CMP
United Business Media

PRODUCT REVIEWS SOFTIMAGE XSI 6.5 ⁘ DECODA 1.07

FEBRUARY 2008

game developer

THE LEADING GAME INDUSTRY MAGAZINE

WAVE-O-LUTION
A PRIMER ON MASSIVE INTERACTIVE SURF

TOUGH IT OUT
UNLOCK THE POWER OF HARD MODES IN GAMES

MMO MONEY, MMO PROBLEMS
COMMUNITY MANAGERS TELL YOU WHAT THEY REALLY DO

POSTMORTEM:
RATCHET & CLANK FUTURE: TOOLS OF DESTRUCTION

game developer

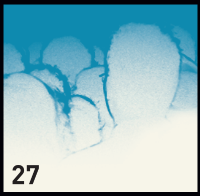
CONTENTS

FEBRUARY 2008
VOLUME 15, NUMBER 2

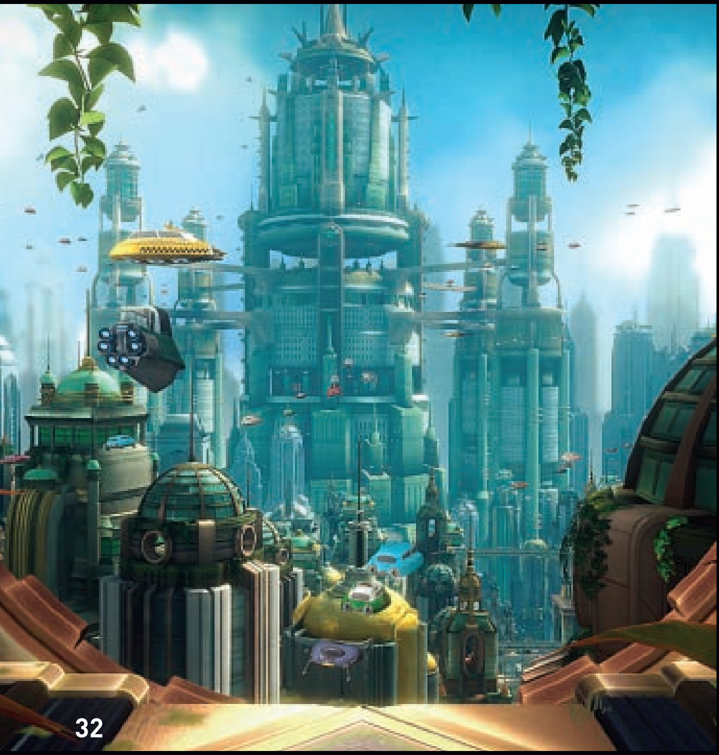
FEATURES

6

14



27



32

POSTMORTEM

DEPARTMENTS

COLUMNS

COVER ART: INSOMNIAC ART TEAM

GAME PLAN

ONE LOVE, ONE FORMAT

LOTS OF FOLKS HAVE BEEN TALKING ABOUT A unified games format recently. I'll leave why I think it's terribly unlikely alone for now, but suffice to say holding consoles, PCs, handhelds, and mobile devices to one standard is going to be one hell of a job. A recent blog post from David Jaffe said "I've yet to have a good argument from anyone ... as to why a single console is wrong." It's true that most reactions have only discussed why it's not possible. I don't think it's wrong, I think it's an admirable goal—but I do see some serious repercussions. It'll be great for consumers, but it would be pretty bad for about 50% of the industry (that's a very rough estimate), and would probably cost a lot of people their jobs.

BIGGER NECESSITATES BETTER

This is an odd position to take, but imagine what would happen if every developer were competing for the same slot—everyone aiming to be king of the hill, without console lines to divide them. Companies like Capcom, Valve, Konami, and Rockstar—they'd all do fine. But what about the mid-level developers? Indies have little overhead, and require low margins—I'm talking about the Italian companies making racing games, or the Japanese companies making giant robot games. These are companies that are competent, but have yet to make a breakout hit to define themselves. Where will their market go if they don't have platforms to define them?

This sounds crazy, but bear with me. Think about the DS market right now. Unless you're Majesco and got in under the wire with Cooking Mama, or Ubisoft with Petz games, releasing a DS title in today's market is like crying into an ocean—nobody will notice your tears unless they're big enough to make waves! What a bizarre metaphor that turned out to be. Anyway, Nintendo just conquered 2007 with both of its consoles, and of course, what do you see on the best-sellers list for DS and Wii? Nintendo games. Are Activision, EA, and Ubisoft on there? Hardly.

It's not only because third parties aren't making games that are good enough. Too many people are trying for the same things. If three equally good RPGs come out for the DS in a given week, the markets for those titles have all been reduced by a third. How many companies cited the "Halo 3 effect" when explaining their poor numbers? Gamers can't necessarily afford to buy Halo 3 and two other games in a given month. But right now, some consumers don't have a 360—and they didn't buy Halo 3, they bought something else.

What if all these games were released on one console? Certainly not everyone wants to play Halo, but 4.8 million people did, in the first few months it was out. Plus, now your game that the publisher told you to make "more like GTA" is now competing against all the other games that publishers told developers to make more like GTA. Of course, this means that in general, games are going to have to get better in order to stand out. I'm in favor of that! But it also means that the developer that makes the slightly less good GTA clone is going to be in the poorhouse pretty quickly, and a lot of people on that team are probably going to be pretty talented. Where will there be room for them? In the 200-person teams working to make a homogenized mass-appeal product. That team will now be much less likely to go on and create the games envisioned when they wanted to get into the industry. After they proved themselves with the GTA clone, they might've been given a shot at an original IP.

But this also brings up another problem. One console or format would mean higher stakes, which would mean bigger risk, which could mean more licenses and lack of control for developers. Increase of potential marketshare makes people with money get both excited and nervous—they want to do more, but they want to make it safer. But maybe this is also a way to circumvent the money? Maybe having one large console space would mean the smallest games can actually get noticed, because they've got the potential attention of the entire gaming populace.

ANARCHY IN THE GAMESPACE

What if Microsoft didn't gate the games on XBLA, releasing them every Wednesday? Would it be as easy for your game to make a splash? Would anyone be able to find it?

That's what I'm not sure of. Thinking about the casual PC market, it's big money—but how do you get the word out? There are tons of these games, so how does one become successful over another? Right now, it's portals that do that, pimping content where it can. Portals are the consoles of the PC, even if you take it up to the Steam level. When I boil it all down, it strikes me that a unified console or standard just doesn't work with our existing publishing and funding models. If we want to move to a single format, we've got to change that first.

- Brandon Sheffield

gamedeveloper
www.gdmag.com

CMP Media, 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*

EDITORIAL
PUBLISHER
Simon Carless *scarless@gdmag.com*
SENIOR EDITOR
Brandon Sheffield *bsheffield@gdmag.com*
SENIOR CONTRIBUTING EDITOR
Jill Duffy *jduffy@gdmag.com*
ART DIRECTOR
Cliff Scorso *cscorso@gdmag.com*
PRODUCTION EDITOR
Jeffrey Fleming
CONTRIBUTING EDITORS
Jesse Harlin *jharlin@gdmag.com*
Noah Falstein *nfalstein@gdmag.com*
Steve Theodore *stheodore@gdmag.com*
Mick West *mwest@gdmag.com*
ADVISORY BOARD
Hal Barwood Designer-at-Large
Ellen Guon Beeman Microsoft
Brad Bulkley Neversoft
Clinton Keith High Moon Studios
Ryan Lesser Harmonix
Mark DeLoura Ubisoft

ADVERTISING SALES
MEDIA ACCOUNT MANAGER
John Watson e: *jmwatson@cmp.com* t: 415.947.6224
GLOBAL SALES MANAGER, RECRUITMENT & EDUCATION
Aaron Murawski e: *amurawski@cmp.com* t: 415.947.6227
SALES REPRESENTATIVE, EDUCATION & RECRUITMENT
Gina Gross e: *ggross@cmp.com* t: 415.947.6241
SR. EVENTS ACCOUNT MANAGER, SOUTHWEST
Jasmin Davé
SR. ACCOUNT MANAGER, EAST COAST, EUROPE & EASTERN CANADA
Cecily Herbst
ACCOUNT MANAGER, WESTERN CANADA, INDIA, AUSTRALIA, & ASIA
Amanda Mae Miller

ADVERTISING PRODUCTION
ADVERTISING PRODUCTION MANAGER Pete C. Scibilia

REPRINTS
PARS INTERNATIONAL
Joe Nunziata t: 212.221.9595 e: *reprints@parsintl.com*

MARKETING
MARKETING MANAGER
Hilary McVicker e: *hmcvicker@cmp.com* t: 415.947.6207

CMP GAME GROUP
SENIOR VP, GAME, DOBB'S, ICMI GROUP Philip Chapnick
GROUP DIRECTOR Kathy Schoback

AUDIENCE DEVELOPMENT
GROUP DIRECTOR Kathy Henry e: *khenry@cmp.com*
DIRECTOR Kristi Cunningham e: *kcunningham@cmp.com*
LIST RENTAL Merit Direct LLC t: 914.368.1000

INTERNATIONAL LICENSING INFORMATION
Mario Salinas t: 650.513.4234 f: 650.513.4482
e: *msalinas@cmp.com*

CMP TECHNOLOGY MANAGEMENT
CHAIRMAN David Levin
CHIEF FINANCIAL OFFICER Scott Mozarsky
PRESIDENT, CMP BUSINESS TECHNOLOGY GROUP Tony Uphoff
PRESIDENT, CMP CHANNEL Robert Faletra
PRESIDENT, CMP ELECTRONICS GROUP Paul Miller
PRESIDENT, CMP GAME, DOBB'S, ICMI GROUP Philip Chapnick
CORPORATE SENIOR VP SALES Anne Marie Miller
SENIOR VP, HUMAN RESOURCES Marvlieu Jolla Hall
SENIOR VP, MANUFACTURING Marie Myers
SENIOR VP, COMMUNICATIONS Alexandra Raine

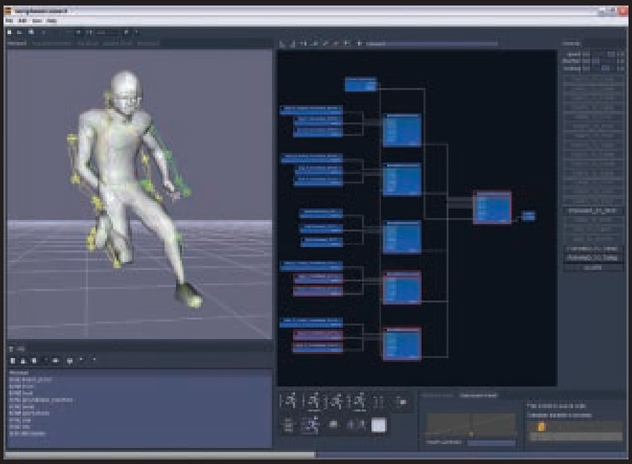
BPA INTERNATIONAL
Game Developer is BPA approved

