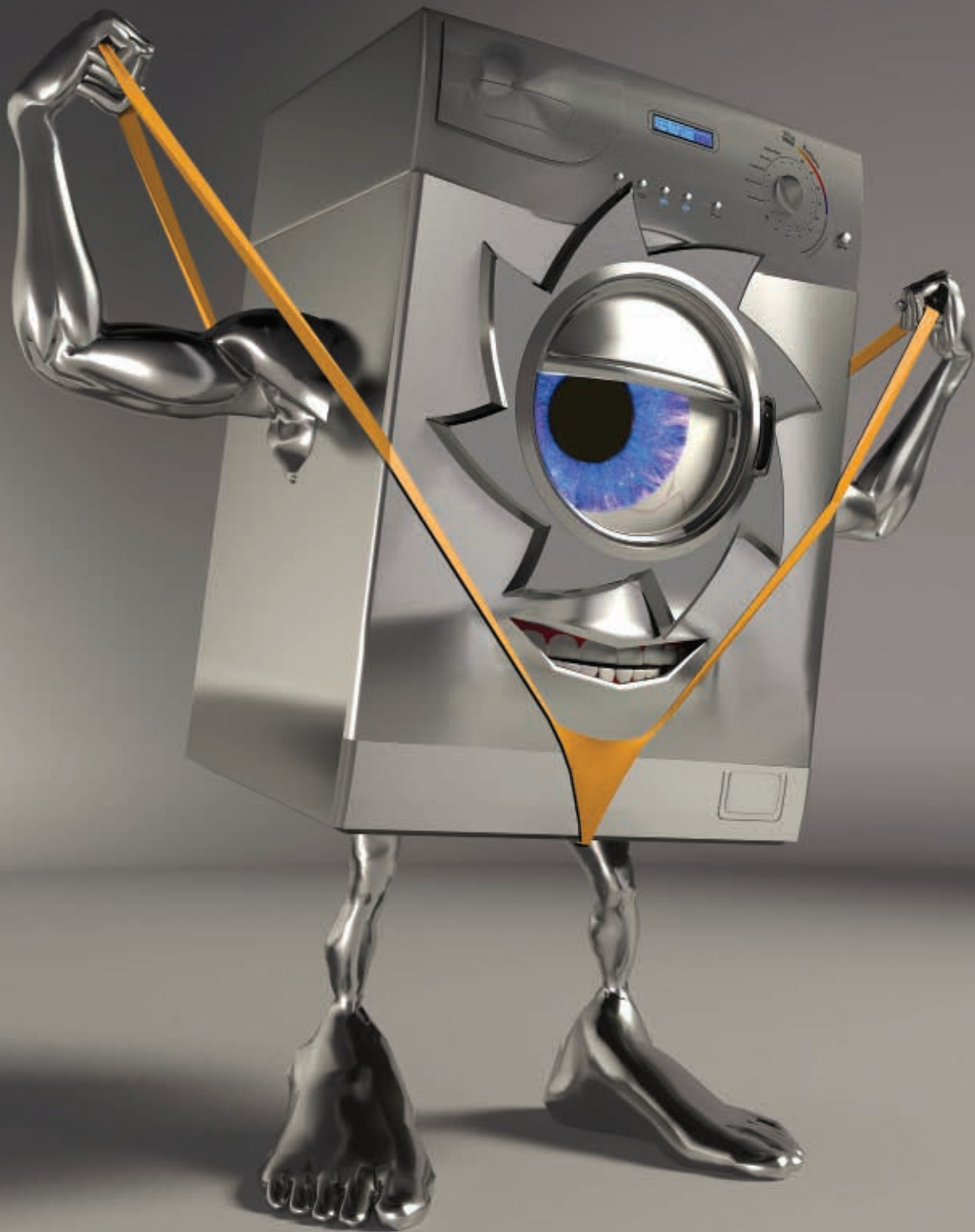
But there was no Plan B. There was no way to have a “partially dynamic narrative.” There was no way to throw money at the problem and render cut scenes at the last minute to tell the story, because in a very real sense, we had not written a “story”—we had written the elements required to support a dynamic narrative that worked with exactly 12 buddies.



CONTINUED ON PG 34



Dynamic behavior. Clothing required.  
Unparalleled support.



Havok Behavior  
Havok Cloth





CONTINUED FROM PG 32

Maybe this sounds like something that belongs in the “What Went Wrong” section, but in fact, it was an all-or-nothing commitment to the important elements of the dynamic narrative that made it possible at all. There remain many peripheral failures in the delivery of the narrative design, but the fact that we were committed to delivering it at all is what allowed us to make those failures, and consequently to learn from them so that we can iterate and improve them in the future.

**3) EMPOWERING CREATIVITY.** As creative director, my job is not to create the game, but to get the most creativity out of an entire team by empowering people to work in a way that allows their creativity to be expressed in their work. *FAR CRY 2*'s team reached that state more successfully than any other I've worked with.

This creative empowerment took a number of forms over the course of the project. In the concept phase of the project, I worked with a small team to harvest and catalogue all our ideas about what *FAR CRY 2* could be, and then sorted through them to aggregate the concepts that seemed to work best. After that, I briefed every new member of the team to ensure they understood the game concept and had someone to talk to about any creative concerns.

In production, I began the long process of turning over creative responsibility to the implementers. By ensuring that designers delivered all documentation on time, and that they then worked closely with implementers, we were able to slowly abandon the conceptual vision as it lived in documentation and in our heads for the reality of what was in the game and in the code. As this transition happened, individuals were encouraged and given the confidence to take creative ownership of their work, whether it was a level, an animation, or a piece of code.

If, toward the end of a project, a creative director is still explaining to people how and why to do something, he has already failed. Under the best circumstances, a creatively invested implementer—not a designer—is the person most qualified to make the decision about how best to deliver on the vision.

**4) A TRUE NEXT-GENERATION ART PIPELINE.** Before we were even finished with our concept phase, it was clear to art director Alex Amancio that a new art pipeline would be needed to deliver on the promises we had made.

Broadly, Amancio envisioned a technically complex art pipeline based on repetition and combination of memory-light compound assets. These assets would be made to appear more complex using run-time processes such as multi-layered shader systems, kit-system assemblers, dynamic weather systems and destructibility. Without the processes we would have required more numerous, heavier assets in order to avoid obvious repetition. Conversely, without light compound assets that reused all these components in dozens or even thousands of permutations, no amount of dynamic process would create the illusion of variety.

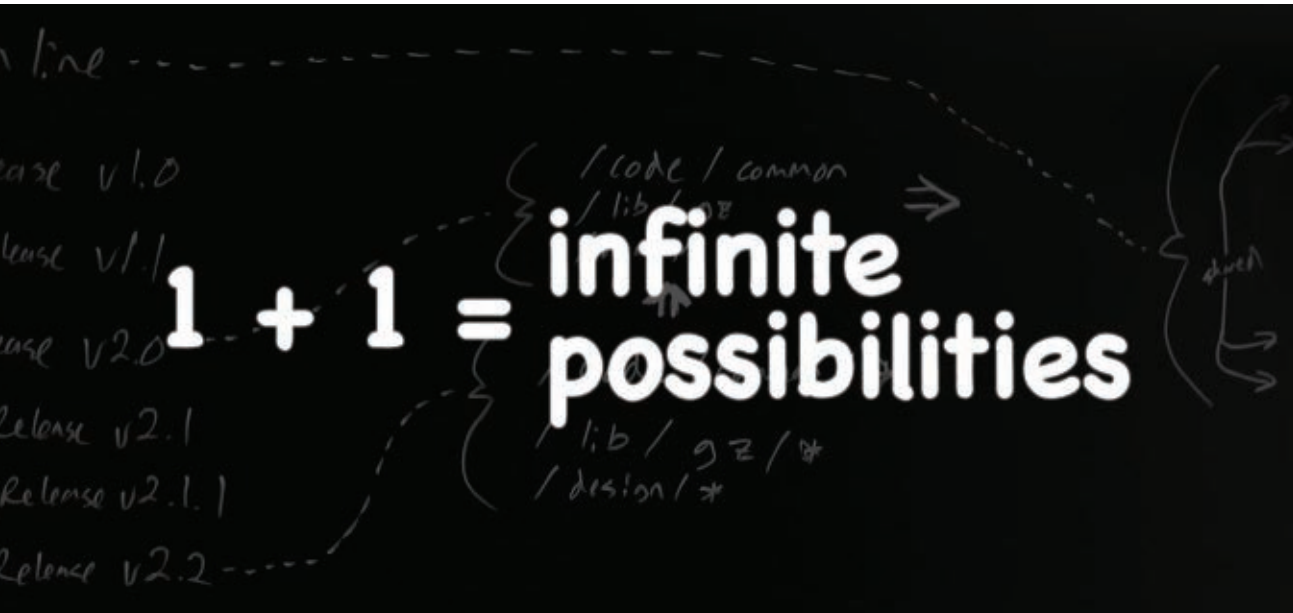
This philosophy was expressed everywhere; from heavily instanced vegetation assets that did not appear instanced at all under a procedural wind, to a “character kit system” that assembled AI characters upon instantiation from a collection of body parts, clothing, and equipment pieces that were all designed and built using the same mapping and reusing the same shaders. Instead of having 20 character models, we had





[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® ICM**  
Test Case Management



**TestTrack® Pro**  
Issue Management





20 character models worth of kit, providing nearly infinite variety at a cost far less than even 10 unique models.

No doubt, there were numerous hurdles to clear, not the least of which were the support demands on the graphics coding team and the training demands on artists joining the team, but in the end it was the commitment to this innovative pipeline—combined with traditional optimization techniques—that allowed us to fit in memory while constantly streaming from the hard drive and never having to dump the player to a load screen during gameplay.

**5) STAFFING UP.** In terms of staffing, FAR CRY 2 was in a difficult position from the very beginning because its development timeline was parallel to SPLINTER CELL: CONVICTION's, and one year behind ASSASSIN'S CREED. Despite the challenges, the FAR CRY 2 leads did an amazing job of finding dedicated and talented individuals who made tremendous contributions to the game.

Technical director Dominic Guay had planned a long and steady ramp-up of the engine team and game programming team, and managed to ease programmers into the code base at a steady rate of one or two per week instead of dropping 20 new coders into a code base maintained by only a dozen. This not only got them working efficiently more quickly, but also let us release stable daily builds.

Lead level designer Jonathan Morin and lead animator Sandra Warren were both uncompromising in their efforts to find the best possible candidates for their teams. Many new hires were experienced people from other projects, such as RAINBOW SIX: VEGAS, but others were graduates of the Ubisoft Campus, young developers directly out of school who had never shipped a game. Many of the least experienced team members outperformed our expectations and made important contributions to the final game.

I personally believe that it is neither the leads nor the veteran developers who make a game great—they are simply the ones who deliver it. The ones who make a game great are the inexperienced ones who exceed your expectations, create breathing room in your planning, and bring passion, energy, and fresh perspective to the team. Despite our staffing challenges, FAR CRY 2 had more than its share of talented and driven young developers.

## WHAT WENT RIGHT

**1) INABILITY TO ITERATE THE GAME FLOW.** Early in development, the design team made the case to the programming team that we would need to manage the dynamic narrative and game flow from a drama management engine written by a programmer. In balance against all the other engine features, this was deemed to be a job that could be handled by higher-level scripting, and that was how we proceeded.

By the middle of production, one scripter became two scripters, then three, and then a whole team. More and more scripting work was offloaded to the level design team. A programmer was required to code the core of the mission manager, and several programmers were needed throughout to provide scripting tool support. Even with the continual influx of resources and support, the scripting team was still badly overloaded and in perpetual crunch.

We were unable to reproduce a successful walkthrough of the entire game until about one week before the first master candidate. Consequently, virtually none of the drama management is tested, and even many of the things we did manage to test and prioritize for fixing were things we simply could not fix on time.

In the shipped game, drama management is driven by discreet player choices as filtered by reputation, an invisible counter that runs under the hood. We realized (too late to act) that a superior design would have driven the dynamic narrative based on a relationship history matrix involving the player and all the other





characters. But even though we had simple designs for a system that could have effectively patched the existing system to incrementally improve it, we were handcuffed by the weight of all our previous mistakes and the impossibility of testing even minor changes to the game flow.

**2) MULTIPLE REDUNDANT OVERLAPPING SYSTEMS FOR AI BEHAVIOR AND ANIMATION.** Early in development, we prioritized having a functional combat AI in place early so we would be able to iterate gameplay. While this had the benefit of guaranteeing we could have a robust and autonomous combat AI that could read the dynamic environment and deal with fire, vehicles, grenades, and whatever the player could throw at it, this approach prevented us from building a general purpose AI that could do all the things we wanted the AI to do outside of combat.

As an example, the smart terrain system that allowed AI characters in their default, non-combat state to wander around and perform actions based on their needs was not compatible with the briefing AI. This meant the autonomous behaviors used by non-combat AI could not be used in the briefings, and all the sequences with the warlords and their aides—the leaders of each faction who would brief the player on his missions—had to be scripted by hand [in all their permutations].

Worse still, the similar briefings with the buddies presented too many permutations for us to script them all, and therefore the buddies literally could do nothing but stand unmoving like statues and talk to the player with simple look-at blends. The simple behaviors of them doing stretches or checking inventory or using cell phones while waiting were hacked together in desperation at the last minute, instead of being designed and built from the beginning as part of a general purpose system that could be reused in any situation.

In the end, this problem massively constrained our ability to make the buddies, warlords, captains, and lieutenants as engaging and dynamic and credible as we wanted them to be.

**3) MEANINGFUL CHARACTERS ARE REALIZATION-DEPENDANT.** FAR CRY 2 is fairly unusual for a shooter in that the player gets to play the high-level gameplay: the dynamic story and the relationships with the characters. The decisions he makes when assassinating a major character, being rescued by a buddy, or dealing with a buddy who has fallen wounded on the battlefield, have lasting repercussions, the effects of which might not be relevant until much later.

In these sorts of sequences, iterating low-level gameplay is not as important as polish and realization. For example, we could have made a surgery mini-game where you try to save the life of a wounded buddy, but frankly, it would have been focusing on the wrong thing. The gameplay that occurs when a buddy is wounded should not be based on reflex skill, but rather on the challenge of making

## GAME DATA

### DEVELOPER

Ubisoft, Montreal Studio

### PUBLISHER

Ubisoft

### RELEASE DATE

U.S.: October 21, 2008

Europe: October 23, 2008

U.K.: October 24, 2008



### PLATFORMS

PC, Xbox 360, PlayStation 3

### NUMBER OF DEVELOPERS

Year 1: peak 65, including testers

Year 2: peak 105, including testers

Year 3: peak 265, including testers

### TOOLS USED

In-house engine Dunia—built from the ground up for FAR CRY 2

### LINES OF CODE

Approximately 2 million

### LENGTH OF DEVELOPMENT

Approximately 43 months



a difficult moral decision with far-reaching repercussions under various situational pressures. What mattered was not iterating the low-level mechanics until they were fun, but achieving a high enough level of realization that the player would be emotionally invested in the unfolding events and would find the decision to euthanize or abandon his wounded friend more challenging than any test of reflexes.

Unfortunately, we underestimated the degree of realization that was required not only in these sequences, but globally. For the player to care deeply about a buddy bleeding to death in his arms, he must find the character credible and engaging all the time. Only then can the climactic moments of the relationship—the moments that determine the course of the rest of the game—take on the emotional weight and resonance they need.



The FAR CRY 2 development team.

**4) LACK OF AN ANIMATION DIRECTOR.** Further compounding our difficulties in bringing well-realized characters into the game was the fact that we were trying to do it without an animation director. Until the game was in full production, we did not have an animation director at all, and even once we were in full production, SPLINTER CELL: CONVICTION animation director Gilles Monteil was only able to assist us on a part-time basis.

Ubisoft works on numerous high profile, third-person action-adventure games in which strong character animation is absolutely critical. Because FAR CRY 2 is a first-person shooter, animation was (perhaps rightly) deemed less important than for, say, ASSASSIN'S CREED, PRINCE OF PERSIA, or SPLINTER CELL.

There is no argument that FAR CRY 2 had ambitious animation goals. With our determination to stay in first-person perspective with no HUD and with the complex animation demands of the wound system, the in-game map system, and the buddies, we definitely needed an animation director.

What is the mistake here? It's not that Ubisoft could not reassign an animation director to our project, as we fully understand that PRINCE OF PERSIA and ASSASSIN'S CREED could not have shipped at all without one. It's not that we didn't identify our need and try to recruit a new animation director. The thing we did wrong, simply, was that we did not bring the requirements of the project and the reality of the staffing situation into line with one another. Whether



we should have recruited more aggressively or perhaps changed the scope of the artistic vision according to our resources remains unknown because the conversation never happened. But it should have.

**5) MANAGING SINGLE-PLAYER AND MULTIPLAYER TEAMS.** FAR CRY 2 had ambitious multiplayer goals. However, with our aim to build an engine in tandem with shipping a first-person open-world game on three platforms, including multiplayer support and a level editor on PC and console, something was going to crack.

From the beginning, our aim was to keep the single player design team in lock-step with the multiplayer design team, and make sure that the multiplayer experience was elaborating on the same sorts of themes and feelings as the single-player experience. Needless to say the realities of a multiplayer versus single-player game are very different, and this creative vision in itself may have been fundamentally flawed.

Eventually, it was decided to split the multiplayer team into its own group, managed separately from the single-player team. Junior multiplayer designers, without support or oversight from the rest of the design team, were left on their own to defend an innovative vision that had not fully steeped into the growing multiplayer team. This led to conflict, inevitable compromises, and ultimately cascading failure of the entire multiplayer design.

Fortunately, the meltdown happened while the multiplayer engineering team was in the process of building its technology, and the pipeline was ultimately delivered. But as late as alpha, the multiplayer team was still without a design. A consultant was hired to help reboot the process, and the team was able to stabilize and grow from that point, but the fact remains that the majority of the multiplayer design was not defined until close to beta.

## EXPECTING THE UNEXPECTED

**F**AR CRY 2 is not a typical title. It breaks many conventions of both the shooter and open-world game genres. It creates certain expectations on which it fails to deliver, while at the same time delivering something unexpected. It has received highly polarized critical reception, and so far has demonstrated an unusual lifecycle as a game, as a cultural artifact, and as a product.

The mysteries of FAR CRY 2 are not going to be revealed in 10 bullet points, but I hope that those who wish to explore deeper will find these notes from our development process a helpful guide through the massive open world of what FAR CRY 2 actually is. ❖







# Cartoon Network made a AAA in-browser MMO. What will you make with Unity?

A WORLD OF CREATIVE POSSIBILITIES DOESN'T HAVE TO COME WITH A BIG PRICE TAG. THE TOOL THAT CARTOON NETWORK AND OTHER INDUSTRY LEADERS AND INNOVATORS USE CAN BE YOURS FOR \$1499. NO BEGGING, NO NEGOTIATION, NO HASSLES.