CMP
United Business Media

WWW.CMPGAME.COM

morpheme™

advanced animation system

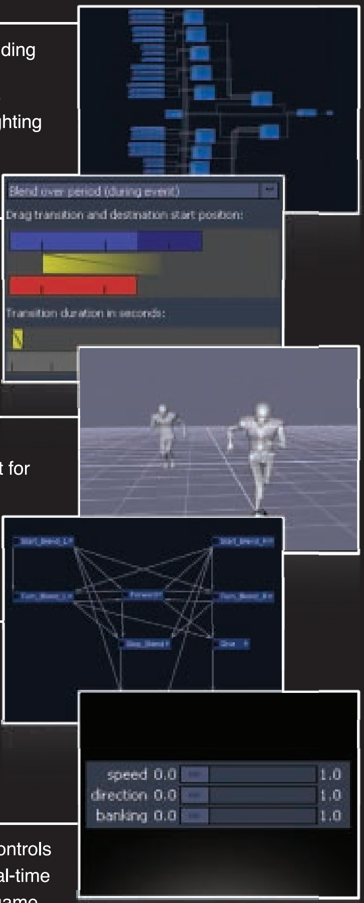


morpheme is the industry's first graphically authorable animation engine. morpheme consists of morpheme:runtime: an advanced runtime animation engine for PLAYSTATION®3, Xbox 360™, Wii™ and PC. morpheme:connect: a highly-customizable 3D authoring application.

morpheme gives animators and developers unprecedented control over the look and feel of their animations in-game: blends, transitions, compression, etc. can all be previewed and modified graphically in morpheme:connect and live on the target platform.

morpheme:runtime ships with full source code and integrates seamlessly with euphoria, NaturalMotion's Dynamic Motion Synthesis technology.

For more information, visit www.naturalmotion.com



Blend over period (during event)
Drag transition and destination start position:
Transition duration in seconds:
speed 0.0 1.0
direction 0.0 1.0
banking 0.0 1.0

blend tree

Advanced graphical tools for building complex blend trees. Real-time visualization of animation source contribution through node highlighting

blending

Graphical control of transition blending between states in the transition graph

multiple characters

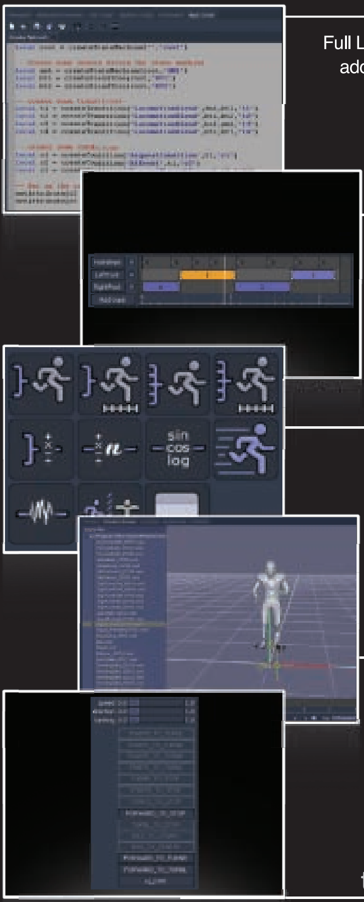
Visualization of multiple runtime characters in morpheme:connect for easy authoring and analysis of character interaction

network

Advanced graphical tools for creating and visualizing transition networks through drag-and-drop

control parameters

Exposure of custom high-level controls for entire animation system. Real-time manipulation through sliders or game pad controller



scripting

Full Lua scripting for automating tasks, adding AI logic or polling game pads for real-time input

timeline

Graphical mark up of animation data to add one-shot and duration events, for highlighting footfalls, sound effects, etc.

node palette

Advanced blend notes for dragging and dropping into transition network. Fully customizable node types through C++ and scripting

animation browser

Easy browsing and selection (drag & drop) of source animation. Animation list is automatically updated to reflect changed source files

transition requests

Exposure of custom transition messages. In-tool emulation of interaction between morpheme:runtime and game AI system

NATURALMOTION

2007 SEES RECORD GAME SALES

$4.8 BILLION IS A LOT OF MONEY, but believe it or not that was the game industry's total revenue for December 2007, up 28% over last year, according to the NPD Group. The charts below show a strong Christmas season with $1.83 billion in hardware sales exceeded by $2.37 billion in software sales.

Dollar Sales	Dec. 2006	Dec. 2007	Annual 2006	Annual 2007
Video Games Total	$3.77 billion	$4.82 billion	$12.53 billion	$17.94 billion
Hardware Total	$1.57 billion	$1.83 billion	$4.58 billion	$7.04 billion
Console Hardware	$1.08 billion	$1.3 billion	$2.96 billion	$5.12 billion
Portable Hardware	$489.5 million	$525.7 million	$1.62 billion	$1.92 billion
Game Software Total	$1.74 billion	$2.37 billion	$6.46 billion	$8.64 billion
Console Software	$1.22 billion	$1.8 billion	$4.77 billion	$6.64 billion
Portable Software	$519.4 million	$574.9 million	$1.69 billion	$2.00 billion
Accessories	$455.4 million	$621.6 million	$1.49 billion	$2.26 billion

The best selling game machine for December was the Nintendo DS with 2.47 million units sold. The Wii followed, putting 1.35 million units under the Christmas tree, slightly edging out the Xbox 360, which moved 1.26 million units.

NPD's numbers show the PlayStation 2 continuing to outsell the PlayStation 3, with 1.1 million units sold in December over the PlayStation 3's 797.6 thousand. The PlayStation Portable continues to make good numbers, selling 1.06 million units in December, and has sold 10.47 million units during its life-to-date.

Nintendo had three titles in the list of December's top ten games including Super Mario Galaxy in the number two spot with 1.40 million copies sold. Activision topped the chart with the Xbox 360 version of Call of Duty 4: Modern Warfare, which sold 1.47 million units. Interestingly, the number three bestselling game for December was Activision's PlayStation 2 version of Guitar Hero III: Legends of Rock which sold 1.25 million copies, compared to the Xbox 360 version's respectable, but far lower number of 624.6 thousand copies.

Hardware Units	Dec. 2007	Annual 2007	Life-to-Date
PlayStation 2	1.1 million	3.97 million	41.12 million
PlayStation 3	797.6 thousand	2.56 million	3.25 million
PlayStation Portable	1.06 million	3.82 million	10.47 million
Xbox 360	1.26 million	4.62 million	9.15 million
Wii	1.35 million	6.29 million	7.38 million
Nintendo DS	2.47 million	8.50 million	17.65 million

Looking at the figures for the entire year of 2007 should warm corporate hearts across the country. As a whole the video game industry brought in $17.94 billion during the year, up 43% over 2006. Hardware takes up a significant portion of that number, leaving $8.64 billion in game sales for developers and publishers to divvy up.

The Nintendo DS was the top selling system of the year with 8.50 million units sold and the Wii came in a close second, moving 6.29 million units in 2007. Although third in 2007 sales, Microsoft's Xbox 360 trends well, selling 9.15 million units in its life-to-date, with the Wii catching up fast at 7.38 million life-to-date. The PlayStation 2 was a strong fourth place seller, with 3.97 million units sold in 2007, again beating its own successor the PlayStation 3, which moved 2.56 million units over the past year. Over its life-to-date the PlayStation 2 has sold 41.12 million units.

Halo 3 was the bestselling game of 2007, selling 4.82 million copies since its release in September. Most of the year's top games were released during the fourth quarter, showing the industry's continued reliance on Christmas spending to drive sales. Nintendo's games have a longer tail, and the company was rewarded with strong sales for Wii Play and Pokemon Diamond, which were released in Q1 2007, and Mario Party 8, which came to shelves in May of 2007.

— Jeffrey Fleming

JESSE HARLIN NOMINATED FOR G.A.N.G. AWARD

JESSE HARLIN'S FEBRUARY, 2007 column "Does it Sound Next-Gen?" has been nominated for a Game Audio Network Guild (G.A.N.G.) award in the Best Game Audio Article, Publication, or Broadcast category. Winners for the G.A.N.G. Awards will be revealed on February 21, and the editors wish him the best of luck.

The winners will be honored at the 6th Annual G.A.N.G. Awards ceremony, taking place during the Game Developers Conference. The awards event will be held at the Moscone Center and includes live video game music performances throughout the evening. For more information visit www.audiogang.org.

—Staff

THQ DROPS FRANCHISES, CLOSES CONCRETE GAMES

THQ RECENTLY ANNOUNCED IT WILL NOT BE CONTINUING ITS JUICED AND Stuntman franchises, and has dropped PS3 and PS2 ports of some multiplatform releases, and is shuttering its Concrete Games studio, all in one fell swoop. The move is designed to cut the less successful elements of its organization, as part of the company's 2008 financial guidance for stockholders.

For THQ's third quarter, sales projections were increased from $490 million to $509 million, but representatives have said the company will report "approximately $20 million in accelerated amortization expense" due to the under performing titles.

THQ CEO Brian Farrell has previously called out Juiced and Stuntman's performance by name, noting in October of last year that neither achieved desired play mechanic and overall quality targets, something that was "particularly disappointing" following the performance of games like Saints Row. "This is not acceptable," said Farrell at the time, "and we've taken steps to address it."

Along with the cancellation of the two franchises, THQ says it has also canceled the PS3 version of Frontlines: Fuel of War and the PS2 version of Destroy All Humans!: Big Willy Unleashed, now a Wii exclusive, as well as two unannounced titles for Xbox 360 and PS3 that were expected to be released in fiscal 2010. Most surprising was the closure of its internal Concrete Games studio, which was working on one of the unannounced games, but had yet to release a full title under the Concrete banner.

The publisher says a number of Concrete employees will be offered positions in other internal studios, and that the company at large will see financial repercussions of the closure stated on its fourth quarter earnings. Though THQ says it saw better-than-expected sales of WWE Smackdown vs. Raw 2008 and MX vs. ATV Untamed, it notes that this was offset by underperformance of a number of its titles, including the recently released Stuntman: Ignition, Ratatouille, and Conan.

Said Farrell in a statement, "In October, we announced certain product quality initiatives, including personnel and structural changes in product development and a more rigorous internal and external product evaluation and feedback process. Consistent with these initiatives, we have taken actions to strengthen our pipeline and position ourselves to compete aggressively with compelling, high quality games."