COME SEE US IN BOOTH #5110 IN THE NORTH HALL AT GDC  
FIND OUT MORE AT [UNITY3D.COM](http://UNITY3D.COM)



The story that follows is a totally imagined portrayal of a fictitious company, Frenzy AB, a game development team based in Malmö, Sweden. None of the characters are real, but you may recognize some of your own work experiences in their efforts to push the technology limits of PC gaming platforms, beat impossible deadlines, and ready a game for mass market release. Intel supports the work of game developers and celebrates the talent, dedication, and expertise it takes to design and develop a successful game title.

# Extreme Exploits: A Fictional Game Development Scenario

## “Should we simplify the final scenes?”

Ernst Maxlo, the director of software engineering sighed, looking strangely vulnerable with no sign of the façade of invincibility he often used to buoy the team during crunch time. “Or, add another thread to handle the fluid movements?” He rested his elbows on the gun-metal gray surface of his desk, deliberately rubbing his temples with his thumbs. Felix Brixden, lead programmer for Frenzy AB on the *Extreme Exploits* game project, usually saw through his posturing, so there was no reason to maintain the front. The two had survived three previous game releases under insane deadlines and along the way had won each others’ respect: Felix for his cool composure when things started unraveling; Maxlo for his methodical approach to problems and sharp analytical insight into software issues and debugging. While Felix shrugged off pressure and deadlines routinely, however, Maxlo tightened under them and he sometimes struggled to maintain an outer aura of calm.

Felix leaned lightly against the partition, the barrier swaying slightly against his weight, his nose sunburned from a recent trip to southern California for some surfing, and his vision turned inward as he pondered the problem.

“Maybe both,” he said. “The focus should be more on the animations of the cross-country skiers. We don’t need so much activity in the background.”

Some of the climactic scenes of the game had broken down under the combined pressures of displaying wave action on a beach in Iceland, the nearby rocks populated with jostling seals as mist rose off a glacier and seabirds soared through the scene. In testing, as soon as the game players (up to eight at a time) started skiing across the headlands, the movements got choppy unless the video was dropped to a blocky, unsatisfactory resolution. *Extreme Exploits* was designed from the start to support a co-op mode of play, so managing the network communications among a group of players also contributed to the complexity.

“Feeling some pressure?” Felix queried, studying Maxlo’s face.

“No more than usual,” Maxlo replied, smiling wanly. “Are the two new hires up to speed on parallel programming?”

“Mostly,” Felix said. “I’ve arranged training<sup>1</sup> for them, so they will be.”

## Reality Check

<sup>1</sup> Intel offers a number of different venues for programmers to learn more about parallelism and multi-core processors. Starting with the Intel® University Program, Intel participates in curriculum design with top universities around the world (more than 1000 to date) to shape undergraduate and graduate coursework in multi-threaded programming and parallel computing. Through the Intel Academic Community (<http://software.intel.com/en-us/academic/>), faculty members can access ready-to-use class curricula, training, and support. Developers can take Web-based online courses and interact with others in the Parallel Programming and Multi-Core Community (<http://software.intel.com/en-us/multi-core/>). Other educational resources include blogs, forums, videos, and wikis, where top Intel software engineers and community members share their expertise. Intel also works with leading universities from a think-tank perspective, nurturing innovation and creativity, and setting the groundwork for future computer graphics architectures.



*Extreme Exploits* provides gamers vicarious thrills, while delivering some strong, believable simulations of actual extreme sports. Through near photo-realistic imagery, dramatic camera angles, a percussive musical soundtrack, and imaginative storyline, players immerse themselves in some of the most daring adventures on the planet, including river kayaking down the Amazon, glacier skiing in northern Canada, and wingsuit base jumping in the Italian Alps. The game contains some aspects similar to an American reality show, *The Amazing Race*, as players compete against each other, moving from destination to destination, striving to achieve the best showing in each individual sport. From the start, however, the ambitious project goals had strained the talents and resources of the modestly sized development team—and called for some pretty dramatic platform tuning to get the results envisioned.

Light streamed through the windows of the former textile mill as the programming team prepared to gather for its weekly status meeting. The desktop computers populating the partitioned workspaces adjacent to the meeting area typically went through a progression in which new technologies replaced old. The animators, modelers, and custom-tool developers usually acquired the latest technology machines<sup>2</sup>, because their work required the extra horsepower. Lower seniority staff members and those with less-demanding work received the last-generation equipment as it was passed down the food chain. Those machines that no longer had useful value and consumed more power<sup>3</sup> than their processing value justified were donated to a local organization, which found new homes for them in schools, libraries, and small businesses.

In one of a half-dozen actual offices on this floor, Matthew Lindstrom, VP of Finance, peered out the half-shuttered blinds of his office window at the assembling programming team, tilting his head downward for a better long-range view through his bifocals, a faint expression of professorial authority pinching his features. As the oldest member on the Frenzy AB staff (and one of the founding members of the company), he was rarely relaxed. In the current economic downturn, the bank had been tightening the Frenzy AB credit line. At its best, game development was a peak-and-valley affair even if a company could manage a string of successes. In an economic downturn, one or two failed titles (given that each takes two or three years of development) could sink the ship. Though Lindstrom had little technical understanding of the rigors of programming a successful game title, he had a keen understanding of the necessary ebb and flow of kronor needed to keep a business humming.

The team was already assembled around the meeting table as Felix and Maxlo arrived. Carla Fontague, slightly flushed from a lunchtime racquetball game, was seated, bouncing a blue racquetball off the brick wall. Of all the team members, her street cred was deepest from the perspective of the gamer community. She had placed well in several tournaments—including the CGDC Quake Tournament and the Queen of the Hill Tournament—and had only recently abandoned the tournament circuit to work with Frenzy. She was still wearing the protective eyewear from her lunchtime match, the curved plastic eye covers distorting her gray eyes so they looked distinctly feline.

## Reality Check

- 2 Perfectly at home in a game development environment, systems powered by Intel® Core™ i7 processors feature Intel® Turbo Boost Technology for delivering performance on demand and Intel® Hyper-Threading (HT) Technology for handling eight threads on a quad-core processor. For more intense processing requirements, dual-processor workstations powered by the Intel® Xeon® 5500 processor series can adeptly handle heavy workloads running Autodesk® 3ds Max® operations and high-resolution image processing in Adobe Photoshop®.
- 3 Power efficiency is an important characteristic of next-generation Intel® processors. For example, the Intel Core i7 processor shuts down power to idle cores to minimize consumption. The hafnium-based 45nm high-k metal gate silicon technology used in the Intel Core i7 processors results in dramatic reductions in energy consumption.

“Boss,” she said, firing a broad, mildly mischievous smile at Maxlo. “What impossible stuff do you have for us today?”

“The difficult we do immediately,” Maxlo primed them.

“The impossible takes a while longer,” the entire group chanted in the mocking, sing-song voices of children.

“And the saccharine sixties still live,” Carla concluded.

“We’ll run the bug list later,” Maxlo said, as the laughter died down, “but first I want to see who is up for our latest real-world adventure. I’ve made arrangements with a river kayak tour guide to sample some rapids. Any takers? We’ve got three slots open.”

Maxlo was convinced that getting the team members to actively participate in some of the experiences they were animating in the game was the best way to help them capture the essence of the extreme sports. Real-world experience should be part of game design, but he wasn’t sure if that principle should be extended to the wingsuits. Most of his team was fairly adventurous, but there was a thin line between adventurous and reckless.

After some bantering and crosstalk, resulting in the selection of volunteers for the kayak trip, Maxlo turned serious. “We’ve got some performance issues in one of the more complex parts of the program. Let’s talk about some of the options to get us past current obstacles.” Obstacles<sup>4</sup> were a part of the daily diet at Frenzy, and figuring out inventive ways around them occupied most of everyone’s waking hours.

For the next hour, the group talked through the current coding issues, performance concerns, and

suggested enhancements. At the end of the session, Lindstrom poked his head out of his office and crooked a finger at Maxlo. Lindstrom shut the door as Maxlo slipped inside and the group could see, but not hear, the conversation, which quickly turned animated.

Watching from around the table, they took turns putting words into the mouths of the two silent-movie silhouettes.

“I want you to get me a date with that smokin’ waitress at The Spoon,” Carla mimicked, as Lindstrom waved a finger emphatically at Maxlo.

“Hey, she wouldn’t go out with you even if your toupee actually fit,” Felix supplied the words to Maxlo’s heated reply.

Transfixed, they watched the exchange for a few minutes, and then Maxlo noticed the group watching. He seemed to make an effort to deliberately quell his anger. After a few more minutes, he stalked out of the office and disappeared down the corridor toward the outside exit.

“That doesn’t bode well,” Felix noted.

To burn off steam, the staff often converged in the dimly lighted comfort of the gaming lounge for an afternoon break, where the focus of entertainment for the last week had been the Nazi Zombie mode of *Call of Duty\*: World at War*. Fighting the walking, brain-eating dead in co-op mode seemed to burn the corrosion off brain cells and get everyone primed for what often turned into a long evening of overtime coding. After about a half hour (punctuated by the returning Maxlo poking his head in the door—not speaking—conveying with his expression alone that break was over), the

## Reality Check

<sup>4</sup> Intel offers a variety of resources for game development companies and many other independent software vendors to help them plan, develop, and market software applications that take advantage of next-generation hardware architectures. Through the Intel® Software Partner Program, members can access technology roadmaps, an online Visual Computing Developer Community, programming tools, an online catalog, and highly visible marketing opportunities to get to market faster and at lower cost.



group drifted back to individual cubicles, and the rhythmic tap of keys flittered about the room again.

Birgit Spader, one of the newer programmers, let out an audible curse and slapped her palm on the desktop, causing the partitions of her cubicle to quiver and a flock of origami birds, suspended aloft on the cross-frames of a mobile, to begin drifting in slow circles.

"You called?" Carla's slitted eyes appeared over the edge of the partition, questioning.

"Sorry," Birgit murmured. "I'm just not getting this . . ."

"Would an example help?"

"A brain transplant might help," Birgit shrugged.

"Sorry, but Lindstrom ruled out brain transplants for staff members. They're well-known budget busters."

"How about a prefrontal lobotomy to take the edge off . . .?" Birgit asked. Carla chuffed, stifling a laugh.

"OK, OK. I suppose an example might help . . ."

Carla eased around beside her where she could reach the computer keyboard, opened the browser, and navigated to [whatif.intel.com](http://whatif.intel.com).<sup>5</sup>

"I've got the perfect example for aspiring pyromaniacs. Here's an example of threading applied to a group of objects as fire consumes the surroundings, swallows soar overhead, horses move away from the ruckus, and barnyard creatures mill around intelligently, guided by AI, of course."

A few downloads later, Birgit was inspecting the source code comments. Carla guided her through the techniques for running physics, effects, AI, and texture mapping on separate threads until Birgit felt she was getting the hang of it.

"Not so hard when you can see how it's done," Birgit noted.

"Glad to hear it," Carla grinned. "Maybe this will help keep your simmering rage in check."

In the adjacent cubicle, the sharpest coder in the group, Peter Renco, a Russian émigré whose father had been on the wrong side of a political dispute, was humming to himself, usually an indication that he was deep into solving a coding problem. The shelves of his area were completely covered with intricate plastic replicas of space vehicles from various TV and movie series, ranging from *Star Trek's* Starship Enterprise to *Battlestar Galactica's* Cloud Nine to the unwieldy scow, *Serenity*, from Firefly. During intense debugging sessions, he sometimes muttered lines from various shows, offering up a credible imitation of Captain Kirk ("I need more power, Scotty!") and a less credible imitation of Scotty's reply delivered in a strong Russian accent overlaid with a Scottish burr ("I can't hold her together much longer, Captain—I'm giving you everything she's got . . .").

At the moment he was testing the game frame-by-frame using the Intel® Graphics Performance Analyzers,<sup>6</sup> ensuring the game would run smoothly on the large installed base of notebook PCs. The tune he was humming sounding vaguely like a Russian folk tune, and his concentration was complete and imperturbable.

As Renco worked, Maxlo pondered staffing issues. Many studios in the film industry followed the practice of bolstering staff when an animated movie was underway and then laying off individuals at the conclusion of a project. Some companies in the video game industry followed similar cycles, making it difficult for the programmers, artists,

## Reality Check

<sup>5</sup> Code illustrating a variety of multi-threading techniques can be downloaded from [whatif.intel.com](http://whatif.intel.com). The Smoke: Game Technology Demo released in early January 2009 illustrates a framework that supports n-way threading for games. The use of Intel® Threading Building Blocks is demonstrated, and physics simulation, using Havok Physics\*, is included in the demo. All source code and build instructions are provided. Two supporting articles, *An Overview of Procedural Fire* and *Designing a Parallel Game Engine*, offer additional insights into practical parallel programming techniques.

<sup>6</sup> Intel® Graphics Performance Analyzers (Intel® GPA) are a suite of tools that help developers pinpoint performance bottlenecks and optimize games for Intel® Graphics-based PCs. For more information on the tool, visit [www.intel.com/software/graphics](http://www.intel.com/software/graphics).

animators, and story developers to maintain any consistency in their employment. During the lull between projects, many programmers and artists wondered whether a new project would arise before their savings dwindled. They usually did get hired on again, but not always.

Maxlo wanted to buck that trend—to stagger projects so staff members who had been trained during the two or three years it took to complete a project would be around to contribute their expertise to the next project. “It’s not economics so much as it’s a matter of keeping the competitive edge,” Maxlo told Felix one afternoon as they discussed the work ahead. “If I spend time and effort helping a programmer master the latest technologies and hone their skills to a sharp edge, I want that person around next time to contribute at a higher level. I want a well-oiled team where everyone knows how to work together. It’s not that different from building a top-notch football team. You don’t just break up the team every season and start over—at least if you’re winning, you don’t.”

“You think we’re winning?” Felix asked.

“We’re not losing,” Maxlo said.

Another month of coding and the game started to feel like a game. Frenzy had arranged an in-house beta test where a group of game aficionados, snared from the local university, had agreed to try out *Extreme Exploits* and relay their impressions to the development team. Each group of four was set up within an ad hoc network in which they then competed in each of three events that had been reasonably well polished in the beta version, though everyone knew there were still problems in the modules for three other events. This initial feedback, however, would go a long way toward determining whether the game had enough of a built-in fun quotient and enough entertainment value to become a hit in the market. The beta testing

went well, even generating enough spirited play that the gamers were hooting and howling as they got the knack for flying through the air in wingsuits or mastering the S-turns on a downhill ski run.

The critical drive at the moment was to make sure the game ran well enough that there would be no embarrassment at the Game Developers Conference in March. Frenzy was sharing a booth at the San Francisco event with Intel<sup>7</sup> and the eyes of the world, as well as the unforgiving fingers of gamers, would be fixed upon them. In some ways, it was a make-or-break showing. Good reviews and a positive buzz could drive sales when *Extreme Exploits* hit the streets. A few glitches in the code, however, or poor performance on the world’s stage would hobble the unreleased title with a taint that would be hard to escape.

This, of course, meant growing pressure on the team, more hours added to an already overtime-laden workweek, and an unrelenting sense that all their skills and expertise would be tested in the remaining weeks. Felix tried to break the tension by taking the team out for smörgåsbord at a raucous local restaurant that included an open-mike night for local musicians. Sometimes on a Saturday, he arranged cross-country ski outings to get some flex back into muscles atrophied from too much computer work. All in all, the team’s spirit remained high, and the bug list grew shorter while the gameplay grew more polished.

On the flight to San Francisco, Maxlo, Felix, Renco, and Carla talked about differences in Swedish and American culture, Felix trying to explain the best and worst parts of the American experience to the two Swedes, neither of whom had ever been to the United States. “San Francisco isn’t much like the rest of the country,” Felix observed. “In some ways, it’s like its own self-contained country.”

“It is good to be visiting,” Renco said, “but it is better to be home.”

## Reality Check

**7** Members of the Intel® Software Partner Program can also apply to participate in co-sponsored conferences, forums, and events. Intel works closely with selected ISVs in mutually beneficial campaigns to help increase worldwide visibility of products and to provide exposure in new market segments with the potential for significant sales.



The others looked at him, questioningly.

"It's a Russian saying," he explained.

Settled into the booth shared with Intel at the conference, an early morning restlessness was evident as multiple cups of coffee were consumed, booth fixtures adjusted, and computers and projection systems checked and double-checked. Maxlo gave the game system<sup>8</sup> a final checkout, launching and running the first few minutes of *Extreme Exploits*. The game video was routed to a large-screen display suspended above the booth to draw in passersby. The plan was for Carla, using the same deft reflexes acquired in tournament play, to show off the visual splendor of the title by using practice mode to take skiers over jumps, kayakers into whitewater turbulence, and rock climbers over impossible ascents. Anyone attracted by the photo-realistic imagery and expressing interest in the game was welcome to sit down and give it a try.

"We're actually here," Maxlo murmured, peering around the immense room and feeling the energy and excitement building.

"Wherever you go, there you are," Felix added sagely.

"You done good, boss," Carla added. More than once, the team had seriously wondered if it was possible to pull this off; the complexities of the networked gameplay, real-time rendering and animation, and cinema-quality visuals seriously taxed their mettle and skills.

The doors opened and a stream of conference attendees poured into the cavernous hall. Carla took a seat in front of the game machine and did some fancy ski turns on a mogul-bumped slope bordered by the black diamonds indicating an expert-only run. One of the first visitors to stop by the booth seemed riveted by the display. Maxlo noted the man's nametag: *Game Developer* magazine.

"Want to give it a try?" Maxlo asked.

"Gladly," the visitor replied.

Let the games begin. ■

## Reality Check

**8** Amidst growing popularity among gamers, the Intel® Core™ i7 processor Extreme Edition topples existing benchmarks and delivers scorching performance to earn the distinction as the fastest performing processor on the planet. Performance is based on select industry benchmarks, game titles, and multimedia-creation applications. Actual performance may vary. See [www.intel.com/performance/desktop/extreme/](http://www.intel.com/performance/desktop/extreme/) for additional information.

To get the latest information of interest to the visual computing industry, subscribe today to Intel Software Dispatch for Visual Computing at: [www.intelsoftwaregraphics.com](http://www.intelsoftwaregraphics.com)



Intel does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of third-party vendors and their devices. All products, dates, and plans are based on current expectations and subject to change without notice. Intel, Intel logo, Intel Core, and Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\*Other names and brands may be claimed as the property of others. | Copyright © 2009. Intel Corporation. All rights reserved.

**"THERE HAS NEVER BEEN  
A BETTER TIME TO INCREASE  
STUDIO PRODUCTIVITY"**



**Could your teams be more efficient? Ship better games faster.**



*Download a free 2-user solution trial at [www.hansoft.se](http://www.hansoft.se)*

>> Today, game development teams in 20 countries are using Hansoft to leverage productivity and ship better games faster. This started when we revolutionised scheduling to be collaborative and made Agile scalable to the size and specifics of game development. More and more teams are also benefiting from using Hansoft for bug tracking. Fully integrated, easily customised, and with free QA accounts to help you bring Quality Assurance closer to development. Now, with Hansoft 5.3, art pipelines can be created in parallel with sign-off workflows to make asset creation more lean. <<



# AUTODESK MUDBOX 2009

Review by Mike de la Flor

**CREATED BY SKYMATTER, MUDBOX** made its debut in 2006 and became an instant favorite with digital sculptors because of its shallow learning curve, intuitive interface, and powerful 3D sculpting tools. However, it wasn't long before Autodesk took notice and in 2007 acquired Mudbox to fill a gaping hole in its software lineup. Now after over a year in development with Autodesk and many promises of enhanced performance, better sculpting tools, and professional 3D painting, Autodesk has released Mudbox 2009 to much fanfare and some criticism.

## INTERFACE AND WORKFLOW

Autodesk wisely kept the intuitive feel of the original release and has added important workflow improvements. For example, Mudbox 2009 can now easily manage tens of millions of polygons in real-time while sculpting, and the new painting system squeezes every bit of power from the latest GPUs to display complex textures

in real time during painting. Also several operations like texture extraction have been multi-threaded to take advantage of multiple processors.

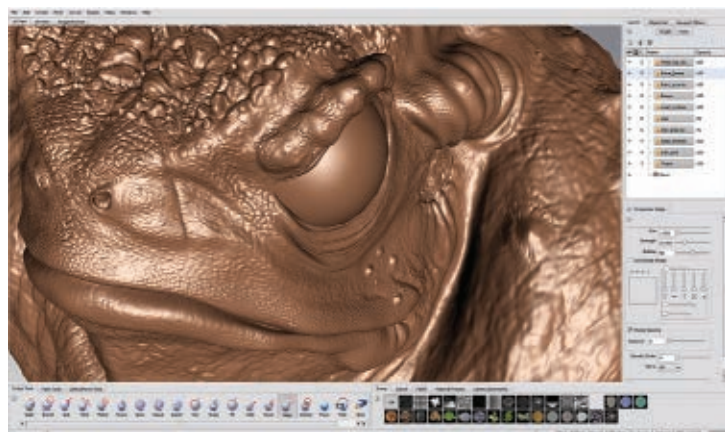
At first glance the interface in Mudbox 2009 appears similar to the old Mudbox, however there are changes that accommodate the new tools and features. Next to the 3D View is a new UV View that displays the model's UV maps, but unfortunately does not do much else. The Layers tab has been overhauled to make room for the new 3D Paint system and next to it are the new Viewport Filters. The tool trays have been reorganized to make room for a slew of new sculpting brushes and paint brushes, and there are new tabs for material and camera presets. Thankfully this release keeps the standard, artist-friendly interface with a few tweaks for the new tools, and the workflow and performance enhancements bring Mudbox up to par for professional pipelines.

## ADVANCED DIGITAL SCULPTING WITH REAL POLYS

The intuitive sculpting brushes that made the initial release of Mudbox a favorite have been considerably expanded in capability and number in Mudbox 2009. In fact, there are so many new brushes that sometimes it is difficult to keep track of which one does what. New brushes include Wax and Scrape, which simulate adding and removing clay or wax, similar to traditional sculpting. Sure to become favorites are the new Grab and Foamy brushes (which do what their names imply). The updated Bulge, Smear, Flatten, Pinch, and Smooth brushes work pretty much as they do in the original Mudbox. There are some differences though—rounding out the new brush toolset are a number of specialty brushes like Spray and Repeat which make tedious sculpted patterns a breeze.

In theory the new brushes can sculpt upward of 100 million polygons in real time with little or no loss in display or sculpting performance. In reality, to get that type of performance out of Mudbox, you would need a powerhouse computer with a lot of RAM, a high-end video card, and a really good CPU. Luckily you don't need a hundred million polygons for most projects. For this review Mudbox was tested on a two-year-old Dell Workstation running XP Pro, with 4GB of RAM, quad-core Xeon processors, and an NVIDIA Quadro FX 4600 video card. I was able to push Mudbox to subdivide into the tens of million of polys with no performance decay, though I did not reach 100 million polygons.

Mudbox's higher-end capabilities are slated for the professional pipelines of the film and gaming industries. Autodesk has pumped Mudbox with the horsepower and scalability to compete in those demanding arenas. The good news is that



Mudbox 2009 has been retooled to manage tens of millions of polygons in real-time while sculpting. On paper Mudbox could display 100 million polys, but that would require serious computing horsepower.

## AUTODESK

★★★★

### STATS

Autodesk, Inc.  
111 McInnis Parkway  
San Rafael, CA 94903  
[www.autodesk.com](http://www.autodesk.com)

**PRICE**  
\$750

### SYSTEM REQUIREMENTS

32-bit version: Windows XP Professional, SP2  
Intel Pentium 4 (or equivalent), 1 GB RAM (2 GB recommended), 650 MB free hard drive space (2 GB recommended), qualified hardware-accelerated OpenGL graphics card, three-button mouse or qualified Wacom tablet

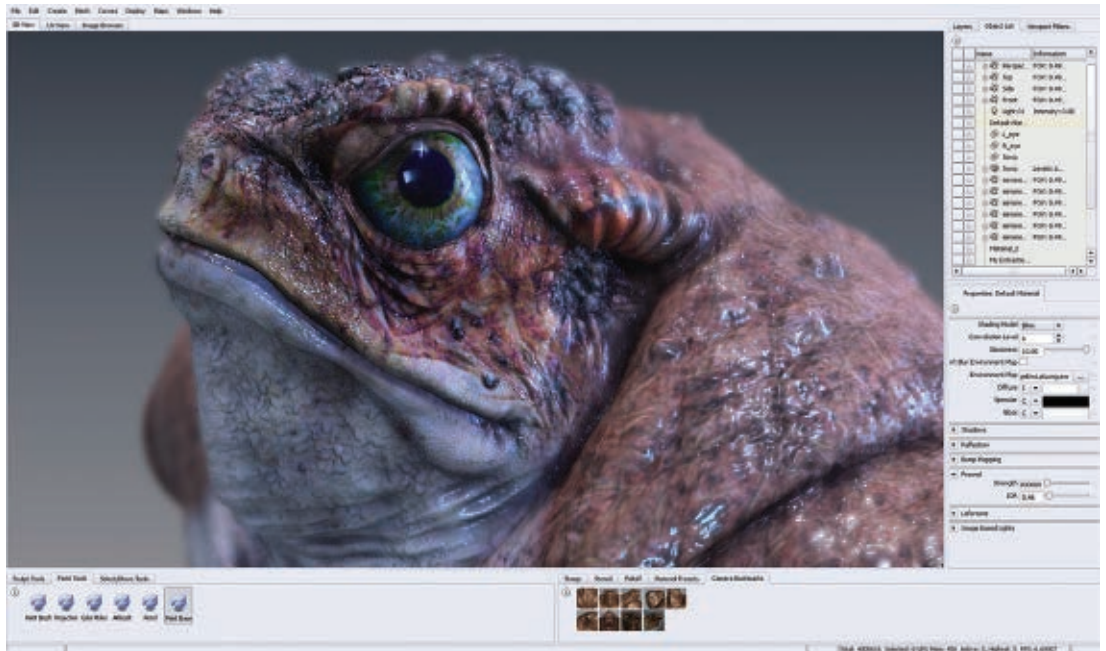
64-bit version: Microsoft Windows Vista Business, SP1, Intel EM64T, AMD Athlon 64, or AMD Opteron processor, 1 GB RAM (2 GB recommended), 650 MB free hard drive space (2 GB recommended), qualified hardware-accelerated OpenGL graphics card, three-button mouse or qualified Wacom tablet

### PROS

- 1 Shallow learning curve and standard interface.
- 2 Advanced, true polygonal 3D digital sculpting.
- 3 Top-notch 3D paint and texture map extraction.

### CONS

- 1 High-end hardware (CPU, RAM, and video card) required to achieve best results.
- 2 No posing tools and some standard paint brushes missing.
- 3 Sparse UV tools.



The production quality 3D paint system makes its debut in Mudbox 2009, but is missing several standard brushes. Nonetheless, the 3D paint system is easy to use and makes creating any complex texture possible.

it will not be long before hardware that can easily manage 100 million or even more polygons becomes affordable and Mudbox will be ready to scale right along.

In Mudbox 2009 the normal and displacement map extraction process has been updated with an improved interface and workflow. The default extraction settings have been optimized for Maya and 3ds Max, and the entire process is multi-threaded to take advantage of multiple processors. This means getting even the finest sculpted detail out of Mudbox and into your target output is a fast and predictable process.

One of the few things that Mudbox 2009 lacks that was indispensable in the initial release is local subdivision. In other words, in order to add local detail to a specific area in the model in Mudbox 2009 you have to subdivide the entire mesh, which seems inefficient. Autodesk counters this criticism by pointing out that Mudbox 2009 has been updated to manage millions of polygons with ease, so local subdivisions are theoretically not necessary.

## PRODUCTION QUALITY TEXTURE PAINTING

As promised, Autodesk delivers production-quality texture painting in Mudbox 2009. However, most of the power in the new production-quality paint system is not

in the new brushes but in its ability to paint multiple diffuse, reflection, bump, and specular textures across multiple maps on dense models regardless of UV distortion or topology. This simply means that very rich and complex textures can be quickly painted in real-time right on the model. However, much like the sculpting brush, to get the most out of the texture painting toolset you will need a hefty video card with at least 512 MB of memory. The various texture types are managed and organized with paint layers which feature modest controls like visibility, locking, and opacity. Unfortunately paint layers cannot have their stacking order changed.

New paint brushes include a handy eraser and a cool 2D Projection Brush, which paints or projects textures of any type onto the model from the camera view. I was disappointed to find that common 3D paint tools such as Smudge, Blur, and Clone (found in programs like modo or BodyPaint3D) were missing. Nonetheless, like everything else in Mudbox, the paint tools are intuitive, powerful, and work exactly as expected.

## REAL-TIME RENDERING

Also new in Mudbox 2009 are Viewport Filters that work in real time to render scenes as they would appear in their target output, such as game consoles. For example, the Tonemapper controls

luminance, while Ambient Occlusion simulates global illumination, and Depth of Field controls focus. When combined with Mudbox's support for advanced image based lighting (including EXR files) and CG shaders, almost any render environment can be simulated.

## THE EXPERIENCE

The best part of working in Mudbox 2009 is the experience. Mudbox is straightforward, intuitive, powerful, and just plain ol' fun to use. The standard interface and shallow learning curve will have you sculpting millions of polygons in minutes, as experience with any polygonal modeling translates instantly into Mudbox. Add top-notch, professional, digital sculpting and production-quality texture painting and Mudbox becomes a solution that is difficult to beat. However, with all that said, keep in mind that to get to the real muscle in Mudbox you may need a monster computer, and there are some critical modeling tools like UV Mapping and Posing, and painting brushes like Smudge and Blur missing. Nonetheless, Mudbox 2009 is still a comprehensive digital sculpting solution that will fit right into most any pipeline.

**MIKE DE LA FLOR** is a Houston, Texas-based medical illustrator and animator, instructor, and writer. He's the author of several CG books and over 75 CG articles. Email him at [mdelaflor@gdmag.com](mailto:mdelaflor@gdmag.com).



# SOLDIERS 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](tel:310.656.6565)  
or by visiting [EndHelmetTyranny.org](http://EndHelmetTyranny.org).

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

New  
 tools  
 for  
 animators

## morphe 2.0 with NVIDIA PhysX NaturalMotion

» NaturalMotion has released morphe 2.0 with integrated NVIDIA PhysX technology. morphe is animation middleware designed to give developers and animators control over the look of their final in-game animation by allowing them to author and preview blends, blend trees, and transition graphs in real time.

morphe 2.0 introduces full integration with NVIDIA PhysX technology, encompassing both graphical authoring in the morphe:connect tool, as well as the animation runtime engine. morphe 2.0 allows for the mixing and matching of animation and physics methods within and across animation skeletons. The tool enables graphical authoring of physics skeletons, collision shape, and joint-limits as well as the mixing of animation with hard- and soft-keyframing and active animation in the same skeleton. It supports different physics modes on

different body parts as well as transitions between animation and physics.

The 2.0 version offers an enhanced multithreading model for runtime performance as well as enhanced animation compression methods. Its LiveLink library promises to simplify connecting applications to remote runtime target and pass-down pins provide support for animation network referencing.

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

## Blade3D Engine Released

### Digini

» Blade3D is built and supported by a group of ex-Microsoft Visual Studios and Games Division executives with the goal of making game development accessible to everyone. It sits on the XNA framework and makes it possible to build games without the need for programming experience, although knowledge of C# allows for the customization of extensions. It deviates

from other engines in the marketplace by offering a subscription model that is as low as \$14.95 per month.

[www.blade3d.com](http://www.blade3d.com)

## Image Metrics Offers New Facial Animation Service Levels

### Image Metrics

Image Metrics has introduced four new facial animation service levels for film and game customers. All offerings use Image Metrics' proprietary facial animation technology to analyze an actor's facial performance and transfer it to a 3D facial rig. The service levels are separated into Value, Pro, Premium, and Elite categories.

The Value service level uses Image Metrics' proprietary technology to automatically generate high volumes of facial animation for secondary characters in games or for pre-visualization purposes.

The Pro service level offers more subtle facial movements on advanced games

**ACTIVISION** How will your characters express themselves?  
 ...absolutely any way you want them to!

**alter ego**

Providing the highest quality, as well as the most flexible, accurate and cost effective facial animation services in the industry

Blending the leading full-performance face+body+camera motion-capture and facial processing technologies to offer an unparalleled range of character performance & pipeline integration services

Infusing your digital characters with the nuances of emotive facial performances - creating a new level of immersion and believability within your pre-rendered & realtime (eg, Unreal3) cut-scenes, trailers and game-play

Join the many industry leaders who are already applying our next-level solutions for their next-gen titles!

619.725.0750

For More Info & Cost Estimates Contact [rob@studiopendulum.com](mailto:rob@studiopendulum.com) [WWW.STUDIOPENDULUM.COM/ALTEREGO](http://WWW.STUDIOPENDULUM.COM/ALTEREGO)

All images are © and TM of their respective holders. All rights reserved.



rigs as well as iterative cycles to help clients capture better performances.

Premium level provides greater creative control with pore-level analysis of facial movement and more fine-tuning of the performance, including optional support for multi-camera performance capture.

Elite offers HD support to optional multi-camera performance capture with superior analysis detail and multiple creative iterations.

For integration into customer pipelines, all service levels include a final output of animation curves that work with the leading 3D software applications.

<http://image-metrics.com>

#### **Allegorithmic's Substance I air** **Allegorithmic**

Substance I air, Allegorithmic's procedural texturing middleware has been updated to improve ease of use with features designed to help artists optimize their

production pipeline. It offers a new runtime engine optimized for generating textures at high speed even on low spec devices and claims the ability to generate detailed graphics that still fit within a few KB, increasing the amount of high-quality content and optimizing the download size by a factor of 10.

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

#### **Enlighten Integrates With Unreal Engine** **Geomerics**

Geomerics' Enlighten now offers integration with the Unreal SDK. The radiosity solution dynamically computes indirect lighting, reducing the time required to light a scene by artists as well as allowing for real-time changes in global illumination. The textures generated by Enlighten store the color and intensity of indirect lighting as well as the major direction of light and a measure of dispersion. This information can then be used in shaders to compute

indirect specular reflections and light normal-mapped surfaces.

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

#### **Softkinetic Releases Iisu 1.5**

##### **Softkinetic**

Softkinetic released an update to Iisu, its 3D gesture recognition software platform, which now includes interface improvements to the middleware's set-up, scene management, and calibration, as well as support for more 3D depth sensing cameras. Iisu recognizes gestures and movements in real-time, captured by a single 3D depth sensing camera. With the SDK, developers can create applications allowing the human body to become the natural input device, removing the need for any physical controllers. The update also adds plug-and-play architecture, a TV gesture plugin, an Iisu Socket Server, and support for Adobe Flash Bridge.

[www.softkinetic.net](http://www.softkinetic.net)

# GAIN INSIGHT INTO NOKIA'S MOBILE GAMING EFFORTS



Experience the technology behind mobile gaming.

Visit booth 5605 in the North Hall

**GDC 2009**  
MOBILE PLATINUM SPONSOR  
n-GAGE | NOKIA | Ovi

[insider.n-gage.com](http://insider.n-gage.com)

© 2009 Nokia

# Game Production

Associate of Science Degree

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

## Learn to Create the Future of Video Games

Join the next generation of gamers, designers, programmers, artists, and producers at The Los Angeles Film School. Learn what it takes to create a video game from start to finish, while working with a game production team in a real world production environment.

Discover the game production industry from the inside, while you go in depth on subjects like game programming, game art, level design, the business of gaming, and much more. It's a complete game production education designed to get you started on a career doing what you love.

Earn your Game Production Associate's Degree in just 14 months.

For more info call  
877.9.LA.FILM or visit LAFILM.EDU

TURN YOUR  
**PASSION** FOR GAMING  
INTO A CAREER

In the Center of Hollywood



6363 Sunset Blvd. Hollywood, CA 90028

© 2009 The Los Angeles Film School. All rights reserved. The term "The Los Angeles Film School" and The Los Angeles Film School logo are either service marks or registered service marks of The Los Angeles Film School, Accredited by ACCET.



### MAX YOUR EDGE WORKING AT WMS

- Lead in the creation and design of video and reel-spinning slot machines
- Innovative designs and games that will lead the gaming industry into the future
- Join a company experiencing exponential growth
- Enjoy the fun work environment which allows you to continually learn and grow



### NOW HIRING

- Software Engineers
- Senior Software Engineers
- Network Engineers
- 2D/3D Artists/Senior Artists
- Game Designers
- Mathematicians
- QA

**WMS**<sup>™</sup>  
MAX YOUR EDGE

COME VISIT OUR BOOTH  
**CP 330**

www.wms.com Chicago-Las Vegas-Reno-London-Sydney





## » THE INNER PRODUCT

## WRITING REUSABLE CODE

## Observations From The Trenches

**AS PROGRAMMERS, WE'RE CONSTANTLY**

reusing code. Sometimes it's in the form of low-level OS function calls or game middleware, and sometimes it's code that our teammates wrote. Unless you're writing top-level game script code, chances are the code you're writing will be used by other humans at some point in its lifetime.

I'm purposefully not labeling reusable code "libraries" or "middleware." Those are just two of the many forms of code we end up reusing. Copying some source files onto your project, calling an API function, and instantiating a class written by someone else on your team are all different forms of code reuse.

**IMPLEMENT FIRST, ABSTRACT LATER**

Without a doubt, the most difficult part of writing reusable code is making sure it solves a problem correctly and meets everybody's needs. That's not an easy task when we're writing code for ourselves, so it's even more difficult when we're doing it for other people. Too many libraries get it only half right, and they solve some problems while introducing a bunch of new ones. Or they force the user to jump through all sorts of hoops to get the desired result.

We need a clear understanding of the exact problem we're trying to solve with our code.

Libraries often fall into the trap of presenting an implementation-centric interface. That is, their interface is

based on the implementation details of the library, rather than how developers are going to use it in their programs.

The best way I've found to address both those shortcomings is to start by implementing code that solves the problem for one person—just one! Forget about multiple users and code reusability for now. If you don't have immediate and constant access to that one person, you need to play that role and create a game or application that is as close as possible to what one of your users is going to be developing.

By using this approach, I find that the interface of the code developed is much more natural because it's based on the experience of having solved the problem at least once. Otherwise, you run the risk of creating an interface that is not a good fit and forcing everything to conform to it in unnatural ways.