An anonymous former employee of Concrete Games had this to say: "Given the sad situation of having your studio shut down, I have to say that THQ has shown great class in the way they handled this situation. THQ has taken measures to reassign all employees affected into other THQ studios. However for those for whom a place could not be found, THQ HR and Recruitment staff are working hard to help place these employees into external studios.

The layoff is effective on Feb 6th and all employees that could not find a position in a THQ studio by that date will get 2 weeks for every year they worked for THQ as well as any vacation time. All in all, I think this was pretty fair given the circumstances."

—Brandon Boyer, Brandon Sheffield

★ INDIE GAMES

WEBLOG UNVEILED

IN JANUARY THE CMP GAME Group (publisher of *Game Developer*) unveiled a news component to IndieGames.com, part of the group's portfolio of web sites focused on submarkets of the video game industry. The site (www.indiegames.com/blog), edited by Wee Tim Boon, looks at the world of independent games with daily coverage of Flash, downloadable freeware, and independent commercial games. Boon and his co-editors are also fleshing out the site with interviews, features, and commentary on the emerging indie scene. "I continue to be surprised by the creativity of developers who produce games on practically no budget which are unlike anything seen in the market today," Boon said.

IndieGames.com—The Weblog has its origins in Boon's Independent Gaming weblog, originally created two and a half years ago as an information clearing-house for independent games. "The problem is that the games are usually not noticed until months, sometimes years, after their initial release," said Boon. "Not many will know about them unless they happen to have a developer's web site bookmarked, or chance upon a game announcement in a forum. I started the site as a place to inform people about the latest projects from these talented developers."

In the years since his old weblog's debut, Boon has seen the barriers to game development crumble as new technology opens the field to anyone with a desire to create. "The tools used for developing games are now easily accessible to anyone with an interest in creating something to share. [Some] game development software has reached a stage where it is nearly as simple to use as MS Paint," Boon said. "A budding designer with no coding skills can follow a few tutorials available on the net to get started, then upload their work to any of the game distribution channels online. There are a lot of creative developers turning out fantastic games and we will see independently developed games gain even more attention from the games industry in the coming years."

-Jeffrey Fleming

CALENDAR

P.I.E.C.E.
Cashman Center
850 North Las Vegas Blvd
Las Vegas
Nevada
March 8–9, 2008
Price: $50
http://piece.eventbrite.com

Indie MMO Game Developers Conference
Minneapolis Convention Center
1301 Second Avenue South
Minneapolis
MN
March 29–30, 2008
Price: $219.99
www.IMGDC.com

ION Game Conference
Seattle Marriott Waterfront Hotel
2100 Alaskan Way
Seattle
WA
May 13–15, 2008
Price: see website
www.ionconference.com

CoGames 2008: The 1st International Workshop on Collaborative Games
The Hyatt Regency Irvine
Irvine, CA
May 19–23, 2008
Price: see website
http://cisedu.us/cis/cts/08/workshops/W3-CoGames2008-cfp.html

Game Connect: Asia Pacific 2008
Brisbane
Queensland
Australia
November, 2008
Price: see website
www.gameconnectap.com
www.gdaa.com.au

>> dan boutros

DIFFICULTY IS DIFFICULT

Designing For Hard Modes In Games

DAN BOUTROS *is a consultant and freelance game designer, currently directing an unannounced DS puzzle game. Email him at* ***dboutros@gdmag.com****.*

DIFFICULT MODES IN GAMES ARE RARELY DISCUSSED AS AN important factor in our business. In some games, they are well-thought out additions, built for the hardcore players. In other games, these modes are an afterthought, provided to appease a publisher, or as an attempt to provide direction to multiple audiences attracted by the same product. For almost all developers though, difficulty modes are tackled at the end of the project when the game is being tuned, and they are tough to implement well without significant time and thought.

For this piece, I'll aim to explore some methods and philosophies behind how difficult play has been successfully implemented in games overall, either in terms of general difficulty, or within an optional objective that recontextualises play (such as Rare's super-tough multiplayer bonus unlockable objective in GoldenEye for the Nintendo 64).

We'll explore where certain methods have worked, failed, and where they are simply not relevant anymore. Since difficulty is so subjective, I'll focus arguments around the following ideas:

> "A player must always feel like the failure of a challenge is entirely his own responsibility, and not a fault of a poorly designed product."

> "The player must understand how and why he failed, so that he can learn from his mistake and increase the feeling of failure being his responsibility."

Choosing a high difficulty is the act of wanting to be tested on the part of the player. The reward of passing a test is a feeling of worth and accomplishment—and to make a test enjoyable is to make it challenging, while also achievable. Tuning difficulty in a quick and dirty way can also change the game's play fundamentally—this is something many developers don't factor into their decisions enough.

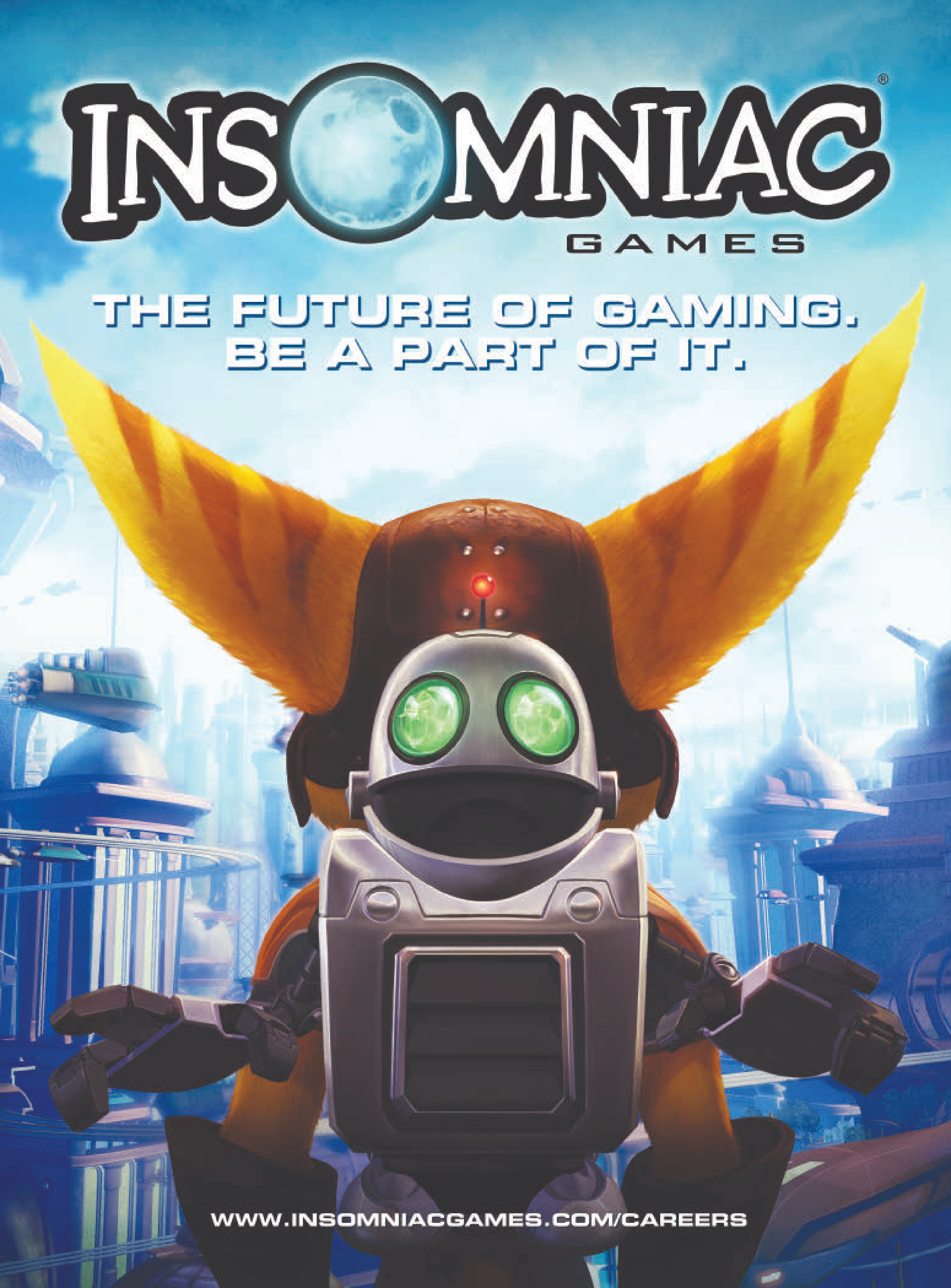
TUNING FOR TOUGH

Many games have sought to copy Rare's N64 GoldenEye model for greater difficulty—double damage from enemies = harder game—but within the context of other factors, doing this can actually change the consistent play type of the game, and thus change the experience in a fundamental and arguably unsatisfying way.

Let's say there's a fictitious FPS called *NaziShoot 2000*. In this game's normal difficulty mode, the player can usually get shot, have a second to think, recover, then react. In tough mode, players cannot risk being shot as the increased damage and AI kills them almost instantly. This forces players to move and act more conservatively. In an ideal-world's well-designed tough game, it would be possible to play through and not die, if the player used the utmost care and thoughtfulness. However, this game had to hit a deadline so the tough mode had to be evolved from the normal mode, and tuned to a formula. In this memorizing bottleneck scenario, surprise snipes to the head, and learning from trial and error become the dominant way to play.

And there is the difference: whereas one mode is a reactionary and lightly memory-reliant experience, in the tough mode, the game becomes very classically rooted in trial and error, using memory play as the core consistent play type. The only way a player can survive with meager resources and a damage disadvantage is by trying, dying, remembering, and restarting.

This is a classic tenet of the old school 2D arcade shooters. In a 3D game where an additional axis dramatically adds to your


INSOMNIAC®
GAMES
THE FUTURE OF GAMING.
BE A PART OF IT.
WWW.INSOMNIACGAMES.COM/CAREERS

things-to-worry-about radar, control complexity is usually increased, and gameplay acts—core gameplay sequences such a shooting something and then grabbing a power-up—are spread across a longer timeline because of the physical world's scale increase. If the player can be killed in one hit, or by other fatal game features, this can often result in an intense feeling of frustration, and quite possibly lead to dissatisfaction with the game overall.

DIFFICULTY TUNING ROUNDUP

There are many elements of gameplay that are worth tweaking for harder difficulty modes, provided there is time to test each of these thoroughly against the other modes, and with players of varying skills. I'll investigate a few here.

Time limit. This is a classic difficulty tuner for racing games, and a few platform games—essentially inspiring the player to get to the end of a level or challenge by X time. This usually forces the player into optimal route-finding play and looking at their navigational tools differently. Had Prince of Persia: Sands of Time embraced non-linearity, we may have had a time limit goal challenge to recontextualize the levels in some very interesting ways. Generally, most racers have a medal-based reward structure where the gold medal will be a very tight time to achieve. Games that have other play types involved in the racing—F-Zero has combat for example—end up forcing the player to also optimize those play types, alongside finding the ideal path.