Once you have implemented a solution, take a moment to think before you dive into doing any more work. Many times you'll find there's no reason to abstract it any further since your code is only going to be used in one place. If that's the case, step away from the keyboard and go do something more productive. You can always come back later when there's a real need to reuse it later in the project or in a future game.

There are exceptions to this approach of implementing a solution first and abstracting it later. Some problems are very simple or very well understood, so we might be able to jump in and implement the reusable solution directly (for example, an optimized search algorithm, or a compression function). It's also possible you've implemented a similar system several times before, meaning you know exactly at what kind of level to expose the interface and how things should look. In that case, it's perfectly valid to draw on your past experience. Just try to avoid the second-

system effect: the tendency to follow up a successful and simple first system with an overly complex one with all the ideas that didn't make it into the first.

**SETTING GOALS**

Most successful reusable code is created with specific goals about how it is meant to be used. Sometimes those goals are explicitly stated, though most of the time they're implied in the code design.

Some of the most common goals are flexibility, protection, simplicity, robustness, and performance. Obviously, a game programmer cannot meet all those goals at once. Even if you could, you probably shouldn't try. It would be a tremendous waste of time and resources. Stop thinking in the abstract, and start thinking about your one user. What does she need? What is the most important goal for her?

Many APIs and middleware packages are designed with protection as one of the primary goals. They don't want the user to accidentally do anything wrong. In itself, it's not a bad goal, and it can often be implemented by having clean and unambiguous interfaces, clearly named types and functions, and strongly typed data types.

Unfortunately, a design with protection as a main goal can often result in encumbered interfaces, slow performance, and inflexible and verbose code. There is nothing more frustrating than wanting to do something that is explicitly being protected against and having to work around the interface.

If your target users are professional game developers, give them the benefit of the doubt and don't try to overprotect all your code. Save that for the scripting API exposed to junior designers and released to the customers with the game. If you're concerned about programmers using your code correctly and not making mistakes, provide good sample code,

tests, and documentation. If that's not enough and you feel that everyone would benefit from some level of protection, try to keep it to a minimum and maybe even provide lower-level functions that bypass it for power users.

Whatever the primary goal, the libraries

and APIs I prefer to work with help me get whatever I need done, while staying out of the way as much as possible.

### ARCHITECTURE

There are many different ways to architect code that is intended for reuse. The best approach will depend on the particular code: How complex is it? How much of it is there? What are its goals?

Unless your goals are to make quick throwaway applications, I strongly recommend against using a framework type of architecture [see Figure 1]. A framework is a system that allows you to create your program by adding a few bits of custom functionality. Framework architectures seem like they will provide a clean and easy way to use complex code, but they are inevitably very restrictive, making it difficult, if not impossible, to do things with them beyond what they were intended to.

A more flexible approach is to use a layered architecture [see Figure 2]. Each layer is relatively simple and provides a well-defined set of functionality. Higher-level layers build on top of lower-level ones to create more complex or more specific functionality.

Keep in mind that it's not necessary or even desirable to have higher-level layers completely abstract out and hide the lower-level ones. By letting layers be fully transparent, they allow you to mix and match the level at which you want to access the code. This can be very important, especially toward the end of a game's development, when the team is fixing some bugs or trying to squeeze some more performance out of the engine.

Here's an example of how layered architecture can work. One layer can expose functionality to create and manipulate pathfinding networks and nodes. Another layer can implement searches and other queries on those networks. A third, higher-level layer, can expose functions to reason on the state of the network.

A toolkit architecture [see Figure 3] is the most flexible of all. It provides small, well-defined modules or functions that have very few dependencies on

other modules. This allows users to pull whatever modules they need into their games to meet their needs and nothing else. Users can also start by reusing some modules, with the intention of replacing them down the line when they want to go beyond the existing functionality. Needless to say, toolkit architectures are particularly well suited to game development.

### OBJECT ORIENTED

I tend to avoid complex class hierarchies in most of my code, but it's especially important to steer clear of them when writing reusable code. The main drawback of class hierarchies is that they impose a very rigid structure on the code.

If you want to remain object-oriented, a better approach is to emphasize the composition of objects instead of inheritance. It allows users to more easily pull in the functionality they need and create their own objects based on their own constraints.

You can even go a step further and provide purely procedural interfaces: plain static functions that operate on data types. Interfaces based on static functions are often much easier to grasp than interfaces that involve classes and inheritance. They are also much more convenient for users to wrap and use in many different ways.

Do you remember what you learned about object-oriented design and having private data? Forget it and keep everything accessible! You may think you're doing the user a favor by making some variables private and reducing the complexity of the interface—and that's partly true—but eventually, your users will want to have access to some of those variables that you took pains to hide. And if they really want to, they will get to them, even if it means direct addressing into an object or vtable.

It's true that large code bases can be intimidating, and exposing all the internal details along with the regular interface would make it unwieldy for new users. An effective solution to this problem is to separate the public interface from what is intended for internal use only, while still making it available through some other

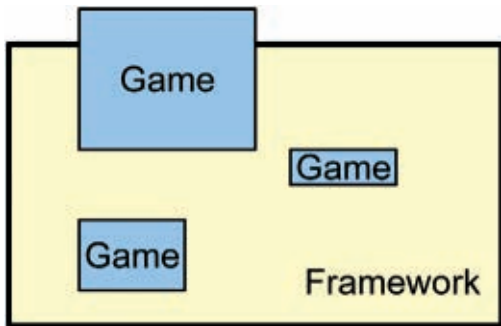


FIGURE 1 Framework Architecture

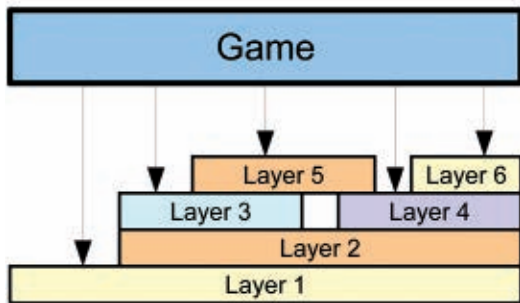


FIGURE 2 Layered Architecture

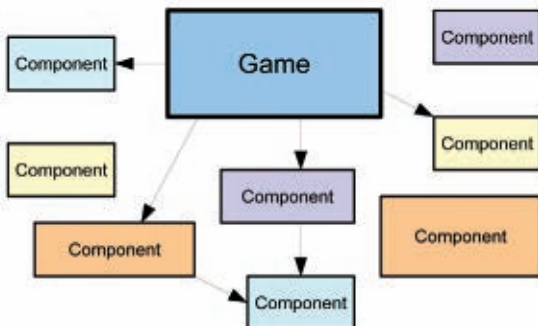


FIGURE 3 Toolkit Architecture



**Perforce** | The *Fast* Software Configuration Management System



## **Perforce Technical Support** **Fast Turnarounds. Precise Answers.**

Our global support teams are always available to share their expertise in person – no scripted responses, answering services, or dispatch centers. At Perforce Software, our highly experienced technical support engineers take pride in providing fast turnarounds with precise answers.

Keeping your projects on track requires a support team that is ready to help right when you need it. You can count on Perforce's Fast SCM System and legendary technical support to give you the winning advantage.

**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.**

All trademarks and registered trademarks are property of their respective owners.

means. For example, private data and functions could be wrapped in a different namespace or simply in a different set of headers. Any method works as long as the interface is clearly separated, but experienced developers can still get to the implementation details to get their hands dirty with them.

### EXTENSIBILITY

As soon as you make your code available to a wide range of developers, you'll find that people want to use it in progressively more complex and bizarre situations. Your code might have completely solved the case for your first couple of users, and since it's layered and modular, it can meet a lot of different requirements, too. But eventually, some people will start taking it to extremes that you hadn't imagined.

You could add more options and more modules and more callbacks to your code, allowing programmers to hook up into almost any part of the code and replace it with their own. The problem, though, is that you've taken something that was relatively simple and made it into a large, ugly, fully customizable behemoth that tries to keep everybody happy. Doesn't that sound like a lot of APIs we know and hate? Most successful products try to meet the needs of some people completely rather than meet everybody's needs part way. That means that, for a small percentage of your users, your code is not going to meet their needs. Instead of altering your code, you should give them the means to adapt it themselves. How? Give them the source code.

Without source code, developers feel caged and constrained. They know they can't look behind interfaces, let alone modify anything in case something goes wrong (and we all know something will go wrong). The more code there is, and the more a project relies on it, the more important it is to have access to the full source code.

Many teams will refuse to use some libraries or middleware unless the full source code is available. As soon as you make source code available to your users, you immediately put them

more at ease because they feel they're in control, and you allow those with special requirements to make whatever modifications they need.

Even those developers without a need to modify the code will be able to browse it and see how certain functions are implemented. Giving developers access to the full source code encourages them to not only use the code, but also fix bugs and suggest performance improvements, so it's a win-win situation for everybody.

For extra bonus points, the source code should be accompanied by a set of tests, and the more comprehensive, the better (unit tests, functional tests, and so forth). Hopefully, you created all those tests while you were developing the code, so distributing it along with the source code shouldn't require any extra effort. But it will make a huge difference to your users. It will give them much more confidence to modify the code to suit their needs if they can see that all the tests are still passing.

### UPGRADES

As soon as you release some code and you have your first user, the question of how to deal with new versions arises. There are many ways you can go about it, depending on how often you'll release new versions, and how important it is to maintain backward compatibility.

One extreme is to change the code and the interface to fit new features, changes in architecture, or for any other reason. Whenever you release a new version, users will have to choose to remain with their current version or upgrade to the latest and make whatever changes are necessary. This is a common approach in open source projects and internal company code.

The opposite extreme is to set the interface in stone, and keep it the same in every version no matter what. This can be good for users because they can get new versions without any extra work on their part, but it can be very constraining as well. It can become impossible to add new features, and it can prevent performance optimizations. This approach is more common on code that will become the foundation of many programs, like OS

libraries and low-level APIs.

A good compromise is to keep interfaces the same during minor version releases and only change them whenever a major version is released. That way, other developers only have to put in the time to upgrade to a major release if they really need the new features at that time.

Another approach is to not change existing functions or classes, but introduce new ones and slowly deprecate the old ones over time. Code continues to work, but users can take advantage of the new functions. After a few versions, you can completely drop deprecated functionality, at which point most people will have upgraded already. If you do this, make sure to label deprecated functions with a `#pragma warning` or something similar, informing developers that the feature will be phased out and that they can start thinking about upgrading to the new interface.

The easier you make the transition to the new version, the better for your users, and the more likely they will be to continue using your code. For example, you can provide scripts that parse their source code and upgrade it to match the new interfaces. That can be a bit risky, but it can be really worthwhile if you have a lot of required changes that are relatively mechanical, such as renamed functions or changed parameter order.

Is there any point to all this talk of interfaces if you made your source code available? Yes, very much so.

Most developers will look at the source code to know how things work under the hood, but they probably won't modify it. Even if they do, they know that's something they do at their own risk, so they'll be more than willing to make a few changes whenever a new version is released. Everybody else will still definitely benefit from a relatively stable interface.

### REDUCE, REUSE, RECYCLE

Writing reusable code starts with solving a problem and solving it well. The rest should all fall from there, and you can pick whichever method is more appropriate for your particular code and how you want to share it. ❖





Instant games.  
(Just add developers.)

**VICIOUS ENGINE**<sup>®</sup>

making games. easier. faster. better.<sup>®</sup>

Visit us @ GDC Booth # 5840 North Hall.  
Book appointments for GDC now.

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

All images and characters are © and TM their respective holders. All rights reserved.



# JOIN *Our* TEAM Disney



## CAREER OPPORTUNITIES

● GAME PRODUCER ● GAME PLAY PROGRAMMER ● GAME DESIGNER ● 3D ARTIST

APPLY ONLINE [DisneyCareers.com](http://DisneyCareers.com) select Disney Interactive Media Group

**Disney**  
INTERACTIVE MEDIA GROUP

©Disney





STEVE THEODORE

## PIXEL PUSHER

# THE BONEYARD

The most important part of your character is invisible

**YOU HEAR THE EERIE RUSTLE OF DRY** bones as they close in around you, the hideous clacking as they shimmy through your pipeline. They are the remorseless bearers of doom. Neither fully alive nor safely dead, they will not rest—nor will you! For they are ...

... the skeletons!

And they're here to make you miserable!

### SHOW SOME BACKBONE

The animation skeleton, like the biological one, is absolutely necessary. We're pretty much stuck with the skeletons we were born with, but for many of our digital creations, the right set of bones takes a long time to develop. The process is often slow and iterative, and if it's handled incorrectly, it can also be very painful.

If you want to exorcise (or exercise) the restless bones in your own characters, you need to start by repeating to yourself this important, but frequently forgotten fact: The skeleton is, bar none, the most important component of any animated character.

The fact that it's invisible doesn't make it some kind of afterthought that can be elbowed off the schedule in favor of more graphically satisfying tasks, like creating concept art or modeling. You must get

the skeleton right if the character is going to succeed aesthetically. And just as importantly, you have to get the skeleton assembled safely if the character is to come in on time.

Let's look at some strategies for taming the skeletal hordes without ending up in the graveyard.

### DEM BONES

If bitter experience hasn't left you convinced that the skeleton is critical, stop and do a little math.

A high quality character mesh represents several weeks of an artist's time. A skeleton, of course, can be assembled in a couple of days. Although the cost of laying down bones may be trivial, the problems that arise when the skeleton has to be changed mid-stream are truly epic.

When you change a character's skeleton, you've invalidated every bit of existing data that depends on the skeleton. This means you may have to reweight its mesh and rebuild every one of its animations. This can be nearly as expensive as redoing them from scratch. It might require weeks or months of work by a whole team of animators, supported by a character rigger. And if the alteration in the skeleton involves a change in the overall proportions of the model, you may need to rework the game mesh as well, which can also result in creating new UVs and thus new textures.

Most teams recognize the costs of skeleton changes and treat them with the kind of enthusiasm usually reserved for a dinner invitation at Castle Dracula. Locking down your skeletons as early as possible is a natural and sensible precaution, given the time costs and potential for chaos that comes with changing them.

Locking down that skeleton is easier said than done, however. The key

to making it stick is planning early. Sending out emails about "discipline" isn't going to cut it. Ensure that your concept process includes the design of the core animations and the skeleton as well as the mesh and textures (we devoted an entire column to this very topic; see "Raw Crude," May 2008). At the very least, you'll need to devise a tough set of standard test animations that a character needs to perform successfully before his skeleton is released to animators and riggers.

Even with planning and tests, you're bound to miss something. For this reason it's good practice to push to the front of the production queue the moves that typically reveal the flaws in a skeleton. That way any changes that become necessary can be caught when only a few animations are in the can.

The worst offenders are often crouching or kneeling poses, which expose poor placement of knee and ankle joints ruthlessly (see "Anatomy for Animators: A Leg to Stand On," November 2005). Gun aiming poses are another good candidate for early testing, as they reveal flaws in the neck and shoulder anatomy as well as the tricky matter of clavicle placement.

### THE HORROR, THE HORROR

Once you've got a skeleton you can believe in, it's still important to distribute it to the animators and communicate that it has been distributed. If you crave ghoulish thrills, watch the eyes of an animator as she realizes she's spent a week perfecting a move on an out-of-date version of the character that won't export anymore. Sending emails or leaving sticky notes on the monitors of harried production folks isn't enough. The tools in your pipeline must make it so that starting new animations is as simple and error-proof as possible.

---

**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).

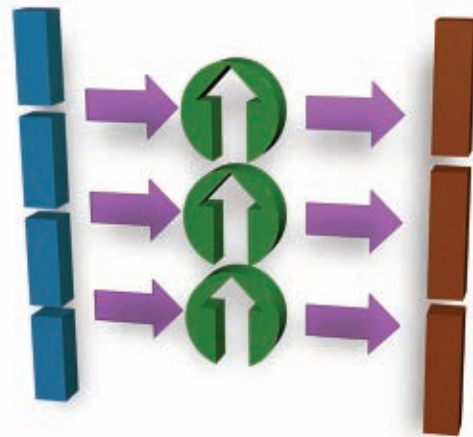
Opinions are split on how to ensure your animators are always working with the authorized skeleton. Some teams like file referencing, while others recoil from it as if it were a necklace of garlic. Referencing proponents point out (correctly) that using references provides free automatic updates and keeps the animation staff current with no extra effort. Detractors counter that referencing breaks too many animation features, makes it hard to apply special-purpose rigs, and has a tendency to break animations when changes do come through. Since referencing tends to arouse a lot of theological passion (see "Clone Wars," October 2008) it's important to focus on the animators' needs rather than fixed ideas about the technology.

Unfortunately, no amount of discipline and planning can keep those restless skeletons locked in the boneyard forever. A good process limits your liability, but it's almost certain that some accident of production or technical glitch will eventually force you to go for a root and branch rework of a character that already has dozens, or even hundreds of completed animations. When that dark day dawns, you're in for some serious pain. However, a little planning and some tools can transform the experience from one of soul-crushing despair into nothing worse than a bad case of indigestion.

The first and most important line of defense is simply to know when things have gone wrong. It's a great idea to check every animation against the "skeleton of record" when you export it. That's a good way to keep people from walking down blind alleys for too long, and it means that minor problems (like bones that accidentally get renamed) can be fixed right when they occur, instead of showing up as subtle bugs or mysteriously failed imports.

### +1 TO TURN UNDEAD

The real horrors, of course, are serious changes to the layout or proportions of the skeleton. These happen for a lot of reasons: a change in the game design, a fatal flaw in the existing skeleton (like an unavoidable gimbal lock that snuck through testing), or an anatomical problem that has to be fixed. When these sorts of situations arise, fixing them is



**FIGURE 1** A Basic retargeting setup. The original skeleton (blue) drives a set of world space constraint objects (green). These, in turn, drive the new skeleton (brown) even if it has different orientations or proportions.

going to be costly. The only question is how much.

The basic strategy for coping with serious skeleton changes is to retarget your existing animations onto the new skeleton, in much the same way that you might apply animations from a mo-cap actor onto a monster. If you already have in-house expertise in motion retargeting, you've got a huge leg up on the problem. Experienced MotionBuilder jockeys can be particularly useful when the bones start going haywire.

Of course, as those old MotionBuilder pros can tell you, retargeting isn't free. It's better than fixing animations by hand, but it's hardly free.

The primary goal of retargeting animation is to get something that looks like the original motion transferred onto a new skeleton. But looking alike is not the same as sharing animation curves and keyframe data, or fancy control rigs. If you have to do major surgery on your skeleton during production, retargeting will help you preserve the look of your animations. But the odds are high that you'll end up with files that look like they were produced using motion-capture: dense, keyframe-by-keyframe data on every channel of every bone in the skeleton, whether you need it or not.

If your animators have spent days carefully tweaking the tangents on their f-curves, they're not going to be happy

when they open a retargeted file and see a graph view that looks like the army ant horde from *Indiana Jones and the Kingdom of the Crystal Skull*. The only consolation is that without retargeting, they'd have to open an empty file and start all over from scratch.

If you don't have an in-house expert who knows mocap retargeting, you can build your own retargeting system in MaxScript, Mel, or Python. It would take years to build a homebrew system that can match the sophistication of dedicated motion-editing software, but a basic version should be well within the capability of any good rigger or technical animator. A good suite of retargeting tools is one of the best investments you can make for your character department.

The essence of all retargeting systems is the old rigger's trick known as "faking and baking." (See Figure 1.) The "faking" step involves creating a world space recording of the original animation. You create dummy nodes for every bone in the skeleton and glue them to the original animation using parent constraints. The "baking" step consists of baking the transforms of the constrained dummies so they're independent of the original bones (that's "collapsing controllers" for Max folks, just as capable, though it doesn't sound as snappy). Baked dummies in hand, you can load up the new skeleton and reverse the process,



# GDC 09

visit us at booth #6422, North Hall



*“Vision Engine 7 convinced us all long the line.”*

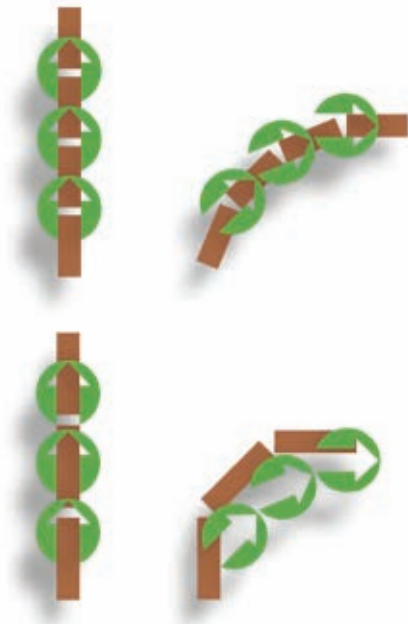
Seung Hoon Han, Team Leader at NEOWIZ®

Vision Engine 7 - leading technology for leading platforms





**FIGURE 2** The animation from the original skeleton is captured on the world space locators. First they are constrained to the original bones, and then the results of those constraints are “baked” or “collapsed” into ordinary keys.



**FIGURE 3** The world space dummies now constrain the new skeleton. If the new skeleton is physically similar (say the rotation orders or joint orientations are the only difference) the animation transfers exactly (top). If the proportions are different, the animation will be similar but not identical to the original—in the bottom example, the third bone’s animation is constrained by the locator from the fourth bone in the original skeleton—this ensures the two chains are parallel, but the three-bone chains is slightly displaced from the original.

constraining the new bones to the existing dummies and then baking the constraints down to keys. (See Figure 2.)

The system automatically accounts for the kinds of changes that make it impossible to simply transplant animation curves. Different rotation orders or re-oriented parents don’t affect the outcome, and the results are as close as the differing skeletons will allow. With the constraint’s ability to maintain initial offsets, you can even compensate for minor changes in position and orientation of the new bones. Although the results of a fake-and-bake retarget are a bit oversupplied with keyframes, they should at least resemble the original animation pretty closely. (See Figure 3.) It’s not MotionBuilder, but it’s a handy hack that doesn’t require a PhD in animation.

### FORBIDDEN LORE

As you experiment with retargeting, you’ll quickly find that there are several common cases that the fake-and-bake technique doesn’t cover.

One common problem is that the animator needs to decide which bones need to mimic the position of their opposite numbers in the old file, and which ones only need rotations. Positional matches are only needed by character roots or IK targets. Most other bones can be retargeted using just rotations. You’ll need to hone reliable, automatic ways of knowing when to use positional retargets and when not to. You don’t want to doom some poor animator to opening hundreds of animations just to select a particular bone and press the “retarget my positions” button by hand.

Another important issue is pose matching. To get good results, you need to get the new skeleton into the right start pose before constraining it to retargeting dummies. Without matching the pose, you can’t leverage the power of the constraint offsets. Again, the key is to find reliable ways to automate that.

Once you’re comfortable with the basics, you can start expanding your repertoire to handle more puzzling situations. For example, once you have a basic set of retargeting scripts, you

can extend them to handle trickier situations. You could apply a global scale by grouping all your retargeting dummies together and scaling them up or down. You could reorient animations by grouping and rotating retargeting dummies. One of the nice things about the dummies is that they are not hierarchical, so your adjustments are applied in a comprehensible way.

### ETERNAL REST?

If this sounds pretty easy, it is—up to a point. Building your own retargeting system, especially if you have no mo-cap experts in-house, is an important investment in protecting your data and keeping your animators from endless do-overs.

It’s important to be realistic, though. Few studios have the technical art chops (or the unscheduled time) to put together a truly bulletproof retargeting system. A successful retarget can involve a serious amount of trial and error. In most cases, there is no truly right way to move motion between skeletons with more than trivial differences, and so you’ll have to rely on the instincts of your retargeting specialist and a lot of very tweaky, case-by-case scripts.

Once animators get over the novelty of your retargeting tools, they’ll be less grateful for their saved work than resentful of all those dense mo-cap-like keys (it’s not fair, but it is human nature). And soon they’ll start demanding the ability to retarget between animation rigs as well as skeletons, a vastly trickier proposition.

If you set out to design the perfect system that will survive all skeletal changes without human intervention, you are going to be very frustrated. If, on the other hand, you can cheerfully stick to a glass-half-full outlook, you can get a lot of value from the simple tricks outlined here. The key is never to forget the terrible curse that is placed on those who trifle with skeletons! Those undead hordes may never be truly laid to rest, but they can certainly be forced back into the shadows where they belong, as long as you are bold enough to grasp the weapons at your disposal. ❖



Switch to DevTrack from ClearQuest®, TeamTrack® or TestTrack Pro® and take advantage of our competitive switch offers

## Choose DevTrack



# DevTrack

The Defect Tracking Choice for Game Development

- Used by the world's largest publishers and studios
- Award-winning workflow design tool
- Integration with popular SCM systems (TFS, Subversion, Perforce, ClearCase, and more)
- Robust, web-service API
- Learn about the ways DevTrack can improve your business at [www.techexcel.com](http://www.techexcel.com)



[www.techexcel.com](http://www.techexcel.com) | 1-800-439-7782

All trademarks are property of their respective owners. © 2008 TechExcel Inc. All rights reserved.



SOREN JOHNSON

## » DESIGN OF THE TIMES

# ASYNCHRONOUS DESIGN

### ONE OF THE FIRST THINGS THAT

separated video games from board, card, and parlor games was real-time interaction. The computer could handle all the details and challenges inherent in allowing two (or more) people to play the same game at the same time. Indeed, despite the term, the first multiplayer video games may have had their roots more in sports than in games. PONG, after all, was inspired by table tennis.

These early experiences were inherently synchronous, meaning players experienced the game together, at the same time, on the same machine. Since then, the synchronous format has been the default model for multiplayer video games, and with the arrival of online gaming, this same experience could be enjoyed even by people who were not necessarily in the same location.

### TOGETHER FOREVER AND NEVER TO PART

The synchronous model is so deeply embedded in the standards and traditions of the video game industry (DOOM, STARCRAFT, MADDEN, and EVERQUEST are all good examples) that few designers consciously consider synchronous play simply as a design choice. But another option does exist: asynchronous play, meaning multi-player games that can be experienced in bite-sized chunks at different times for each player.

The board game world is chock full of games that can be played asynchronously. Play-by-mail chess is a prime example. The most successful game for this format is *Diplomacy*, a classic game of backstabbing that rewards secret negotiations and hidden pacts, which are difficult to achieve if played synchronously. Indeed, with the appearance of the Web, a number of unofficial sites have sprung up giving players a moderated, asynchronous *Diplomacy* experience online.

One of the reasons *Diplomacy* works so well as an asynchronous game is that the turns are executed simultaneously. In other words, unlike sequential games like chess, in which players take turns performing actions, all moves in *Diplomacy* are done at the same time. Players submit their orders secretly to a gamemaster who then handles all interactions and conflicts according to the carefully crafted rules.

This style of play is ideal for an asynchronous experience because all the players get to make a decision every single turn. More traditional board games, from *Risk* and *Monopoly* to *Carcassonne* and *Ticket to Ride*, would slow down to a painful crawl if played asynchronously because the vast majority of turns are spent waiting for other players to make their moves. Thus, asynchronous play favors a specific style of game mechanics, one that minimizes waiting and keeps players involved as much as possible.

### GAMES FOR REAL PEOPLE

Asynchronous games hold a number of advantages over their synchronous counterparts. For starters, the time pressure of a standard turn-based game is eliminated. No longer are four or five players sitting around a table, waiting for the slow player to make up his mind. Rather, a player could take an hour deciding what to do next without

negatively affecting the flow of the game.

Additionally, asynchronous play allows multiplayer gaming—still the richest and most engaging experience available—to fit the schedule of regular people with busy lives and unpredictable free time, across multiple time zones. Few adults can afford the total devotion required to participate in a five-hour, 40-man MMO raid. In contrast, asynchronous play lets a large group of friends play together as long as each player can find 15 minutes a day to check the game. In *Diplomacy*, the English player can submit her moves in the morning, and the French can do it at night, or vice-versa—whatever works best for each one.

Indeed, the ideal online asynchronous game goes a step further than *Diplomacy*, which can still hang if one player neglects to send in a turn, by moving to a real-time format in which the game progresses regardless of an individual player's specific actions. Fantasy sports games follow exactly this model. Once a league is initiated, scores are tabulated each day of the season whether players log on or not. However, the players are all full participants in their league whether they check their teams once every other week or hit the waiver wire multiple times a day.

The strength of this model can clearly be seen by the astounding popularity of online fantasy leagues, with at least 30 million North American players in 2007, according to a recent study by the Fantasy Sports Trade Association. One could even make the case that fantasy sports comprise the most popular form of multi-player gaming in the world. Players with different commitment levels can play together and still enjoy the experience, a statement that definitely cannot be made about the typical RTS.

### LOOKING TO THE WEB

Few good examples of asynchronous gameplay exist for AAA retail video

---

**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).



games, besides some play-by-email modes for older strategy games. For CIVILIZATION IV, we created a PitBoss (persistent turn-based server) option that allowed for large games of up to 32 players, in which players could log on at any time to execute their turns. Combined with simultaneous movement and a 24-hour turn timer, epic games of CIVILIZATION were finally manageable thanks to the asynchronous format.

One could also say that WORLD OF WARCRAFT's focus on solo content is a form of asynchronous play in that players can participate in a traditional MMO without having to juggle the logistics of managing a raid schedule or looking for a pick-up group. Furthermore, leader boards and achievements are also a form of asynchronous interaction layered on top of traditional single-player or synchronous multi-player games, enabling an extra level of socialization for gamers across multiple sessions.

Though some console games have picked up the asynchronous habit, most of the innovative asynchronous titles exist on the web, a platform already built upon asynchronous interactions. Many Facebook games, such as WORDSCRAPER (née SCRABULOUS), manage the persistence of simple turn-based games while using the social networking aspects of Facebook to make it easier to challenge one's friends. Games can be played between two or more friends over a few hours or a few months, whatever matches their level of commitment. Asynchronous MMOs exist as well, such as MOB WARS and KNIGHTHOOD on Facebook, or NILE ONLINE and TRAVIAN on their own sites.

All these games let players grow and develop some entity within a larger world, for prestige or challenge or the simple pleasures of leveling. In NILE ONLINE, for example, players control a city on the banks of the Nile, each one with a unique resource, such as cedar, gold, or oil. As the cities grow, they begin trading with nearby players to acquire the resources they need, perhaps bronze for sculptures or emeralds for jewelry, or to sell their own excess goods for a profit. Eventually, players can see their cities rise in the global rankings or create great monuments for further renown.

### MEANINGFUL INTERACTION?

The challenge with these asynchronous MMOs is that, while they do have some of the advantages of a multiplayer environment, they tend to feel more like a less predictable single-player game. Player interaction is fairly light, as most of the mechanics focus simply on developing one's own domain, without much concern for the neighbors.

Allowing meaningful interaction between players is a challenge because, by definition, the system can only assume one player is logged on at a time. If one player could wipe out another player's city, what if the latter player is asleep? Would it be fun to wake up and discover all of one's hard-earned progress destroyed without a chance to counter the attack?

Thus, most of the games include options to lessen the impact of other players' actions. In TRAVIAN, for example, a player can build a cranny that automatically protects her resources when another player ransacks the town.

However, these mechanics are ultimately self-defeating. Player interaction is either meaningful or it's not. If zero-sum mechanics, like resource raids, are too powerful and negate the advantages of asynchronous play (the ability to set one's own play schedule), then the developers should focus on the parallel competition mechanics of the game instead, building a wonder first or achieving economic dominance.

One asynchronous web-based game that tries to solve this problem while keeping meaningful zero-sum mechanics is DUELS, a fantasy-themed MMO in which characters level up by fighting one another. The system is asynchronous because players do not actually need to be online when their characters fight. Instead, a warrior might challenge a wizard to a duel, which is only played out when the wizard actually accepts the challenge later that same day. The advantage is that while the conflict and interaction is meaningful, the players themselves can still play the game at whatever pace they prefer without worrying about looking for games in the lobby or rage-quitters spoiling the battles.

The problem, though, is that because players can be offline when combat occurs, no meaningful decisions actually happen during the duel itself. Thus, combat is a "black box" which takes in two characters and spits out a result. If a good game should be a series of interesting decisions, DUELS paints itself into a corner by taking control away from the player.

### NATIVE ASYNCHRONOUS PLAY

Truth to be told, asynchronous games are still in their infancy from a design perspective. Their future is promising, as the potential audience for asynchronous multi-player games is much greater than the potential audience for synchronous ones. Although anyone who can find time for synchronous games can find time for asynchronous ones, the opposite is not true.

The challenge is finding game mechanics native to the format itself, ones which make sense only in an asynchronous world, instead of aping mechanics from established synchronous games. The best example of such a game is PARKING WARS, a Facebook game in which players earn money by parking for an extended period of time on another player's street. The trick is that if a car is parked illegally, then the owner of that street can steal all the money the car had earned by handing out a parking ticket.

Thus, the best strategy is knowing what times one's friends are less likely to be checking their streets for illegally parked cars and using that knowledge to earn money. The counter-strategy, of course, is to check one's own street at unexpected times to catch one's friends trying to do the same. Thus, the game cleverly uses the actual time players are offline as the game's content. Unlike the mechanics of the other asynchronous games mentioned previously, the rules behind PARKING WARS could not work at all in a synchronous environment.

Designers of future asynchronous games should follow this precedent. The time has come to stop retrofitting synchronous mechanics into an asynchronous shell and to find the format's native voice. ❖



# I WANT YOU FOR INSOMNIAC

VISIT [WWW.INSOMNIACGAMES.COM/CAREERS](http://WWW.INSOMNIACGAMES.COM/CAREERS)





JESSE HARLIN

## » AURAL FIXATION

# TOOLS OF THE TRADE

## LittleBigChallenges



**MEDIA MOLECULE'S LITTLEBIGPLANET** (LBP) was one of the most innovative platformers of 2008, putting players into a dream world of knit heroes and cardboard props. Aside from the single player game, LBP offers players a rich level creation system, downloadable content, and a multiplayer online network for sharing user-generated content.

Developer Media Molecule's Kenny Young was tasked with tackling the game's audio. I asked him to talk me through the tech, trials, and triumphs of bringing audio to LITTLEBIGPLANET.

**JESSE HARLIN:** *LBP is equal parts sandbox game, side scrolling platformer, and multiplayer party game. Can you start with a bit of a discussion about your creative vision for the game's audio?*

**KENNY YOUNG:** A bit of a Jekyll and Hyde project! The requirements for the audio in play mode were relatively easy to establish—that thought process was very much like traditional game development where you dictate the player's experience. Establishing how a creator will control all of these elements when building their own level in create mode was where the head scratching came in.

It was clear that in order to make the audio features fun and accessible we couldn't encumber creators with having to specify and control every aspect of the audio. Taken to the other extreme, we obviously couldn't handle all the audio automatically because that's getting in to mind-reading sci-fi territory, not to



mention taking away all the fun. The trick was finding a balance between the two.

What helped me find that balance was categorizing the audio into "real sounds" and "imaginary sounds." Real sounds, such as the physics audio, HUD, character and gameplay sounds—are automatically handled by the game. Imaginary sounds are those unknowns which the creator inserts to bring their level to life or influence a player's experience. That's where the sound and music objects (built-in audio tools) come in to play.

In terms of my aesthetic vision for LBP's audio, I took my lead from the game's distinctive style; I guess you could say the game defined its own audio requirements and I just did what Sack Boy told me.

**JH:** *The intro logos show that you used FMOD. Can you talk some about that decision?*

**KY:** We're a small team at Media Molecule (30 of us) and writing our own "next gen" audio tech from scratch would have

unnecessarily strained our resources. I'd previously worked for Sony so I was a ninja SCREAM user, and I loved its granular scripting system; but when I was making these decisions back in 2007, the PS3 version was lacking some features I wanted. At that time, Wwise was unproven as far as I was concerned so going with FMOD was an easy decision.

Although not unique to FMOD, I hadn't had the luxury of abundant real-time parameter controls before—those totally rocked my world and I used them on just about everything. Manipulating sounds to adapt them to the changing context in the game is an indispensable necessity, especially when the *raison d'être* of your game is for players to mess around with everything. FMOD made that very easy to set up.

**JH:** *One of the more charming elements of the level creation tools was the ability for players to remix the interactive music as they saw fit. Did this feature offer any challenges*

**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).

**as you added it into your tool set? Did composers simply deliver stems to you that you then looped and integrated?**

**KY:** The tech is very simple—it's just a couple of multichannel files locked together to give us our six parallel stereo pairs. You could do this using FMOD's event system, or you could just grab the files straight out of a soundbank and manipulate them directly as we did.

When technical director Alex Evans suggested a simple multi-stemmed loop with no transitions or any other sexy interactive music stuff, I was skeptical,

but came around to the idea. From there I devised the concept of the music object and its functionality.

Despite the simplicity of the design, it's

still quite an abstract concept for people to get their heads around. I've noticed that our users easily get the notion of setting the mix to make the music sound the way they want, but then when their level progresses, rather than change the mix the majority of them opt to change the track! That is most certainly their call, and catered for by the tech, but it's fascinating to see something you've designed out in the wild being used and, often more interestingly, abused.

Initially, I was quite hands-on with composer Mat Clark's stems. He's very experienced but hadn't written any interactive music previously and needed guidance on the "gotchas" inherent in having the music's intensity and mix change in real-time. So some of the tracks were a true collaboration, but I eventually moved to acting more as arranger and producer once Mat had found his feet. The interactive tracks are great, with the improvisational performances by the instrumentalists brought in by Mat being a highlight for me.

**JH:** *The level creation tools are a massive part of LBP's charm, at once easy to use and extremely deep. How*

*similar are the in-game tools to your actual development tools?*

**KY:** The in-game tools are the development tools! I add music to levels using the music objects, and I add spot sounds using the sound objects. The line is drawn when I have to set up all the tools so that they work as desired in-game—I need to set the relative levels, fade time and surround send for each music stem, and I need to specify the fall-off and other parameters for each sound effect in a sound object. Then, of course, there are all those automatically handled sounds which the user doesn't need to worry about—those are implemented by a programmer or by myself using our proprietary scripting system.