Damage Dynamics. This is the easiest thing to tweak when making a game harder—just make everything hurt more. The problem is that without careful tuning of other factors, such as enemy placement and resources, this can bring about one-hit kill syndrome, which in a 5–10 minute stretch of play can come across lazy and unfair. It's also the easiest way to change the feel of the game from an evaluate/solve game to a predominantly memory-play based affair.

The enemy AI in Halo shows increased aggression at higher difficulty settings.

The best difficult modes in games tend to share the same formula for upping the difficulty, which is contextual to the level portion. Sometimes this involves more enemy spawns, and most of the time there's additional damage—and in some places, less ammo.

Simon says. If the game is dependent on button presses, like DDR, Beatmania, or Guitar Hero, then demanding more presses per time unit makes sense as a difficulty increase. Pressure is increased through the players having to parse what they see, and combine that with what their currently engaged feet, fingers, or other digits are already doing.

Increased AI aggression. Halo 1 and 3 are perfect examples of how to pull this off, as are the Infinity Ward installments of the Call of Duty series. Sometimes, the enemy just behaves more aggressively, forcing the player to act more quickly and confidently, rather than wait around trying to aim a perfect head shot. In Halo, the grunts will do suicide kills in the harder difficulties.

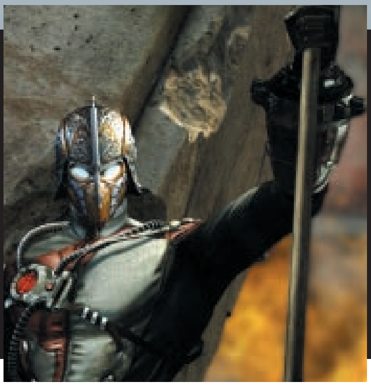
Make your enemy AI take more risks, but on the flipside don't make the enemy AI be able to see the player from a million miles away with the aim of a laser-eye-surgeried hawk, or you'll get a Mega Man-style scenario in which players feel cheated through an unknown and unseen kill, such as from an off-screen bullet. In some contexts it's realistic, but in most games of this sort that I've played, it's rarely fun. It could just be me, but it made me stop playing Medal of Honor: Airborne.

Reduced neutral zones. This is when safe areas and unused areas are removed or replaced with hazards, such as spikes you can't stand on or instant-kill zones. The Challenge levels in Valve's Portal are a great example. This has the added side-affect of pin-holing the player's focus o n very few options, and with good level design can create a high-pressure puzzling environment.

HUD restrictions. In some games, challenge can be tuned by altering the interface. Games with multiple feedback systems work best, as numbing or reducing just one of your in-game senses can cause player decision-making to be more tense. The Metal Gear Solid series has done this since the beginning, where harder difficulties would remove the in-game radar, meaning the player had to rely not only on their memory of the

Maximum Speed.

Zero Drag.



LightWave 3D®

"With LightWave's insane speed across the board, not to mention its flexibility and ease of use, it has no equal. For WarDevil, it has become the core of the project and continues to provide us with solutions where other 3D applications give us dead ends."
— Kelly Myers, VFX Supervisor, *The WarDevil Project,* Digi-Guys, Ltd

LightWave® v9 Get it done.

Visit **www.lightwave3d.com/getitdone** to download a white paper on how LightWave gets it done for games.

NewTek
www.newtek.com

game's layout but also their reflexes and knowledge of the AI behaviors to survive.

Resource stinginess. The Resident Evil series chose to limit ammo as difficulty levels ramped up. This results in extreme tension, helping to add to the horror, but also greater pressure on the player to be resourceful. Many skillful Professional Mode Resident Evil 4 players rely on sharp-shooting kneecaps to bring the enemy to its knees, then finish it off with the knife to get through the game on a minimal bullet-budget.

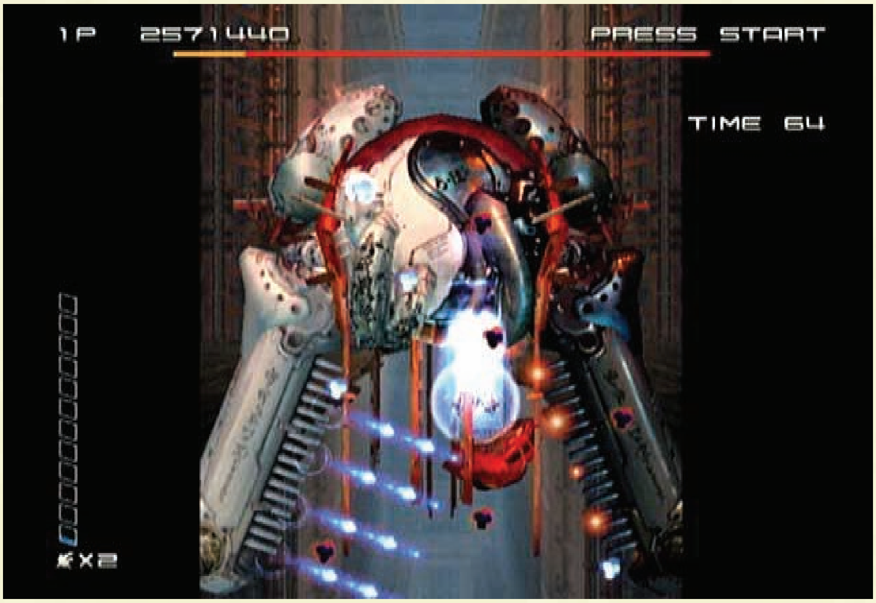
THE ILLUSION OF FAIRNESS

It's a common perception that the key to keeping a challenge fun for many is to make it feel fair to the player. Of course, classic video game scenarios—such as one man against an army of aliens, or a flying saucer against numerous battalions of space ships—is completely unfair.

Few games, if any, are "fair." The illusion of fairness to the player is what's important, and this comes in the form of convenience mechanics. This includes things like health packs placed before and after sections with large enemy counts, generously placed ammo pick-ups, strength/weakness matrices for certain weapons a player may have versus present enemies onscreen, and smart checkpointing that checks in regularly, usually before and after these kinds of areas. Above all this, communication and feedback to the player is necessary. Here are some examples.

Physical feedback. This is the staple of many FPSes, but Resistance: Fall of Man lacked it (the game has since been patched for Sony's new rumble controller), and in my opinion really suffered for it. Aside from obvious elements like weapons firing and vehicles moving, most games nowadays tend to use feedback in the controller to communicate how close you are to death. For example, if the controller is at a light stage of rumble, the player is being attacked, but is not about to die, or the enemy is attacking from afar with weak munitions. If the rumble is intense, chances are that the enemy is attacking at close range, or with an attack that is bringing the player nearer to death.

Treasure's Ikaruga challenges players to navigate dense curtains of fire while maintaining awareness of their ship's polarity.



Hazard feedback. Flashing red markers onscreen and other screen filter effects are great for games where your avatar is in 3D space and can be attacked from all angles. Half-Life pioneered this with 4-way red flashes to indicate if damage was greeting you from the sides, behind or in front, but later games like Call of Duty enhanced it by specifying exactly which angle.

Loss feedback. This is somewhat rare in games. Usually, this feature is considered taken care of if the game is tightly designed. Some games, such as the critically acclaimed racing game Burnout, use an instant replay function. Team Fortress 2 also does a great job of telling players how they died. Upon being killed, a player's assailant is displayed onscreen immediately crushing any illusions that the player was robbed of a victory by a poorly built game—an illusion suffered by many sore losers (myself included).

SAVING AND CHECKPOINTS

Checkpointing can polarize players' opinions, when points are spread further apart as a measure of difficulty. Some feel that it's more "hardcore" to play through a large number of acts, for a longer period of time than normal, and eventually succeed. Others feel that save and checkpointing should remain a convenience mechanic and be offered generously, with an "every 5–10 minutes" philosophy at the core.

The truth is, wider checkpointing tests patience and memory as much as it does skill, which can frustrate and put off players. Call of Duty 2 had the best checkpointing implementation I experienced in the new generation, which was expanded even further with Call of Duty 4. Not only was each intensely exhilarating section in veteran mode beatable within 5–10 minutes (in Call of Duty 2, specifically), players would also get save-able checkpoints which appear before and after a cut-scene, saving players from classic tedium moments such as the "Look at all dat juice" scene from Gears of War.

This infamous portion of Gears of War placed players against a group of the tough Theron Guard enemies as they laid siege to a pumping station. Dying in battle restarted the player a long walk back before the action began, replaying a verbal scene in which one of the characters utters the immortal line, "Look at all dat juice." Repeating this became annoying after a while.

Optional quests are quite a common design choice when providing for hardcore players, as are special missions, or optional pick-ups in the environment. Younger-skewed games, like Jak and Daxter, Ratchet & Clank, and Lego Star Wars tend to reward players with an auto-save when finding optional pick-ups. Gears of War also does this with its COG Tag bonus system.

The great thing about doing this is that players are still rewarded for the task they performed, but if they die, they don't need to do it again—they only need focus on the challenge of core play.

Some games, including many PC FPS, rely on players to use the game's save-anywhere feature, and skimp on checkpoints. Others choose to space checkpoints far apart, also forcing players to redo all the sub-tasks in between, meaning maximum frustration for completists who wish to challenge their skills more than their patience. Black by EA Criterion is one of my most remembered offenders in this respect.

The STREET FIGHTER series popularized complex button sequences as a risk/reward relationship in fighting games.

In that game, upon walking down a hill, shooting across a field, entering a compound, killing everyone and destroying the place, only to be shot and killed on my way out, I would have to not only restart the whole sequence (20 minutes of play), but also collect all the optional objective items again, costing me that same large chunk of time each and every try. Eventually I found an optimal route and a superhuman level of patience, but until that point, I utterly hated the experience for one section of what is otherwise a very well-crafted game.

The common rationalization designers give when confronted with these criticisms is usually, "the player doesn't have to do all that. It's their fault because we made that optional," but what they forget is that the players most likely to undertake harder skill modes are precisely the ones most likely to force themselves to get all these things and perform all these acts. In essence, the designer is trying to justify these choices with semantics, when what they're really doing is forgetting the truth about their audience. It's a key part of understanding the whole point behind these modes.

ENEMY COUNT

This can also be regarded as bullet count in top-down shooters, or basically any increase in kinetic or AI-driven hazards. Obviously, the more of any of these elements a player has to deal with, the quicker they have to think, and the greater the sense of pressure and stress, thus the more satisfying it feels when success is earned. But balancing the pressure with actual achievability is quite difficult.