**JH:** *How did the addition of DLC for LBP affect the design of your audio tech or content?*

**KY:** Basically this meant making sure we had a system in place for patching in new content, and I was always conscious of trying not to shoot ourselves in the foot.

For example, let's assume that we were able to squeeze all of the game's 1200-odd sounds in to memory. Great, but where's the space for the DLC sounds? The real issue here is that it is the player who unwittingly defines the amount of sound in a level, not the audio designer. This isn't a new problem per se—games featuring open worlds have to deal with this same issue. But an open world is a defined, known entity which you can work around, whereas a user-created LBP level is a big unknown. Our solution was to create a culling system which kicks out old sounds when we are running out of RAM. This way, our paltry 16MB sound budget can handle our 40MB main bank of sounds plus as much DLC as we can throw at it.

**JH:** *Stephen Fry's voiceover work gives the game a storybook quality that is both unexpected and fitting. Can you talk some about the choice to use him and if this storybook take was something he brought to the game or your choice of direction?*

**KY:** We identified the need for tutorials to help explain the basics of the game,

as well as the ins and outs of create mode, relatively late in development. But Stephen Fry's name was being thrown about from the start of that process. 1970s British kids TV shows were a reference point for the project, evident in the hand-crafted, worn visual style and the eclectic "2nd hand shop bin" soundtrack. It was clear we wanted a warm, friendly, British granddad-like story-telling voice. Getting Stephen interested in the project wasn't difficult, but finding a couple of hours where the man is free to come in to a recording studio is damned near impossible, so it very nearly didn't happen. But we're very grateful that it did.

My main concern was how the narrator's disembodied voice was going to be introduced to the player. Motion graphics artist Rex Crowle didn't originally conceive his intro as having any voiceover, which would've left us with this voice coming at you out of nowhere and could have felt rather sinister. Fortunately, Stephen's voiceover gelled with Rex's visuals and composer Daniel Pemberton's music perfectly, and the intro video acts as a wonderful, disarming introduction to the narrator character and, of course, the game as a whole.

**JH:** *Is there anything you weren't able to do in LBP that you'd love to do in a sequel?*

**KY:** We don't need a sequel to add cool new stuff, and that's what we're all working on right now. I was able to squeeze in just about every idea on my wish list into v1.0 of the game, (including the squeaky right-stick on the pod computer!). The only things missing which I really wanted were a satisfying "dong" when you slap someone whilst equipped with the frying pan (it's in the soundbank waiting to rock!), and the ability to listen to your own music library whilst in create mode. Sure, there were also those bluesky ideas that were never a serious consideration given the initial time frame of the project, but there's no reason why we can't resurrect those now and use them to help grow the LITTLEBIGPLANET experience and its community. ✨





# We prepare our students well.



Join the growing ranks of industry leaders across the country that rely on DeVry University for qualified, experienced Game and Simulation Programming (GSP) graduates. Our bachelor's degree program provides students with a comprehensive, experiential, career-oriented education.

Please visit us at Booth # 6214 in the Expo Hall at GDC in San Francisco to learn more about our Game and Simulation Programming degree program and our graduates.

**For more information, visit us today at [DeVry.edu](http://DeVry.edu) or call us at 877.273.3879.**

A special thanks to GDC for supplying San Francisco show passes to the 8 student winners of DeVry University's National GSP Contest.

DeVry is certified to operate by the State Council of Higher Education for Virginia. AC0060  
In New York, DeVry University operates as DeVry College of New York.  
Programs vary by location.  
©2009 DeVry University. All rights reserved.

**"Learn from Legends: How to Break Into the Game Industry – and Succeed!"**

Meet two DeVry University alumni who will deliver the straight talk on this rapidly growing and evolving industry.

**Who:** David Crane, co-founder of Activision, Absolute Entertainment and Skyworks Interactive

Steve Cartwright, creator of the Electronic Arts Tiger Woods franchise

**When:** March 26, 2009, 9:00 a.m.

**Where:** GDC 2009 San Francisco

**DeVry**   
**University**

## SAN FRANCISCO BAY AREA

### Early Dreams of the Future



**THE CALIFORNIA DREAM OF GOLD DUST,** celluloid, and silicon has a magnetic pull. Mystics and capitalists of every stripe find themselves drawn to the San Francisco Bay Area, where they often meet with astonishing success or spectacular defeat. Frequently both.

#### ATARI IS GO

As a student at the University of Utah in the mid-1960s, Nolan Bushnell spent his time studying philosophy and the FORTRAN programming language. His downtime was spent immersed in SPACEWAR! battles on the school's mainframe. During summer he worked the midway at a local amusement park learning how to entice players into "just one more try" at the games.

After graduating, Bushnell moved to the Bay area to take a job at the Ampex Corporation. In his off hours he began work on a version of SPACEWAR! that would run on low-cost components rather than the exotic computers on which it had been designed. In 1971 he partnered with Ted Dabney to form Syzygy Engineering and the two created the COMPUTER SPACE arcade machine for Nutting Associates. Unfortunately, while the game was popular with college students, the complicated controls made it a hard sell at working-class pubs.

Despite the failure of COMPUTER SPACE, Bushnell and Dabney incorporated Atari in 1972. The first developer hired at Atari was Al Alcorn who quickly set to work on PONG, completing the first version of

the game in less than two weeks. When the location test machine broke down within a day because it had been jammed with too many quarters, they knew things were going to happen quickly for Atari.

Following the success of PONG, the arcade scene began to bloom not only for Atari, but also for its competitors. After a string of coin-op titles including QUADRAPONG, SPACE RACE, and GRAN TRACK 10, as well as the Kee Games-distributed TANK, Atari created a home version of PONG in 1975.

Encouraged by the success of the HOME PONG console, Atari began work on a far more ambitious cartridge-based machine named Stella. Principally designed by an Atari-owned firm called Cyan Engineering along with assistance from Jay Miner and Joe Decuir (who would later help design the Amiga for Commodore), Stella was released in 1977 as the Video Computer System, also commonly known as the Atari 2600.

Getting the VCS to market was a costly endeavor for Atari and in order to keep on solid financial ground Bushnell sold the company to Warner Communications with Bushnell remaining as chairman. Despite a slow start, the console soon found favor with consumers and would go on to sell more than 25 million units over its 14-year lifespan.

During its formative years Atari had the reputation for being a somewhat loosely-run operation were it would not be unusual to see business meetings conducted through a haze of joints and beer. However, with millions of dollars pouring in from the VCS as well as its thriving arcade business, Atari under Warner became a much more stable and corporate environment. The

relationship between Warner and Bushnell began to sour, and by 1978 he was removed from the company's board. As for Bushnell, he was already looking toward a far more lucrative world where cheap pizza and animatronics beckoned.

#### THE GANG OF FOUR

While Warner's steady hand kept the wheels of commerce turning for Atari, the developers who were creating million-selling games for the company began to wonder why they were still being paid starvation wages.

Initially a group of Atari programmers, including David Crane, Alan Miller, Larry Kaplan, and Bob Whitehead approached the company with a proposal for a standard contract that would give creators design credits and royalties on the games they developed. Dismissed by management, the Gang of Four left Atari and took the bold step of forming Activision in 1979, the first independent development and publishing company for home consoles.

Despite Atari's attempts to derail Activision with lawsuits, the company brought its first games DRAGSTER, FISHING DERBY, CHECKERS, and BOXING to market in 1980. Activision was soon riding high on the crest of the VCS wave. Standout games included Crane's GRAND PRIX and PITFALL, Kaplan's KABOOM!, Whitehead's CHOPPER COMMAND, Miller's STARMASTER, Steve Cartwright's BARNSTORMING, and Carol Shaw's RIVER RAID.

It was a golden time for the VCS with Atari, Activision, and a host of smaller companies reaping huge profits from the console's popularity. However, by 1983 the ride was coming to an end. Costly duds like E.T., a poor port of PAC-MAN, and a profusion of shovelware from fly-by-night publishers had done much to erode consumer confidence in the VCS.

Atari's next-generation machine, the 5200, was introduced in 1982 but struggled to gain a foothold, and was poorly supported by the company. With nothing compelling on the horizon, players began to lose interest, and sales dropped.

What should have been a short-term contraction of the market quickly spiraled into a complete meltdown as smaller publishers went out of business, leaving retailers with unsold cartridges. Unable to return the carts, retailers tossed them into deeply discounted bargain bins where they were in competition with full-priced offerings from Atari and Activision. Feeling the Christmas crunch, shoppers turned to the bargain bins for their games, driving the sales of new titles into the ground.

The crash was devastating to both companies' profits. Warner wanted out, and in 1984 the company was sold off to Commodore founder Jack Tramiel. Activision soldiered on, although its founders had all left by 1986. The company diversified into PC games and productivity software over the next several years but the damage from the crash exacerbated by bad management was too

**JEFFREY FLEMING** is production editor of Game Developer. He lives in Oakland. Email him at [jfleming@gdmag.com](mailto:jfleming@gdmag.com).



great to overcome. In 1991 Robert Kotick bought the company, restructured, and moved it Los Angeles.

## ARTISTS UNITED

Trip Hawkins joined Apple in 1978 just after completing his MBA at Stanford. The company had released its Apple II computer the year before and the machine was beginning its meteoric ascent. By 1980 the Bay Area company had its initial public offering of stock and made overnight millionaires of many of its employees.

Hawkins saw first hand how low-cost personal computers from Apple, Atari, and Commodore were being rapidly assimilated into American life. Machines of sufficient computing power were now on the market to support sophisticated game experiences that went well beyond the simple hand-eye exercises that were predominant on consoles. The time was right for a new computer game publishing company, one that would cater to adults and put video games on equal artistic footing with books, films, and music.

In 1982 Hawkins left Apple and formed Electronic Arts, modeling his new company on the example of the United Artists film studio in which artists maintained creative control of their work. The company's first games were released

in 1983 and included such future classics as Dani Bunten's M.U.L.E. and Bill Budge's PINBALL CONSTRUCTION SET. The packaging of the games was notable for its hip "album cover" graphic design and most importantly, for its emphasis on crediting the developer.

The company was off to a strong start but fallout from the Atari crash began to undermine Hawkins' original business plan. Although Electronic Arts was focused on PC game publishing, retailers were understandably hesitant. Hawkins responded by cultivating relationships with retailers one by one and building a strong sales team that could deal with each directly. As the eighties progressed, Electronic Arts began developing its sports games franchises with JORDAN VS. BIRD: ONE ON ONE and EARL WEAVER BASEBALL. The MADDEN series saw its first iteration as JOHN MADDEN FOOTBALL in 1988 as well.

By the end of the decade the home console was finally emerging from its post-crash deep freeze thanks to the efforts of Nintendo. The Nintendo Entertainment System proved that a market still existed for console games, and Sega was preparing to enter the fray with its new 16-bit Genesis machine. Previously focused on PC titles, Hawkins saw this as an opportunity for Electronic

Arts to dive into console publishing. The Genesis utilized the Motorola 68000 processor—a chip that Electronic Arts was already familiar with from its years of PC development—making it an easy platform to work with.

Electronic Arts was used to publishing games its own way though, and was unwilling to conform to Sega's restrictive licensing terms. Having already reverse-engineered the Genesis, Hawkins threatened to release games without Sega's approval unless the hardware company offered a more favorable deal. With the Genesis' 1989 North American launch fast approaching, Sega relented to Electronic Arts' demands. The result turned out to be a windfall for both companies, as Electronic Arts was able to quickly turn out quality titles like SYNDICATE, KING'S BOUNTY, THE IMMORTAL, STARFLIGHT, and JOHN MADDEN FOOTBALL, ensuring a strong line-up of games for the Genesis.

By 1991 Hawkins was ready to move on to the next big thing—3D polygons and low-cost CD-ROMs. He appointed Larry Probst CEO and ultimately left the company in 1994 so that he could put all of his efforts into creating the 3DO, a machine that he believed would revolutionize the game industry. It didn't quite work out that way, but that is a Bay Area story for another day. ❖



**Riverside Community College District**  
Riverside, CA

Riverside Community College District...  
**your choice, your future.**

## Digital Art Instructor (Norco)

**Salary: \$54,155 to \$83,438**  
**Deadline to apply: 3-16-2009**

The successful candidate will be responsible for lecture and/or laboratory instruction in studio art and digital art including 3-D modeling, animation and/or computer gaming. The assignment may also include other courses in the discipline.

Only online electronic applications  
**Apply at [jobs.rcc.edu](http://jobs.rcc.edu)**  
**For assistance call (951) 222-8595**

An Equal Opportunity Employer

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

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



**THINK SERVICES**  
**GAME GROUP**

**NETWORK OF SITES**



**indiegames.com**

**WORLDS IN MOTION.BIZ**

**GAMES ON DECK**

**FG FINGER GAMING**

**GAME SET WATCH**

**game developer** The Leading Game Industry Magazine

**EVENTS CALENDAR 2009**

**MARCH 23-27, 2009**

**GDC<sup>09</sup>**

Game Developers Conference®  
Moscone Center,  
San Francisco, CA  
[www.gdconf.com](http://www.gdconf.com)

**MARCH 27, 2009**

**GAME CAREER SEMINAR**

Game Career Seminar at GDC09  
Moscone Center,  
San Francisco, CA  
[www.gamecareerseminar.com](http://www.gamecareerseminar.com)

**MAY 12-13, 2009**

**GDC<sup>09</sup> Canada**

Game Developers Conference® Canada  
Vancouver Convention and Exhibition Centre,  
Vancouver, BC  
[www.gdc-canada.com](http://www.gdc-canada.com)

**AUGUST 17-19, 2009**

**GDC<sup>09</sup> Europe**

Game Developers Conference® Europe  
Cologne, Germany  
[www.gdceurope.com](http://www.gdceurope.com)

**September 15-18, 2009**

**GDC<sup>09</sup> Austin**

Game Developers Conference® Austin  
Austin Convention Center, Austin, TX  
[www.gdcaustin.com](http://www.gdcaustin.com)

**OCTOBER 11-13, 2009**

**GDC<sup>09</sup> China**

Game Developers Conference® China  
Shanghai International Convention Center  
Shanghai, China  
[www.china.gdconf.com](http://www.china.gdconf.com)

FOR UPDATES AND MORE INFORMATION ON OUR EVENTS VISIT:

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





At **DigiPen Institute of Technology**, we believe that there are no shortcuts to a serious career in the field of digital interactive entertainment. DigiPen has combined a comprehensive curriculum and world-class faculty to provide a rigorous educational experience in the following programs.

### Computer Science

#### ■ BS in Real-Time Interactive Simulation

The Real-Time Interactive Simulation (RTIS) undergraduate degree focuses on the technology and computer science behind video game development, including the development of game engines, graphics, physics, artificial intelligence, and networking.

#### ■ MS in Computer Science

At the graduate level, students extend their studies to areas such as graphics, physics, artificial intelligence, and game design.

### Production Art

#### ■ BFA in Production Animation

Extensive traditional art and animation skills are taught alongside cutting-edge industry supertools. This approach allows graduates to work in virtually any animation environment.

### Computer Engineering

#### ■ BS in Computer Engineering

This multidisciplinary program integrates the fields of electrical engineering and computer science with a specialized focus on video game applications, such as developing a handheld game console.

#### Game Design

#### ■ BS in Game Design

The BS of Game Design educates students to become technical game designers with the skills necessary to design levels, program, script, and work in this dynamic field.

#### ■ BA in Game Design

The BA of Game Design prepares students to become artistic level designers with the skills to create worlds, levels, and the art for games.

Learn more at [WWW.DIGIPEN.EDU...](http://WWW.DIGIPEN.EDU...)



G<sup>09</sup>DC continued...

 myGDC  
Vault

Exclusive OnDemand access to  
VIDEO · AUDIO · SLIDES  
from GDC events

[myGDC.GDConf.com](http://myGDC.GDConf.com)



# TALENT WANTED

BUILDING THE **NEXT GENERATION**  
GAME COMPANY



**JOBS.MIDWAY.COM**  
CHICAGO SEATTLE SAN DIEGO NEWCASTLE



Wheelman © 2009 Midway Home Entertainment Inc. All rights reserved. WHEELMAN is a trademark of Midway Home Entertainment Inc. Used by permission. MIDWAY and the Midway logo are trademarks or registered trademarks of Midway Amusement Games, LLC. Midway Home Entertainment Inc. and its affiliates do not monitor, endorse or accept responsibility for the content of any non-Midway website. Used by permission. "PlayStation", "PLAYSTATION" and "PS" Family logo are registered trademarks of Sony Computer Entertainment Inc. Microsoft, Xbox, Xbox 360, Xbox LIVE, and the Xbox logos are trademarks of the Microsoft group of companies and are used under license from Microsoft. Software platform logo™ & © IEMA 2006. The Tigon logo is a trademark of Tigon Studios, Inc.



Strong Language  
Suggestive Themes  
Violence



Come see us at the GDC Career Pavilion - Stand #529

# creators of emotion

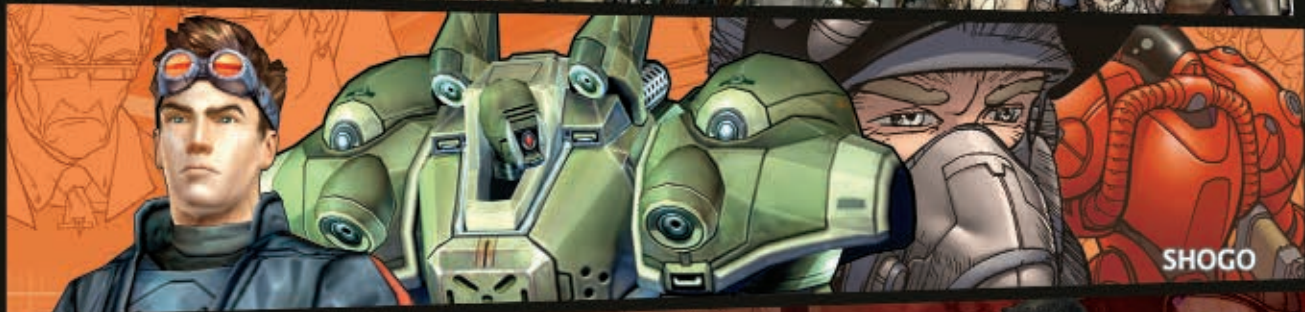
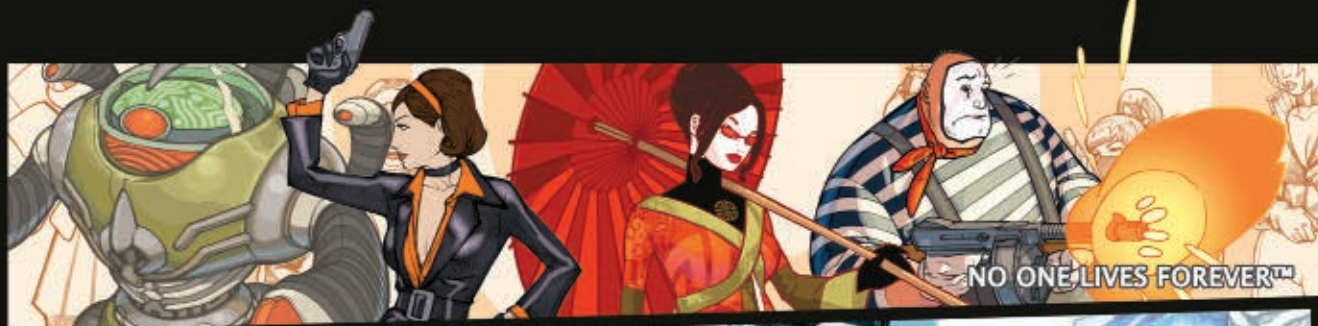
art design programming business >> [www.creatorsofemotion.com](http://www.creatorsofemotion.com)



Montreal · Quebec City · Morrisville (NC) · Paris · Montpellier · Annecy · Newcastle · Malmö · Bucharest · Sofia · Kiev · Dusseldorf · Milan · Barcelona · Shanghai · Cheng Du · Pune · Singapore · Nagoya · Casablanca · Sao Paulo

UBISOFT.COM





Join the developer of No One Lives Forever™,  
Tron® 2.0, Condemned™, Aliens vs. Predator 2™,  
Shogo, and F.E.A.R.™.