IKARUGA has arguably done this well. IKARUGA is a top-down 2D shooter with a simple mechanic that also made it equal parts puzzle game. The ship has black and white sides, switchable with a button press. White can absorb white bullets and do double damage to black ships. Black can absorb black bullets and do double damage to white ships.

In easy mode, when a player shoots an enemy, it dies and that's it. In normal mode, enemies you kill with the same color bullet as your ship explode in a hail of bullets. In hard mode, all enemies explode in a hail of bullets. At each stage of difficulty, the puzzle element is changed slightly so that players have to alter their memorized plans of action and possibly adapt their reflexes to be a little quicker. This also adds to the overall stress aspect.

GRADIUS, R-TYPE, and other scrolling shooters have a memory-play aspect in terms of knowing the level's shapes and movements, coupled with testing of reflexes at great speeds. This is the simplest level of difficulty adjustment, as mentioned

gamedeveloper FRONTLINE FINALIST '05

"ReplayDIRECTOR is one of those tools that you really recognize the value of when you're in crunch.... we were able to get turn-around times down from days, to minutes."

John Chowanec, Lead Producer, Eidos

FREEZE.

RECORD. REPLAY. FIXED.

ReplayDIRECTOR™ gives you **Deep Recording.** This is much **more than just video capture**. Replay records every line of code that you execute and makes certain that it will Replay with the same path of execution through your code. Every time. Instantly Replay any bug you can find. Seriously.

DEEP RECORDING. NO SOURCE MODS.

download today at www.replaysolutions.com
email us at info@replaysolutions.com

REPLAY SOLUTIONS
1065 E. Hillsdale Blvd. Suite 307 Foster City, CA, 94404 - Tel: 650-472-2208 Fax: 650-472-2210

CRYSTAL DYNAMICS eidos XBOX 360 nVIDIA. XBOX at the start of the article. The odd additional turrent or enemy in a PC FPS can yield somewhat similar results, while tweaking the player's existing memory map of a level.

Scrolling shooters tend to be perfect for this feature, as their levels are small and the bullets can behave somewhat organically within a fixed environment, blending memory play with reflex play, though players do tend to share optimal solutions for playing through these games.

INPUT COMPLEXITY

Street Fighter was one of the first games to create a risk-reward relationship between input complexity and onscreen actions. For some time, Zangief's spinning piledriver was the holy grail of hand-eye coordination, requiring a full circular movement of the joystick followed by a punch in close range, which would reward the player by depleting almost a third of the opponent's energy bar.

The risk was getting in close to the opponent, coupled with the fast spin of the stick, which if done too slowly or out of range, would cause Zangief to jump and leave him vulnerable. In comparison, Ryu and Ken's fireball motions were simple, requiring only a quarter circle motion and press of the punch at any distance. The by-product of using input complexity as a difficulty barrier is that it doubled additionally as an entry barrier for new players, meaning an instantly limited audience, though culturally, its relevance changed gaming for the next ten years.

Interestingly, David Sirlin, lead designer of Super Street Fighter 2 Turbo HD Remix has decided to change some of the move inputs from the original game to be more accessible—and this includes Zangief's spinning piledriver. Sirlin believes the difficulty should come from the think-on-your-feet strategy aspect, rather than the player's ability to swirl inputs.

"I think [Super Street Fighter 2 Turbo HD Remix] is much improved with the easier commands," says Sirlin. "Instead of T.Hawk and Fei Long being practically non-characters [as they were in the past], they are fairly easy to start playing and much more fun. The strategy in Street Fighter is much better—not much worse—when both players can execute the moves. It's not like being good at the game has gotten much easier just because special move commands are more forgiving. It's more like being able to play it at all in the first place is easier.

When the best players in the United States play the in-development version of this game, they say that it's more strategically interesting than before. The new commands for moves are only a very small part of that. But what's more important than any of that are the actual balance changes that allow the previously weak characters to compete more fairly with everyone else. There are several match-ups that were simply boring because one character had a very dominant pattern to rely on. Most of these have been greatly toned down so that new mind-games take the place of old, robotic patterns.

New Street Fighter players will, I hope, discover a much more interesting path to improvement by focusing on these strategic match-ups, rather than simple move execution."

LEGENDS OF DIFFICULTY

Bungie's Halo series is often praised for its excellent execution of difficult play in the form of its Legendary mode. Not surprisingly, the team took a very well-thought out approach to introducing and tuning difficult play. Halo 3 gameplay designer Francois Boucher-Genesse explains that it's not just a case of one formula fits all.

"It's not like we just cranked every enemy's health by 200% and called it Legendary," he said. "There was a good amount of custom changes made per mission as well. In that sense we encourage players with previous Halo experience to play at least on Heroic, since they get to see the game in its full scale."

Street Fighter 2 Turbo HD remix has altered control input for ease of play.

"A really similar formula was used for every Halo game," continued Boucher-Genesse. "What did make a difference was the time spent tweaking and fixing issues to make the game fun on every difficulty level. All titles had more bad guys, stronger and more accurate enemies with faster projectiles. And they used similar numbers for each of these parameters."

When your company budgets for tuning of difficulty levels, the result is extended longevity for the game, and increased enjoyment for players at all levels. It goes without saying that games should be appropriate for their target players on all levels, providing challenges for each skillset. One important thing to remember, which has been creeping into games recently, is that players aren't often able to determine their own difficulty levels. They may think their skill level is far above or far below the actuality.

In Metal Gear Solid 3, players are asked if they've played a previous iteration of the series, and are assigned a difficulty level based on this. This is a pretty good way to go, and Call of Duty 4 takes it further. Players go through a training course in the beginning of the game, which also gauges their ability, recommending a difficulty level at the end. Most players will take this recommendation when it's directly from the game universe, rather than making a choice on a list before ever having touched the game. With all the elements in place, the life of a game's single-player campaign is extended enormously.

>> adi bar-lev

BIG WAVES

CREATING SWELLS, BREAKS, AND EVERYTHING IN-BETWEEN

ADI BAR-LEV
has been a senior 3D programmer at Ubisoft for the past two years, developing the water effects and wave technology for the game SURF'S UP. He is currently working on a yet to-be-announced game developing mass planting technology, and cutting-edge 3D. Before joining Ubisoft he worked as a technical lead at Samsung developing game engine technology for mobile devices, and at Simbionix as a team lead developing physical surgery simulations. Email him at ***abarlev@gdmag.com****.*

IN THE PAST 10 YEARS THE USE OF WATER IN GAMES HAS become more and more noticeable. Until not long ago, the technology to create real time convincing-looking water was not feasible, so this kind of high quality water was used mainly in movies where it was generated offline. This was due to the large amount of polygons needed for representing the water surface, as well as the rendering complexity of the shaders. These days, with the vast increase of GPUs' computational power and programming flexibility, most of the required technology is feasible for games as well.

As a result, a number of articles have now been published in this area, which describe optimal methods for rendering various types of water and effects for games (see Jensen, Kass, Tessendorf, and Neyret in the Resources). Unfortunately, none of them address the issue of large waves, although many surfing games do exist.

In this article I will describe the techniques we used in the game SURF'S UP to create the water model and surf waves. My main focus will be on two aspects—describing the main existing methods to create and render 'infinite' water models, and applying big surf waves to the water surface.

WATER MODEL CREATION METHODS

An important question to ask before implementing any CG water model is what you need it for. The answer can give you a rough idea of the best approach to the creation of a water model for your application. Since it is likely to affect the rendering algorithms, this step should be taken at an early stage.

First, I'll go over several of the main techniques for creating a water model. In this part of my article I'll focus more on the general ideas, rather than the actual implementation.

CONTINUED ON PG 16

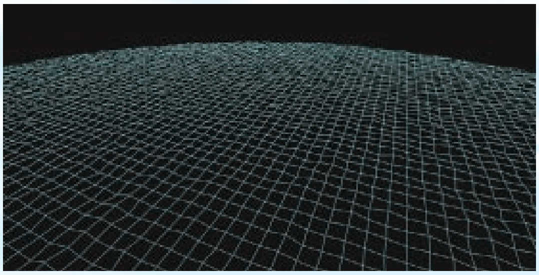


FIGURE 1 The uniform grid is shown.

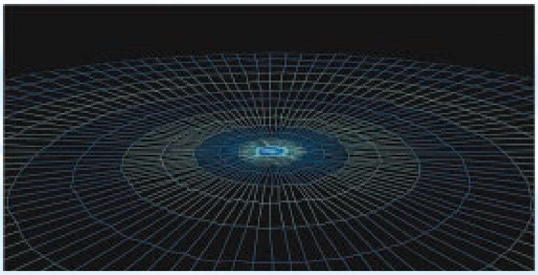


FIGURE 2 The radial grid is shown from a bird's eye view.

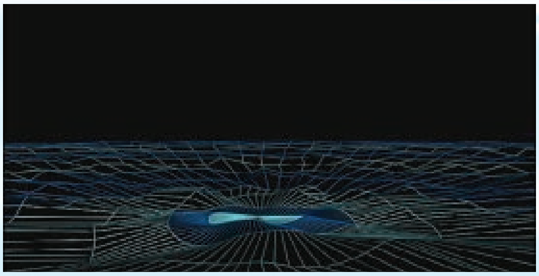


FIGURE 3 The radial grid is shown from a shallow angle view.

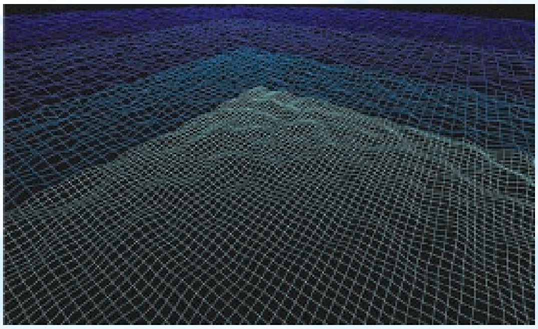
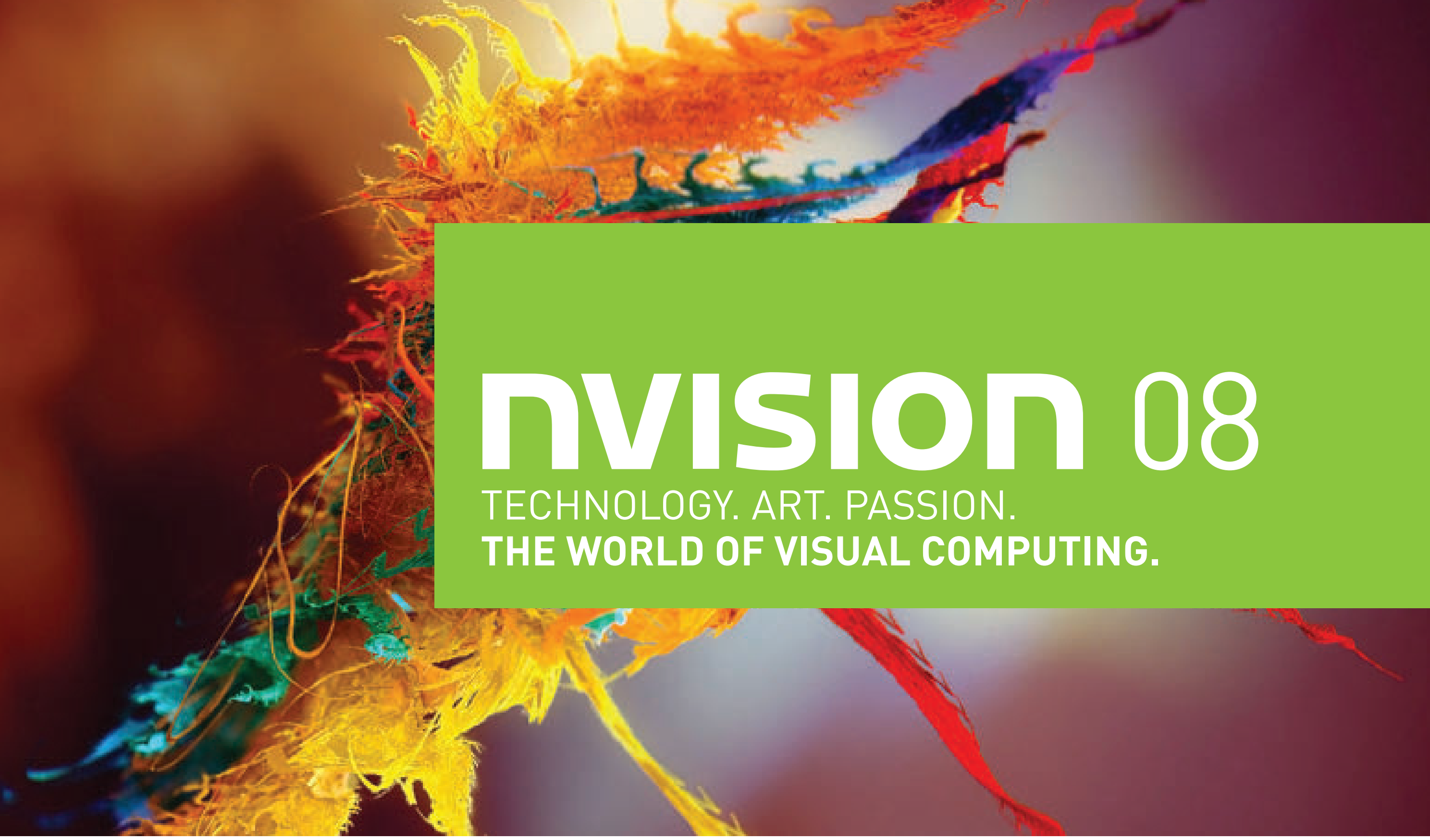
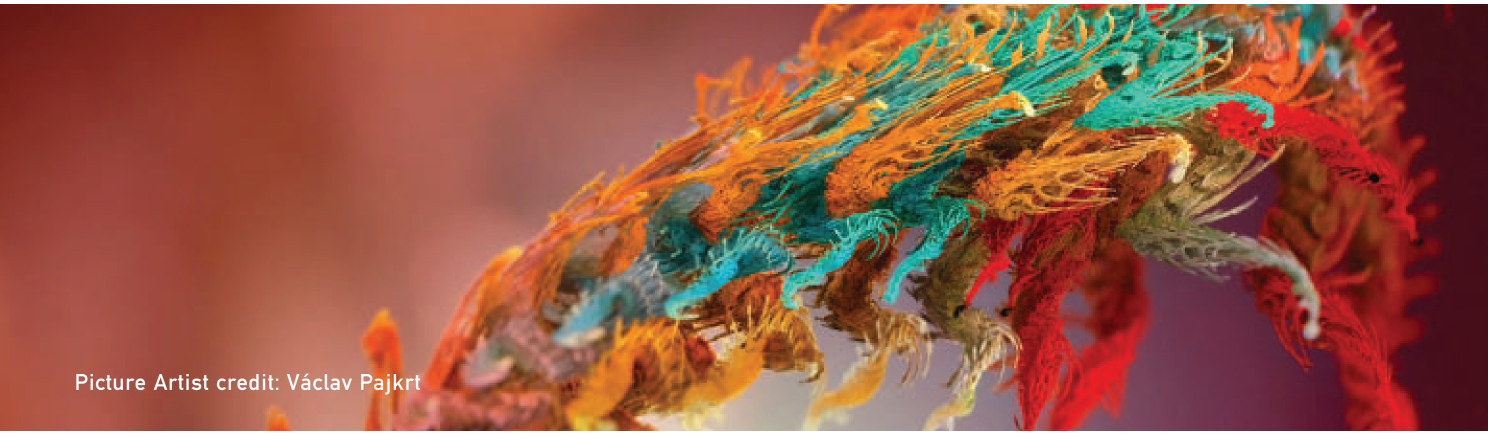


FIGURE 4 The LOD surface with four levels is shown.

PHOTOGRAPH BY LIVAH45@MORGUEFILE.COM





Picture Artist credit: Václav Pajkrt

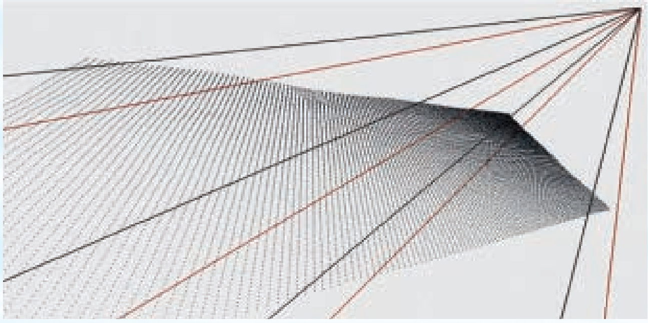


FIGURE 5 Projected grid: camera boundaries are shown.

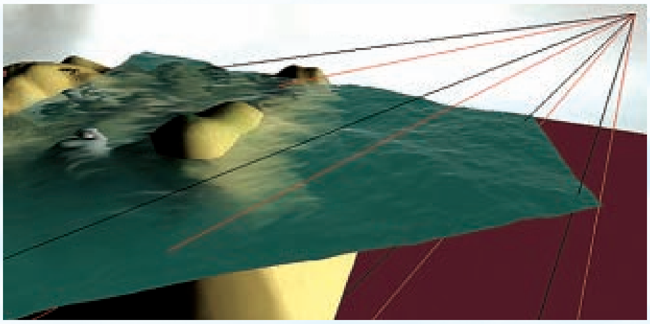


FIGURE 6 Projected grid: the render of the grid, and the grid division distribution is shown.

FIGURE 7 SURF'S UP character Chicken Joe riding a big wave is shown.

CONTINUED FROM PG 14

LOW POLYGON AND UNIFORM GRID MODELS

These types of models were the most common in games for some time, and were typically used for small lakes or puddles of water. One or two layers of normal maps, with some oscillation with added reflection and transparency results in what most (non-water sport) games used to have up until not long ago. The approach is still widely used for simulating small areas of water in games today due to its simplicity.

THE RADIAL GRID MODEL

Representing large water areas such as oceans necessitates displacing the model's vertices to achieve a good visual quality for the ripples. Any regular grid division methods will result in having a large amount of vertices that need to be displaced every frame. For example—a coarse 1 square meter uniform division over a radius of 2 km will generate over 12 million vertices. Covering only the active field of vision at that resolution will still yield over 3 million vertices!

A good and simple solution—which is also suited to be implemented on the GPU—is the Radial Grid [see Yuri Kryachko in Resources for more]. The model is combined of circular strips, cut by perpendicular lines with a defined angular resolution. The strips' radii follow a formula that dictates a radius increase as we move further away from the center of the grid. An example of such a grid is shown in figures 2 and 3.

In his article, Kryachko sampled the grid as follows:

$$r_i = a_0 + a_1 i^4 : i = [0..N-1]$$

$$X_{i,j = r\cos}\left({}^{2\pi j}/_{K}\right) : j = [0..K-1]$$

$$Y_{i,j = r\sin}\left({}^{2\pi j}/_{K}\right)$$

$$a_0 = a_1 = 0.1\text{m} ; r_n = 40\text{Km} \longrightarrow n \cong 25$$

Here, K is the radial resolution and a1 and a0 are the initial distance from the camera and the distance multiplier, respectively. Note that while the initial length resolution is high, the grid covers a huge distance yet maintains a reasonable amount of vertices.

When using this method for shallow camera angles (close to the water) we get almost a fixed triangle's area in screen space. However, if the water has high perturbations, the medium range distance resolution is not high enough, hence a lower exponent (3 is a good choice) or a different distance division formula might be more desirable.

I suggest here a scheme which yields a better density distribution over the close-to-medium range (up to several hundred meters) while maintaining a far distance of about 12 km with the same amount of vertices:

$$r_i = a_0 + a_1 \cdot (\text{Base}^i - 1) : i = [0..N-1]$$

$$a_0 = 0.1\text{m} ; a_1 = 0.5 ; \text{Base} = 1.5 ; r_n = 12.6\text{Km} \longrightarrow n = 25$$

To maintain the highest tessellation of the grid near the camera (and the grid as an infinite surface), the grid center should always follow the horizontal location of the camera (adaptive location).

This grid movement can cause a noticeable artifact over many of the adaptive location methods. When shifting the grid over a geometric function, the eye detects the movement because the division of the grid is finite. If the surface polygons are larger than a few screen space pixels, the vertices' change in locations over the geometric function is apparent and non-continuous. I call this the "blanket effect." Usually the solution when using this method is to refine the division since it can be solved in a different way for other methods.

THE LOD SURFACE DIVISION

Much like the previous method, the aim here is to have a good resolution at the close and middle range from the viewer while covering an overall large distance.

The surface is composed of multiple Levels Of Detail (LOD), where each LOD is a closed rectangular strip of polygons

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 has won Game Developer Magazine's Front Line Award for Best Game Engine for the past three years, and Epic was awarded Best Studio at the 2006 Spike TV Video Game Awards. Epic's "Gears of War" won GameSpot's overall Game of the Year and sold over 4,000,000 units on Xbox 360. Epic recently shipped the PC version of "Gears of War" for publisher Microsoft Game Studios, as well as "Unreal Tournament 3" for PC and PlayStation 3 for publisher Midway.

Upcoming Epic Attended Events:

D.I.C.E. Summit
Las Vegas, NV
February 6-8, 2008

GDC 2008
San Francisco, CA
February 18-22, 2008

Please email:
mrein@epicgames.com
for appointments.