**NOW HIRING**   
[WWW.LITH.COM](http://WWW.LITH.COM) **MONOLITH**

Monolith is an Equal Opportunity Employer.  
™ and © indicate trademarks or registered trademarks  
of their respective owners. All rights reserved.



learn  
network  
inspire

# GDC Canada

Vancouver, BC

May 12-13, 2009

Game Developers Conference® Canada

Vancouver Convention & Exhibition Centre

Register by April 15th and save CDN\$100  
on conference passes.

**THINK**  
SERVICES  
A DIVISION OF UNITED BUSINESS MEDIA LLC



[GDC-Canada.com](http://GDC-Canada.com)





# DEADLY SKILLS?



PlayStation®

SCEA will be attending GDC in San Francisco from March 23rd - March 27th.

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

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

Please drop by Moscone Career Pavilion, booth 310 to learn more!

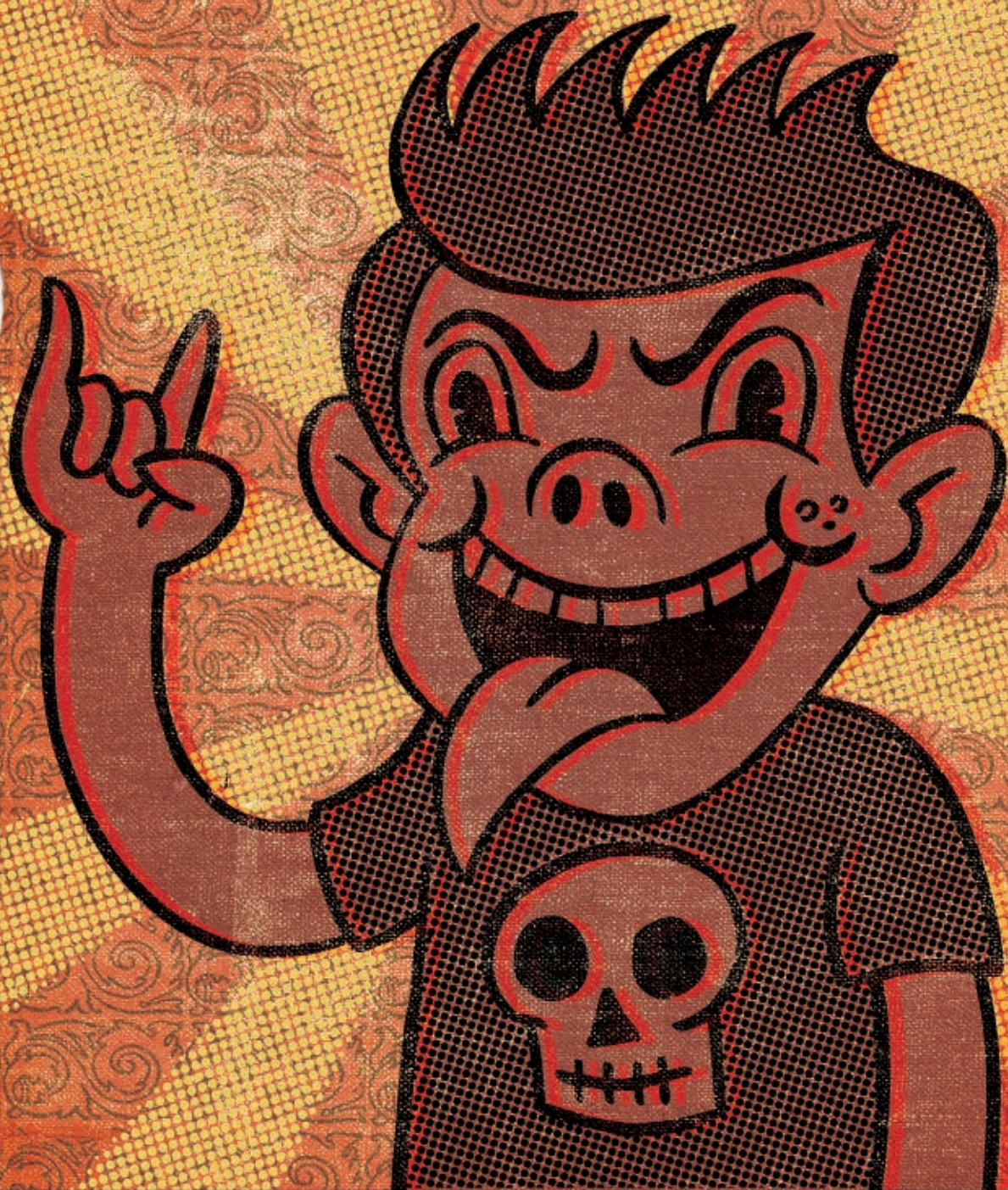
SONY



COMPUTER ENTERTAINMENT



**JOIN NEVERSOFT**  
**WE'RE KICKA%\$ AWESOME**



Neversoff @ GDC 09: Activision Booth #510 West Hall - [www.neversoff.com](http://www.neversoff.com)



# DISNEY INTERACTIVE STUDIOS IS HIRING

OUR DEVELOPMENT STUDIOS ARE LOOKING  
FOR TALENTED:

- Programmers • Artists • Designers
- Producers • Directors



BRIGHTON, UK



VANCOUVER, BRITISH  
COLUMBIA, CANADA



SALT LAKE CITY,  
UTAH, USA



AUSTIN, TEXAS, USA

EMAIL YOUR RESUME TO [JOBS@DISTUDIOS.NET](mailto:JOBS@DISTUDIOS.NET) AND INCLUDE:

- Position that interests you
- Preferred studio

*Develop your career with one of the largest entertainment companies in the world!*



Part of the Disney Interactive Media Group

[WWW.DISNEY.COM/VIDEOGAMES](http://WWW.DISNEY.COM/VIDEOGAMES)

© Disney.

# ACTIVISION®

## GREAT GAMES

START WITH

## GREAT PEOPLE



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

**COME VISIT US AT GDC BOOTH #510**



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



# 109 AWARDS

MARCH 25, 2009  
6:30PM @ GDC

PRODUCED AND HOSTED BY

Game Developers  
Conference

PRESENTED BY



gamedeveloper

11TH ANNUAL



INDEPENDENT  
GAMES FESTIVAL

8TH ANNUAL



OFFICIAL DOWNLOAD PARTNER



THINK  
SERVICES  
A DIVISION OF UNITED BUSINESS MEDIA, LLC

# BILZARD®

ENTERTAINMENT

## 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)

Visit us in the GDC Career Pavilion at Booth #410.



# GET IN TO THE GAME

## TURN YOUR PASSION INTO A CAREER

Game Programming Video Game Data Structures Mobile Game Programming 3D Game Programming XNA Game Development  
Video Game MODification Video Game Data Structures Mobile Game Programming  
Game Programming XNA Game Development  
Mobile Game Programming 3D Game Programming XNA Game Development  
Data Structures XBOX Live Arcade Project



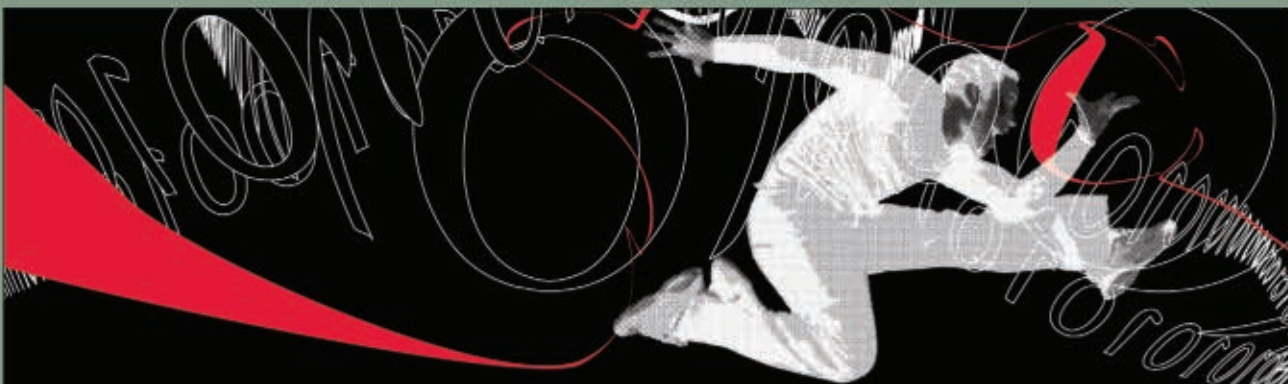
**Video Game Design & Development + Internship**  
<78 week Diploma + 16 week Video Game Company Internship>



**trios COLLEGE**  
*BUSINESS ♦ TECHNOLOGY ♦ HEALTHCARE*

**1.888.805.0533**  
**codemycareer.com**

WINDSOR LONDON KITCHENER HAMILTON MISSISSAUGA TORONTO OSHAWA  
Toronto Campus: 425 Bloor St. E. Suite 200



## BECOME A MASTER OF DIGITAL MEDIA APPLY NOW – DEADLINE MARCH 31, 2009

We offer a 20-month master's degree in entertainment technology and digital media. This powerhouse graduate program combines industry-facing curriculum, real world projects and a 4-month internship in Vancouver, Canada: videogame capital of the world. Learn more. Contact [Alison\\_robb@gnwc.ca](mailto:Alison_robb@gnwc.ca), drop by our booth (#326) at the San Francisco GDC Career Pavilion or visit [mdm.gnwc.ca](http://mdm.gnwc.ca)

**CENTRE FOR  
DIGITAL MEDIA**

Masters of Digital Media Program  
@ The Great Northern Way



EMILY CARR  
UNIVERSITY



*a collaborative university campus environment*

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

VFS student work by  
Thaddeus Maharaj



## THE CAREER-SEEKER'S GUIDE TO GDC

### For Students and Professional Developers

#### GAME CAREER SEMINAR

While the Game Developers Conference has traditionally been seen as a private club for professionals only, the show has made huge strides in the past few years to extend a hand to game development students and other people with dreams of one day getting paid to make games for a living. The most significant bridge between these two worlds is the Game Career Seminar (Student Pass, \$75 at the door).

Sometimes I feel like the Game Career Seminar is one of the industry's best-kept secrets. It's not meant to be secret, but it's still in its nascent years, meaning the people who attend it are blessed with a fairly intimate atmosphere and a good deal of personal attention from and interaction with the speakers in between sessions.

The Game Career Seminar is for people who know they want to work in the game industry some day, but either don't know how to get there, or are close to getting there and need a little more help, guidance, advice, and connections. It's a bridge into GDC, and the fact that it's held at GDC, where attendees can get a good whiff of what goes on at the larger conference, offers something that doesn't happen in other industries.

Topping the agenda at this year's Seminar, the opening keynote address will be an interview with Emil Pagliarulo, lead game designer and writer of *FALLOUT 3*, who will be speaking live on stage with Geoff Keighley of Spike TV. Pagliarulo will share some of his game design philosophies, as well as the story of how he came into his present career in game development (prior to working on games, he fully intended to become an elementary school teacher).

Two other notable speakers are Tom Sloper, who runs the generously devoted yet humble game career advice site [sloperama.com](http://sloperama.com), and Brenda Brathwaite, game design veteran of 20 some-odd years and chair of the game design department at Savannah College of Art and Design.

The knowledge and industry wisdom

that these speakers will graciously share with the attendees could easily take someone years to accumulate on her own. (In other words, going to the Game Career Seminar can be a fast track through a whole lot of B.S.)

#### CAREER PAVILION

The Student Pass grants access to not only the Game Career Seminar, but also the Career Pavilion and the GDC Expo show floor—though only on Friday, March 27. (Most other GDC pass holders have access to the Career Pavilion as well.)

The Career Pavilion at GDC is a one-stop job shop. It provides an extremely efficient way for job seekers to make initial contact with dozens of prospective employers, and vice versa, in just a few hours. While it's best to arrange interviews ahead of time, there's nothing wrong with showing up on the spot with a resume and demo reel, portfolio, and game projects. Know before you go: Do have something to show in addition to a resume.

A few of the confirmed companies which will have a presence in the Career Pavilion include Blizzard, Ubisoft, Bioware, Electronic Arts, Microsoft, Sony, Activision, LucasArts, and THQ. The Pavilion is located on the first floor of West Hall.

#### CAREER BUMPERS

When the economy is in the crapper and people are worried about job security, they often look for proactive things they can do to strengthen their position in the job market and make themselves more valuable to their employers. While the Game Career Seminar is really focused on helping students and other people who are not yet in the industry, GDC does offer a number of one- and two-day tutorials for professionals looking for a bit of career enhancement.

One popular career-enhancing tutorial is Evan Skolnick's (Vicarious Visions) Learn Better Writing in a Day (Monday, March 23). Another great career booster, especially for game developers in programming, art, Q/A, or production

who want to side-wind their way into game design, is Ernest Adams' hands-on Fundamentals of Game Design Workshop (Tuesday, March 24), where attendees will devise a user interface, learn how to produce concept drawings, come up with the main characters and storyline of a game, and brainstorm levels of a game.

Lastly, a two-day career booster for programmers is the intermediate level Math for Programmers/Physics for Programmers, which will take place all day Monday (math portion) and Tuesday (physics portion), March 23 and 24.

Day one will be a shot in the arm about the core mathematics necessary for sophisticated 3D graphics and interactive physical simulations, while day two will be about creating physics engines. Speakers will include Erin Catto, a physics programmer at Blizzard Entertainment, the mellow Squirrel Eiserloh, formerly of Mumbo Jumbo and now with TrueThought, and Jim Van Verth, a senior engineer at Nvidia, among others.

*JILL DUFFY is editor-in-chief of [gamecareerguide.com](http://gamecareerguide.com). Email her at [jduffy@gdmag.com](mailto:jduffy@gdmag.com).*

#### SEMINARS AND TUTORIALS

##### Game Career Seminar

Friday, March 27  
9 a.m.–5 p.m.  
Moscone Center, West Hall

##### Learn Better Writing in a Day

**Evan Skolnick**  
Monday, March 23  
10 a.m.–6 p.m.  
Moscone Center

##### Fundamentals of Game Design Workshop

**Ernest Adams**  
Tuesday, March 24  
10 a.m.–6 p.m.  
Moscone Center

##### Math for Programmers/Physics for Programmers (Intermediate Level)

Monday and Tuesday, March 23 and 24  
10 a.m.–6 p.m.  
Moscone Center

# 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



THE GRADUATE SCHOOL  
OF GAME AND INTERACTIVE  
MEDIA WELCOMES YOU  
IN FRANCE!

- ❑ Video games is your passion? Join us!
- ❑ Master degree in 6 speciality areas
- ❑ Admission: applicants develop a project on a set theme

APRIL THE 2<sup>ND</sup>  
SUBJECT AVAILABLE ON LINE  
[WWW.ENJMIN.FR](http://WWW.ENJMIN.FR)

**ENJMIN**  
121 rue de Bordeaux  
16021 ANGOULÊME CEDEX - FRANCE  
Phone: +33(0)5 45 38 65 68  
Fax: +33(0)5 45 38 65 66  
contact@enjmin.fr

THE ONLY ONE SCHOOL OF ITS KIND IN EUROPE !

Come to see us at the **GDC 2009**, booth #5143, North hall floor

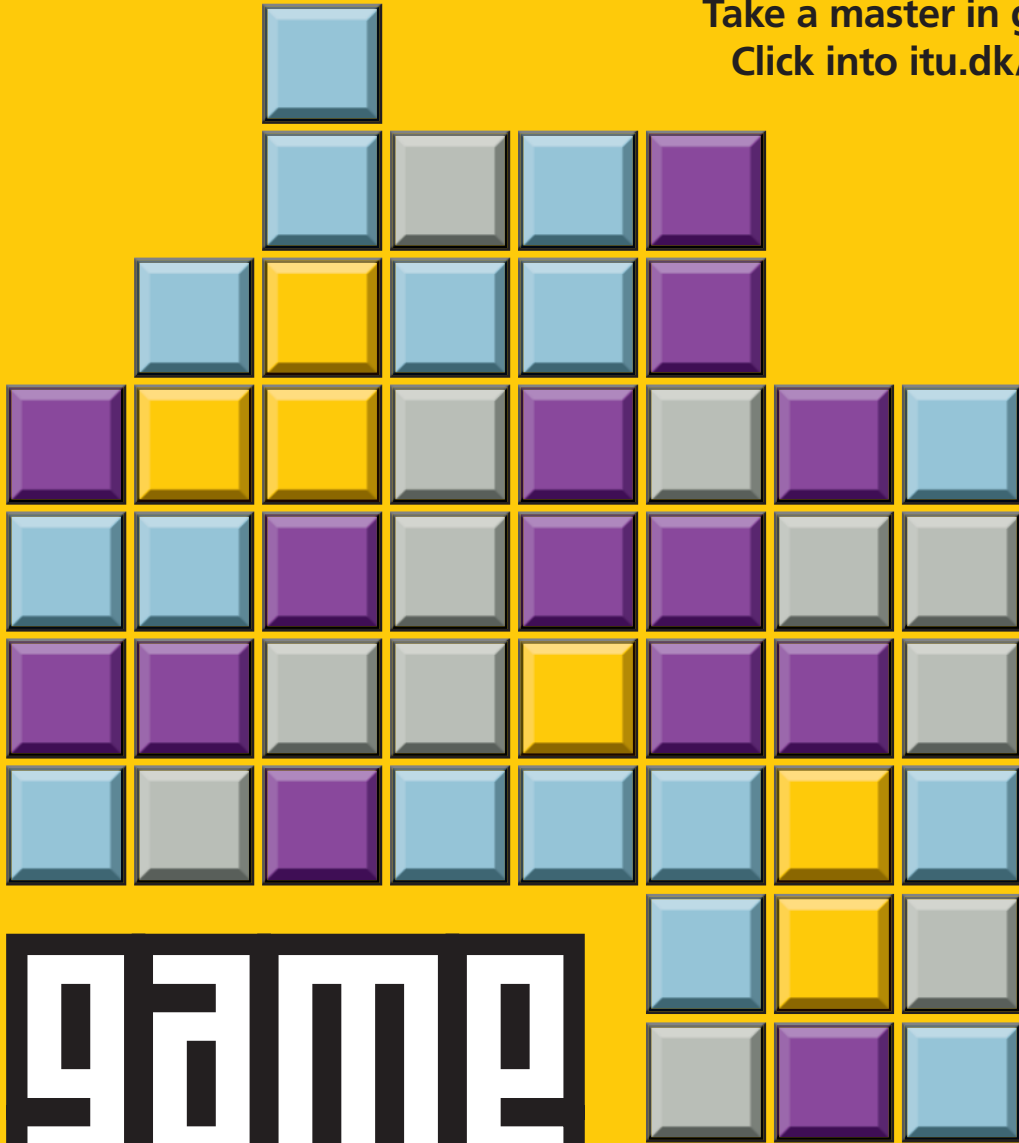
Summer language training  
on the Atlantic coast

-  GAME DESIGN
-  VISUAL DESIGN
-  SOUND AND MUSIC DESIGN
-  SOFTWARE DESIGN AND DEVELOPMENT
-  ERGONOMICS
-  PROJECT MANAGEMENT





Study your passion  
Take a master in games  
Click into [itu.dk/game](http://itu.dk/game)



# game academia denmark

IT UNIVERSITY OF COPENHAGEN

DO YOU IMAGINE A CAREER IN  
**GAME DESIGN OR ANIMATION?**



**Imagine** getting the next “big idea” for a new video game and actually making it happen. You can with an education in Game Design or Computer Animation at the Academy.