RED MILE PICKS UE3 FOR SIN CITY GAME

Red Mile Entertainment has licensed Unreal Engine 3 for the production of its upcoming *Sin City* game based on Frank Miller's popular graphic novels.

"The Unreal Engine is one of the most technically advanced engines on the market and is a perfect fit for our *Sin City* video games," said Glenn Wong, president and chief operating officer of Red Mile. "The engine has powered several of the best-selling games of all time, and I am very glad we have the opportunity to arm our developers with such a world-class product."

The design, scriptwriting and story generation of *Sin City* will be overseen by Flint Dille, who has previously won awards for stories he produced for *The Chronicles of Riddick: Escape from Butcher Bay* and *Dead to Rights*.

HKS TO REVOLUTIONIZE ARCHITECTURAL VISUALIZATION WITH UE3

HKS Architects, ranked among the nation's top three architectural firms, has signed a licensing agreement with Epic that will change the landscape of architectural design through its adoption of Unreal Engine 3.

Powerhouse architecture firm HKS is using Unreal Engine 3 for projects like the new $1 billion Dallas Cowboys stadium.

HKS ARCHengine, powered by UE3 technology, creates functional 3D visualizations of architectural space complete with moving figures, operational elevators and running escalators.

"This next-generation game technology incorporates real-time shadows, lighting and people into our architectural projects," said Pat Carmichael, manager, advanced technologies, HKS. "When designing a stadium, the client – meaning a life-like version of the client and their families – is able to walk into a stadium, order and eat a hot dog, and interact as they watch a baseball game."

"HKS is the first architectural firm to apply Unreal Engine 3 technology to its designs," said Ralph Hawkins, FAIA, FACHA, president and CEO of HKS. "We are pleased to offer our clients the sophisticated tools contained in this unique program, which allows us to powerfully portray our designs through its versatility and strength."

HKS's clients, which include the Dallas Cowboys and W Hotels developer Gatehouse Development, are taking them up on the new technology at a cost that ranges from $65,000 to $150,000 per project.

When Dallas Cowboys owner Jerry Jones was ready to select the site of his owner's suite, HKS ARCHengine allowed him to decide by offering a 3D look at each and every view and vista at the new $1 billion Dallas Cowboys stadium.

Before spending $1 million on a condo unit, potential buyers at the W Dallas Victory Hotel and Residences were given a full tour of the building – from the 30-story glass balcony views of the Ghostbar to the picturesque city views from each and every unit.

"HKS ARCHengine offers clients a higher level of graphic photorealism, communicating multiple design concepts rapidly and cost effectively," said Chris Roberts, senior developer, HKS. "It renders 30 frames per second instead of one per hour – that's 3,000 times faster than traditional animation methods. This makes it a first (and only) in the architectural industry."

UE3 GETS UNRIVALED GLOBAL ILLUMINATION AS ILLUMINATE LABS JOINS INTEGRATED PARTNERS PROGRAM

Illuminate Labs, creator of next-gen rendering and lighting technology, has joined Epic's prestigious Integrated Partners Program (IPP). Per the partnership, Illuminate Labs has incorporated its Beast global illumination system into Unreal Engine 3, and the lighting technology is immediately available to any publishers or developers that license UE3.

Beast integration in UE3 enables artists to seamlessly add global illumination to development environments using the renowned Unreal Editor. Beast is also a fully platform-independent baker that utilizes the open-standard FBX file format, which allows users to easily add extremely complex baking functionality to 3D asset creation tools. UE3 licensees can instantly start taking advantage of Beast's features that are new to the engine, which include global Illumination baking, skylights from HDR images and baked soft shadows with transparency.

For UE3 licensing inquiries email:
licensing@epicgames.com

For Epic job information visit:
www.epicgames.com/epic_jobs.html

WWW.EPICGAMES.COM

Image from Killzone 2 courtesy of Guerrilla

buddy.se

Test drive the most powerful lighting technology in the industry!

Illuminate Labs' lighting solutions, based on the proprietary LiquidLight® technology, are used in many of the most exciting game titles like Aliens, Killzone 2, Mass Effect and Mirror's Edge. The solutions combine state of the art global illumination with an efficient workflow to shorten production time and to reduce cost.

Illuminate Labs' clients include BioWare, Electronic Arts Digital Illusions Creative Entertainment (DICE), Gearbox, Guerrilla, Insomniac, Midway, Ninja Theory, Nintendo, Polyphony Digital, Sony Computer Entertainment Santa Monica and Valve Software. For additional information about Illuminate Labs and to test drive our products, visit www.illuminatelabs.com

Illuminate Labs at GDC

During GDC in San Francisco you will find Illuminate Labs in the Nordic Game booth (located at the main entrance). We are also hosting two sessions, one on Wednesday, February 20th, and one on Thursday, Febraury 21st. Check out the GDC conference guide for more information.

Turtle

Turtle™, for Maya by Autodesk, is the only product on the market that combines advanced rendering technology with sophisticated baking functionality in one integrated toolset.

beäst

INTEGRATED UNREAL ENGINE PARTNER

Beast™ is a platform-independent pipeline product that allows users to add global illumination and extremely powerful baking functionality to their internal tools. Beast is compatible with the Unreal Engine.

ILLUMINATE LABS®
LIGHTING THE NEXT GENERATION OF GAMES

surrounding the previous levels, and the first level is a rectangle (See Figure 4). Usually, the polygons' sizes double between two adjacent levels.

The generation of the described LODs can be approached in two ways; artist-created pre-designed models, and procedurally-generated batches, which together form the grid. Artist-created models are easy to implement, and can be gameplay-dependent according to desired dynamic areas of high resolution. The memory consumption is equal to the sum of all models' vertices, and there is a low amount of overall processed vertices due to the manual design.

With procedurally-generated batches, the grid has different LODs, each combined of many identical polygons batches. It's harder to implement and tune to gameplay needs, but very cheap in GPU memory and bandwidth (only a few batches are needed). Procedurally-generated models have a higher number of processed vertices than artist-created ones, but only a fraction when compared to the uniform grid.

To construct a water surface using the procedurally-generated grid, one needs to send to the GPU the scale and offset of each batch, and the geometric deformation function. In addition, a blend between tiles should be implemented in order to hide any discontinuity artifacts between two adjacent LODs (especially when the camera is moving). For a pre-designed static mesh, this solution is not feasible.

Hughes Hoppe (see Resources) presented an excellent article of the procedural approach and, although it was aimed toward real time generation of terrain, it was tested for water surface as well by my colleague Christian Desautels and was proven to be a viable solution.

In addition, the adaptive LOD approach can easily be adapted to a change in camera view from a shallow angle to bird's-eye view. Using the camera distance from the surface to scale grid size will maintain visual quality while reducing the amount of GPU overhead.

Using this method, the blanket effect is quite noticeable when the camera is moving. The solution is to use the fact that the vertices are distributed evenly at each level and doubled between adjacent levels. Imagine that the camera center is located within the largest cell in the grid. The idea is then to move the grid only when the camera crosses to the next cell, though we still need to compensate for the in-cell fraction movement of the camera. Doing that, we always shift vertices to locations that were already populated. The only exceptions are the areas between adjacent levels, but this can be compensated for in various ways. To sum it up:

```
Overall Shift = Number of Grid Cells to Shift * Cell
Size + Camera In-Cell Shift
```

The reason we should pick the largest cell size for this is to avoid shifts of coarser levels of vertices to unpopulated locations (the equivalent of a phase shift of the sample function).

PROJECTED GRID

One of the most desirable results of tessellating a large water surface is that the surface's projection on the screen will yield polygons of equal screen size with little dependency on the camera angle and location. In other words, we aim to achieve a close-to-fixed polygon size in screen space.

This is the idea behind the projected grid method. It is also fully adaptable to work on the GPU while remaining fairly easy to implement.

The naive projected grid implementation follows the next steps:

- Create a uniform planar grid
- Find the corners' screen-space intersection points with the water surface (before any displacement)
- Inverse the projection of the screen-space corner' points getting the model-space coordinates
- Map the uniform grid corners to the model-space corners
- Apply the water displacement's function, and render

A very good reference of this approach was written by Claes Johanson (see Resources) who also solved the different camera end cases. Figures 5 and 6 were generated using his demo application.

This approach assumes a close to flat perturbation of the water. As the perturbations get larger a "buffer height" can be applied so that the projected area covers the full extent of the water area on screen (after the ripple displacements are done). However, at extreme displacements (such as big surf waves), many more polygons outside the screen area are needed and their projection on the screen no longer reflects the desired fixed size we want. The first point can be addressed by using multiple patches and rendering only the visible ones, which we can find by doing a minimal displacement pass.

VISUAL CHEATS

As with many computer graphics techniques we need to add some fine tuning and small cheats in order to make things look more convincing. When we created the ocean water effects for our game, we encountered several problems that forced us to look for solutions.

A classic example was the usage of a sky dome to represent the far horizon. If the water plane does not stretch far enough (as might occur in the case of the LOD and radial grids) and we introduce large vertical movements for characters such as big

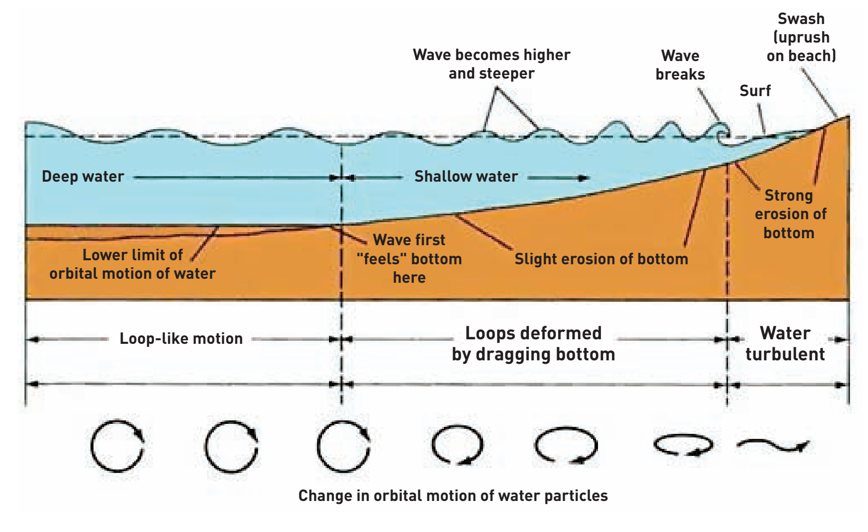

Swash (uprush on beach)
Wave becomes higher and steeper
Wave breaks
Surf
Deep water
Shallow water
Strong erosion of bottom
Wave first "feels" bottom here
Lower limit of orbital motion of water
Slight erosion of bottom
Loop-like motion
Loops deformed by dragging bottom
Water turbulent
Change in orbital motion of water particles

FIGURE 8 This wave diagram shows the effect of water depth on waves.