Now's the time to take your imagination to the next level!

 **International Academy of  
Design & Technology**  
You imagine. We can get you there.™

Call today for more information

**1.888.704.2111**

CAMPUS LOCATIONS:

**CHICAGO**

One N. State St.  
Suite 500  
Chicago, IL 60602  
[iadtchicago.edu](http://iadtchicago.edu)

**DETROIT**

1850 Research Dr.  
Troy, MI 48083  
[iadtdetroit.com](http://iadtdetroit.com)

**LAS VEGAS**

2495 Village View Dr.  
Henderson, NV 89074  
[iadtvegas.com](http://iadtvegas.com)

**ORLANDO**

5959 Lake Ellenor Dr.  
Orlando, FL 32809  
[iadt.edu](http://iadt.edu)

**SEATTLE**

645 Andove Park West  
Seattle, WA 98188  
[iadtseattle.com](http://iadtseattle.com)

**TAMPA**

5104 Eisenhower Blvd.  
Tampa, FL 33634  
[academy.edu](http://academy.edu)



So you're ready to start developing **the next killer title** that's going to make the big dogs say, **"Holy crap, where did this guy come from?"** but you're going to need a talented team of people to make it happen and you're hoping to find them in a place that's not overpriced and full of posers because you don't have the patience to deal with that on top of the hellish 18-hours-a-day crunch to meet your milestones and you're also thinking that **it wouldn't suck** if someone could **give you money** to help with expenses **before your title is actually launched.**

go to **bradic.org**

**BRADIC**

Visit us at GDC Booth #5201 Contact: Stacey Simmons at 225.389.7182

THINK SERVICES

**GAME CAREERGUIDE.COM**

**GET YOUR GAME ON!**  
**EVERYTHING YOU NEED TO KNOW TO GET INTO THE GAME INDUSTRY!**

- News and FEATURES FOR STUDENTS and educators
- GETTING STARTED section – an invaluable HOW-TO guide
- Message Boards

Download your FREE digital edition of the 2008 game developer Game Career Guide ONLINE AT: [www.gamecareerguide.com](http://www.gamecareerguide.com)

**DIGITAL COUNSELOR** **NEW!**

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

Sponsored by **AI** The Art Institute of Pittsburgh<sup>®</sup> Online Division

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

THINK SERVICES

**GAME CAREERSEMINAR SERIES**

PROGRAMMED BY THE GAME DEVELOPERS CONFERENCE IN ASSOCIATION WITH GAMECAREERGUIDE.COM

**BREAK INTO THE GAME INDUSTRY TODAY!**

*Don't Miss the Opportunity to Network and Learn From Game Industry Professionals.*

APPEARING AT:

**GDC 09** March 27th, 2009

**GDC Canada** May 13th, 2009

WHO KNOWS WHERE WE MIGHT BE NEXT - STAY TUNED.

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



Jobs on **GAMASUTRA**  
The Art & Business of Making Games



The Leading Professional Game Source for Job Seekers and Employers

Access the Web's Largest Game Industry Resume Database and Job Board

Take Control of Your Future Today!

Log onto [www.gamasutra.com/jobs](http://www.gamasutra.com/jobs)




**Cogswell**  
Polytechnical College

- Digital Art and Animation
  - Game Design
  - Modeling
  - Animation
- Digital Audio Technology
  - Audio Engineering
- Digital Arts Engineering
  - Software Engineering
  - Computer Engineering

WASC Accredited  
Financial Aid Available

admissions@cogswell.edu  
800.264.7955

[www.cogswell.edu](http://www.cogswell.edu)  
[youtube.com/CogswellCollege](http://youtube.com/CogswellCollege)

Robot model by John Conelis, © 2009 Cogswell Polytechnical College

COME PLAY THE GAMES CREATED AT THIS YEARS  
**GLOBAL GAME JAM**  
COGSWELL BOOTH #5338

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



# focal press books

KILLER  
GAME DESIGN  
RESOURCES



**Vintage Games**  
By Bill Loguidice & Matt Barton  
ISBN: 9780240811468  
\$34.95



**The Art of Game Design**  
By Jesse Schell  
ISBN: 9780123694966  
\$59.95



**Game Feel**  
By Steve Swink  
ISBN: 9780123743282  
\$44.95



**Game Art Complete**  
Edited by Andrew Gatan  
ISBN: 9780240811475  
\$69.95



Visit [www.focalpress.com](http://www.focalpress.com)! View video tutorials, chapter samples, podcasts and more. Sign up for specialized e-newsletters with tips and specials throughout the year.

*Focal Press helps you maintain your competitive edge. Buy online or in your favorite bookstore today!*

learn · master · create

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

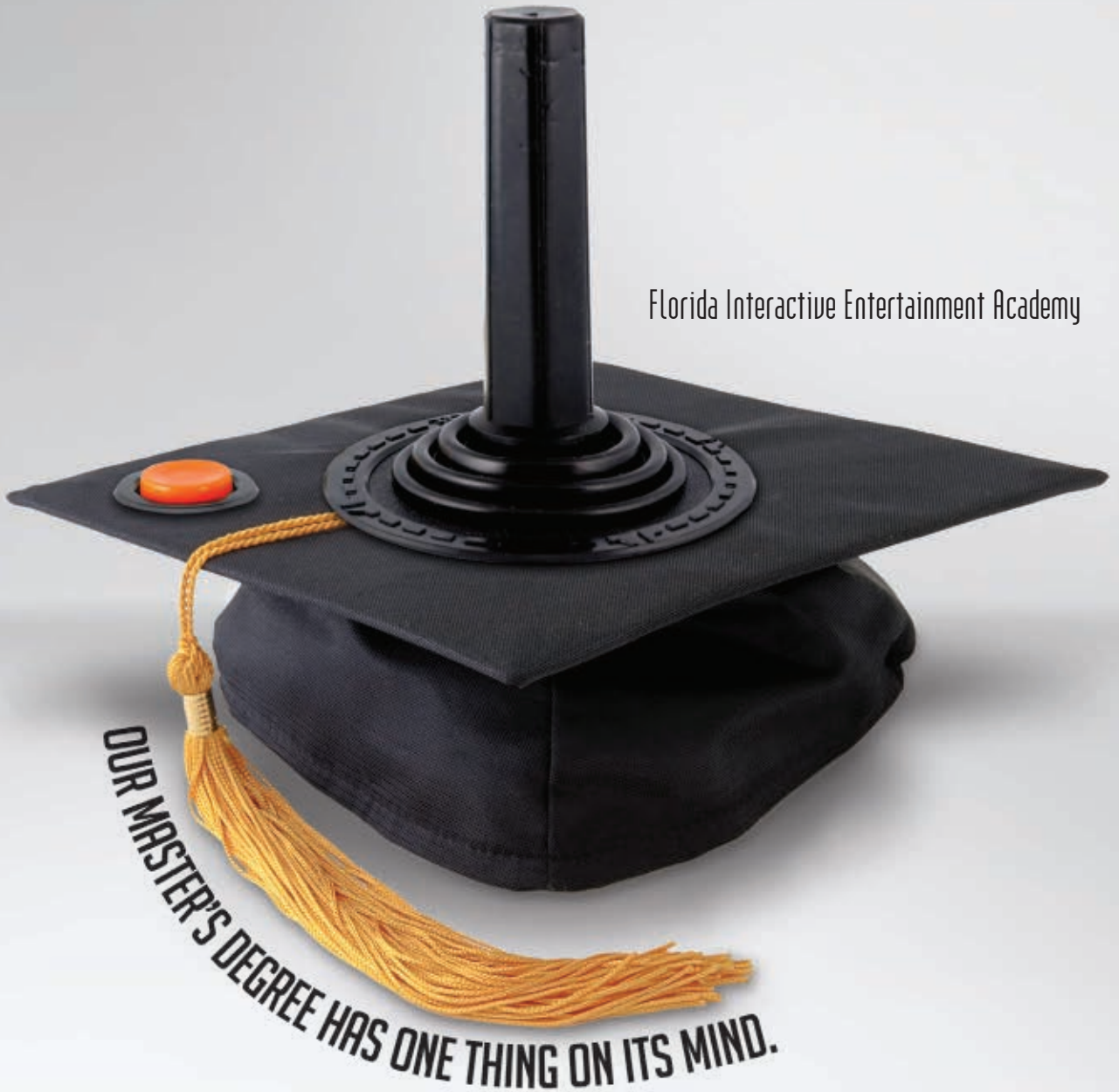
Rochester Institute of Technology's B. Thomas Golisano College of Computing & Information Sciences proudly offers both a BS and an MS in Game Design & Development. Our world renowned games programs are based on a broad view of computing that incorporates several areas of study ranging from the traditional to the extremely applied. We are also the proud recipients of a Microsoft XNA Game Studio Express Innovation Award.

Anyone with a passion for creating the games and game technologies of tomorrow is invited to visit us online at [games.rit.edu](http://games.rit.edu), and at the Game Developer's Conference in 2009!



**GAMES.RIT.EDU**  
STUDENT WORK IN GAME DEVELOPMENT

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)







[ GEEKED AT BIRTH. ]



You can talk the talk.  
Can you walk the walk?  
Here's your chance to prove it.  
Please geek responsibly.

- LEARN:**
- |                             |                                 |
|-----------------------------|---------------------------------|
| ADVANCING COMPUTER SCIENCE  | NETWORK ENGINEERING             |
| ARTIFICIAL LIFE PROGRAMMING | NETWORK SECURITY                |
| DIGITAL MEDIA               | ROBOTICS & EMBEDDED SYSTEMS     |
| DIGITAL VIDEO               | TECHNOLOGY FORENSICS            |
| GAME ART AND ANIMATION      | TECHNOLOGY MANAGEMENT           |
| GAME DESIGN                 | VIRTUAL MODELING & DESIGN       |
| GAME PROGRAMMING            | WEB & SOCIAL MEDIA TECHNOLOGIES |

[www.uat.edu](http://www.uat.edu) > 800.UAT.GEEK  
877.828.4335

# DESIGN YOUR FUTURE

With a career in:  
**Video Game Design**

**With Studies in:**

- 3-D Modeling
- Animation
- Illustration
- Sound and Video Editing



## KEISER UNIVERSITY

Call toll free to speak with  
an Admissions Counselor

**1.888.  
200.5527**

**Admissions Hours:**  
Mon - Thurs 9am - 8pm,  
Fri 9am - 5pm, Sat 9am - 2pm

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

### ADVERTISER INDEX

COMPANY NAME	PAGE
3DVIA .....	3
3VIS .....	17
Activision .....	82
Blade Games World .....	6
Blizzard Entertainment .....	84
BRADIC—Louisiana State University .....	91
Career Education Corporation .....	90
Center for Digital Imaging .....	92
Cogswell Polytechnical College .....	92
Course Technology .....	27
Devry University .....	69
DigiPen .....	73
Disney .....	58, 81
Elsevier .....	93
Enjmin .....	88
Epic Games .....	11
Eyetratics .....	C2
Full Sail Real World Education .....	88
Georgian Dept. of Economics .....	25
Hansoft .....	46
Havok .....	33
Image Metrics .....	49
Insomniac Games .....	66
Intel .....	40-45, C3
IT University of Copenhagen .....	89
Keiser University .....	95
Killer Tracks .....	19
LA Film School .....	52
Masters of Digital Media .....	86
Midway Amusement Games .....	75
Monolith Productions .....	77
Neversoft Entertainment .....	80
Nokia .....	51
North Side .....	14
Pendulum Studios .....	50
Perforce Software .....	55
Pocket Soft .....	71
Rad Game Tools .....	C4
Riverside Community College .....	71
Rochester Institute of Technology .....	93
Seapine Software .....	35
Sony Computer Entertainment .....	79
TechExcel .....	63
Trinity .....	61
TRIOS College .....	85
Ubisoft .....	76
University of Advancing Technology .....	95
Unity Technologies .....	39
University of Central Florida .....	94
Vancouver Film School .....	86
Vicious Cycle Software .....	57
WMS Gaming .....	52
Xaitment .....	29
XSENS Technologies .....	22

Game Developer (ISSN 1073-922X) is published monthly by United Business Media LLC, 600 Harrison St., 8th Fl., San Francisco, CA 94107, (415) 947-8000. 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: \$89.95 (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, 4601 W. 8th 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.



## ARRESTED DEVELOPMENT

# THE VOICE OF EXPERIENCE

## Things I Learned Running a Game Studio

### PAY YOUR EMPLOYEES

The first thing I learned running a game studio is that you have to pay your employees. In fact, it turns out that it's pretty much against the law not to! Plus, everyone will kind of get angry if you don't. Sure, I tried all the tricks: pretending not to hear people when they asked about their salary checks bouncing, telling them the dog ate everyone's W-2s, even using the old standby about the solar flare that knocked out power on the payroll satellite that was in geosynchronous orbit above our location. After a while they're on to these kinds of excuses and there's not a whole lot you can do, other than skip town. So that would be my first piece of advice.

### RETAIN KEY TALENT

Another thing I learned is that some of the people on your team do important work, and if they leave, that's kind of bad. Oh, and get this: did you know that employees are allowed to quit jobs if they don't like them? I mean, just like that? I used to say things like, "if you don't like it here, then why don't you leave" to all the whiners in the office, figuring there was no way they really would. I thought I made it pretty apparent that leaving the company was not what a cool, hip person like me would do. But I guess that's just not enough for some people. So I recommend figuring out who those important guys are, and

making sure they stay around. Threaten them that you'll be really, really sad and disappointed if they accept a job elsewhere. Even pay them if you have to (see above).

### BE NIMBLE, BE FLEXIBLE

Game development is all about thinking on your feet and dealing with situations you didn't expect. One time, I really had to turn on a dime and make our art intern write some network code in order to finish up an MMO we were working on. I saw what needed to be done, so I just ran out to the store and bought him a copy of Visual Basic, since it's, you know, visual, and basic. The tears of happiness and appreciation streaming down that intern's face when I told him I was entrusting the rest of the server-side load-balancing system to him were really touching. Another time, I sped up a project's delivery schedule by deleting the bug database. That's the kind of rapid problem solving you need to perform to survive in this business.

### STRIVE FOR QUALITY

How do you turn your AAAA game into an AAAAA game? Simple: strive for quality in every aspect of the project, but most of all graphics. Now you may not be an artist yourself, but come on, you know what looks like crap when you see it, don't you? Make sure your employees in the art department know that their "good enough" isn't good enough. Use phrases like, "so when is this game going to stop looking like throw-up?" and, "maybe I should fire all you jokers and get some real artists in here." This kind of gentle cajoling will inspire your workers to reach even greater heights! Don't ever let them get complacent.

### MAINTAIN GOOD MORALE

Nothing kills a buzz like pointing out those cold, harsh project realities, so you should avoid doing that at any cost. And if you see anyone else starting to, put a stop to it right away! You've got enough to deal with already without Chicken Little running around saying the sky is falling. Sure, there's the odd challenge here and there: nobody knows what they're doing, the game isn't playable yet but it's time to ship, and so on. The point is, these are things that will be overcome— as long as you don't think too hard about them. Remind the team once in a while that everything is going great, and soon enough it will come true.

### REWARD RESULTS

Finally, you want your employees going after the thing that matters most: results. So you might consider offering a small incentive for when those results are achieved. Don't be stingy, even though you're the one who's done most of the hard work along the way. The guys on the team probably deserve a break once in a while. At the end of our last project, I took every team member who didn't experience a nervous breakdown (reward results, not failure) to a special dinner at Dairy Queen. Nothing says appreciation like getting in on some \$2 for two cheeseburgers action! I even let the art intern borrow a nickel from me when he came up short.

### I DESERVE A BONUS!

That's mostly it! There's a bunch of technical mumbo-jumbo too, but I've found it's okay to ignore all that since it barely affects anything. With my tips and tricks, a couple of movie licenses and a confident smile, you'll have everything it takes to run your very own successful game studio! ✨

---

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

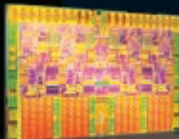


HOW MANY  
PERSONALITIES  
DO YOU HAVE?  
POWER THEM WITH  
VISUAL ADRENALINE.



AFTER ALL...  
**YOU ARE WHAT YOU CREATE.**

Subscribe to Intel® Software Dispatch for  
Visual Adrenaline today and receive relevant  
information about all things graphics and gaming.  
[www.intelsoftwaregraphics.com](http://www.intelsoftwaregraphics.com)



**VISUAL  
ADRENALINE**



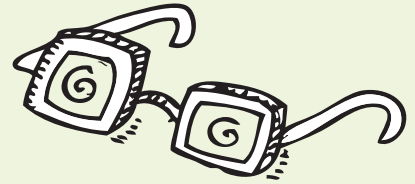
Software

WHAT'S IT LIKE USING



# RAD

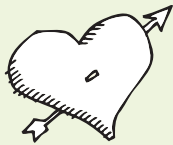
GAME TOOLS ?



It feels like you have



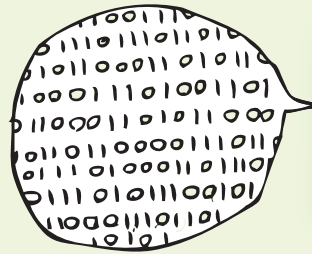
when you use Bink - our amazing, super *FAST* video and audio codec - all in a simple, clean API.



You'll fall in LOVE with Granny, our run-time dynamic

## 3D ANIMATION SYSTEM

with AMAZING content exporters.



And with Miles, our multichannel



sound system, you get the world's *FASTEST*

## MP3 and Ogg DECODERS

+ way cool *new* high-level audio tools.



# IN A WORD, USING OUR TOOLS IS rad!



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