WAVE TERMS

Wave front: The curved line that defines the front of the wave when looking from above.

Wave's dimensions: Wave height, length and wave front length.

Cut: We divide the wave along its wave front, getting a local profile at each such 'cut'.

Wave lip: The tip of the wave where it starts to break.

jumps over a wave, it is easy to detect a change in the far horizon—the fake horizon height will seem to move differently!

Several solutions can be chosen here; changing the vertical location of the sky dome in relation to the camera movement, adding a slight bowl effect to the water plane at the rims, which imitates the real horizon effect, or updating the camera FOV before rendering the sky box so that it will compensate for the height movement—a method that was implemented by my colleague Lionel Berenuier. We tried all of these, and the last two gave the most visually pleasing results. While adding a bowl effect can be done very fast using the GPU, it is platform-dependent and so we used the last one.

Another nice effect implemented by my colleague Frederic Gaudet was to fade distant objects with both fog and alpha blend, hence overcoming any popping artifacts. As an added bonus we don't need to render a vast amount of entities as we move. One should take care to apply the same effects in the reflection pass as well.

In our game the waves traversed at high velocity, so we were able to use this fact when considering the visibility of objects in the scene. Although the scene is infinite, we can use the wave as a blocker and cut the scene along the wave's path into chunks that help define the visibility at run time.

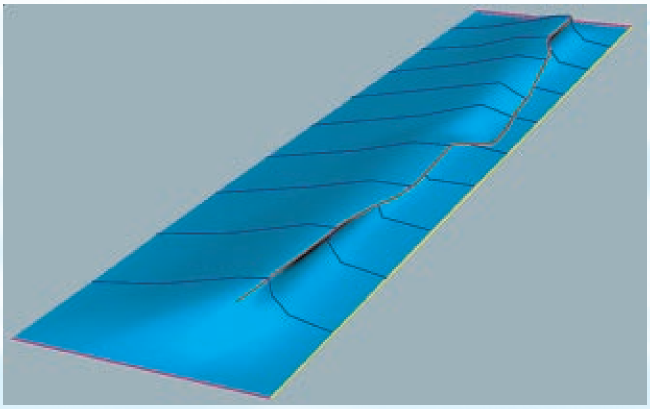


FIGURE 9 The wave surface: I highlighted the wave length in red, wave lip in orange, wave cuts in blue, and the wave front in yellow. Although displayed as a line in the image, the wave front is an animated curve.

THE FINAL CUT

Like in almost any field in life, the best way to decide which method to use is to have as many references as possible. If you have enough time, I would recommend implementing several techniques (without spending too much time on rendering effects) or getting demos that can help you test the properties you wish to have. The following are several points that can help you decide.

How perturbed is the water? Large noisy perturbations with big ripples necessitate higher density vertices and force your system to have vertex displacement. On the other hand, quiet lake water might not need any vertex movements at all, thus allowing you to choose a fixed lower resolution model. For relatively calm water you can cheat your way during the pixel shader (PS) render stage by constructing ripples mainly through a smart usage of normal maps and texture coordinate oscillation.

How big is the water surface that you need? For small pools you can simply use a fixed polygonal model and relatively low amount of polygons—this is by far the easiest to implement and can also be used as a benchmark for more advanced methods. However, if you deal with infinite water models (as seen in FARCRY or CRYSIS) you'd have to choose a different approach such as a projected grid, updating the grid location at each frame.

Screen-space vertices resolution. As mentioned before, in order to enable good visual quality when animating the vertices you should aim to have a close-to-uniform triangle's size in screen space, hence distributing vertices non-linearly. How exactly you do that is for you to decide.

Artifacts due to velocity and camera movement. Since both affect the visual quality grasped by the viewer, it is important to understand what exactly will be required. High velocity at shallow camera angles requires very high tessellation when using a projected grid, while bird's eye view movement can expose the shift in grid LODs if this method is chosen. In both cases there are ways to overcome the problem, but it is best to be aware of it in advance.

These are the main approaches in this field. Again, I only present the basics here, and for each method there are many variations and ways to solve the different end cases, and most can be found in the literature in Resources.

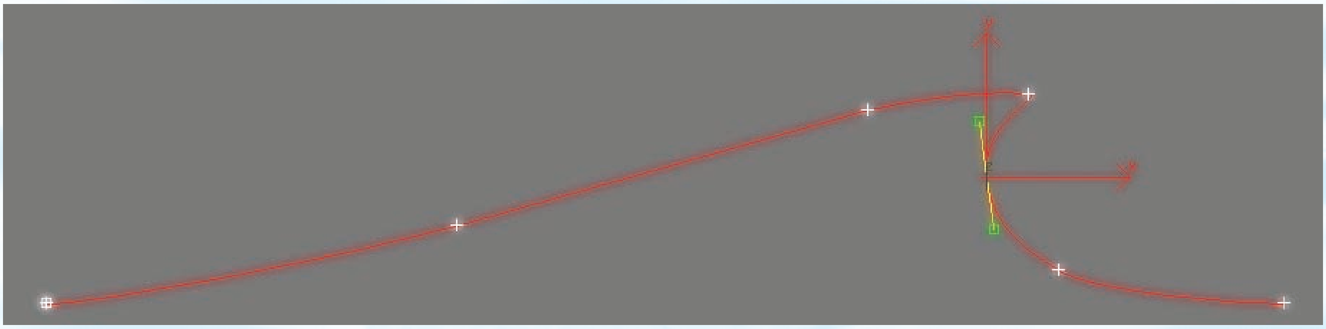


FIGURE 10 The wave cut profile: The cubic piecewise Bezier curve is in red, its vertices control points are in white and tangents of a selected vertex are marked in yellow and green.

BIG WAVES CREATION 101

For SURF'S UP, big waves meant 10 to 25 meters in height, 250 meters long, with the insane velocity of close to 100 km/h—these are almost Tsunami-sized. The player can do tricks on the wave for a full three minutes, so different techniques had to be developed just to control the wave.

What follows are some of these techniques, and the effects we implemented for the wave. Notice the size of the wave and the various effects—wave lip foam, particles, water trail and spray, different wave color, and discontinuity along some of the wave lip (Figure 7).

I began the journey by trying to understand how such waves are created in nature. To summarize in a simplified way—most waves are created as a result of a strong wind over the ocean surface. They are shaped as they travel by the wind, other waves, and the increased turbulence as a result of water viscosity when the depth changes near land.

The following equation describes the connection between a wave's velocity and water depth:

$c^2 = \frac{g}{k}\tanh(kd)$ where, $k = \frac{2\pi}{L}$, c is the wave's velocity, g is gravity, d is the water depth and L is the length of the wave.

Simplifying it, the equation becomes:

$c^2 = \frac{g}{k} = \frac{gL}{2\pi}$ while at deep water and $c^2 = gd$ when at a shallow water (when d<<0.5L), emphasizing how much the wave is slowed down as it approaches the shore, and the effect on its shape.

CHOOSING A SUITABLE METHOD

During my trip in the land of big waves, I implemented and tested three different methods—heuristic, physics-based, and traditional approaches.

The heuristic approach. Following articles such as the one written by Stefan Jeschke (see Resources) I tested several different equations which combined would generate a wave according to velocity and other provided parameters, and would generate and animate the wave progression. This approach, although providing good visual results, was difficult and unintuitive to control, and since we needed fine control over the wave, it was dismissed at an early stage.

The physics-based approach. Examining the physics involved, I tried to simplify the process by isolating several wave profiles and treating them separately. The profile of the wave was divided along the vertical axis into a finite number of uniform bands, on which I ran the physical simulation. Defining the wave's initial velocity, amplitude, wave length, water viscosity, and the shape of the sea floor allowed me to simulate the behavior based on the velocity equation of the wave's base.

Since our game required a continuous big wave, this approach was also put aside—adjusting the ocean floor as a means to control the shape of the wave was not really an intuitive approach.

The traditional approach. The need to control the wave for over close to 5 km forced me to give up and resort to animating control points on a piecewise surface—the most artist-friendly way. It also meant that any changes in gameplay that affect the wave's animation can be done almost without any need for additional programming.

CONTINUED ON PG 22

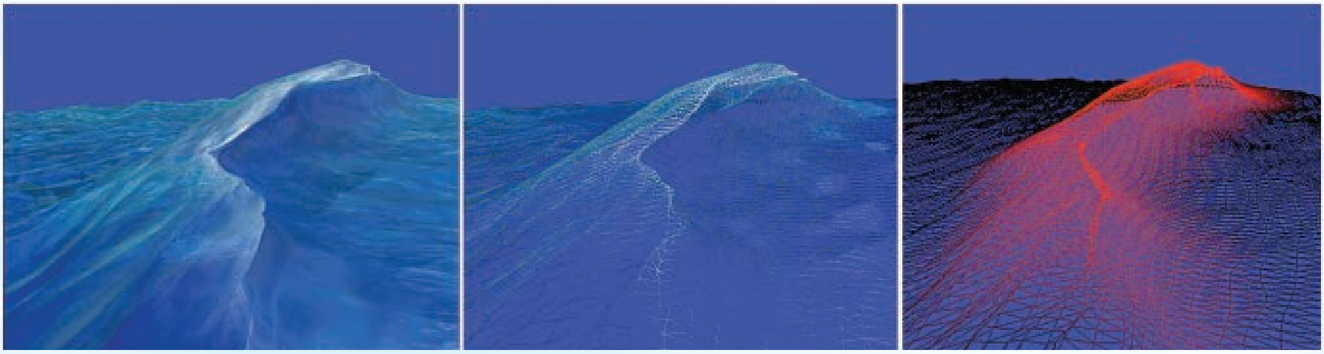


FIGURE 11 Ripples blend on the wave and render effects. On the right is the ripples' blend factor (in red). The ripples are gradually blended with the emerging wave for a more natural look. Notice how the noisy wave lip breaks the smoothness of that area, and the emerging foam texture that makes the foam particles blend better.

CONTINUED FROM PG 21

SURFACE REPRESENTATION

This section assumes a basic understanding of Bezier surfaces. See Alan Watt and Fabio Policarpo in Resources for more.

We constructed the wave as a piecewise cubic Bezier, with a grid of combined parallel cuts spread along the wave front (about 20 meters apart). Each cut contained 3N control points (N sets, each combined of incoming tangent point, vertex point, and outgoing tangent point as shown in Figure 10).

I will go very briefly over the equations. I start with a point along the curve:

$$P(u) = P_0(1-u)^3 + 3P_1u(1-u)^2 + 3P_2u^2(1-u) + P_3u^3$$

And the tangent:

$$T(u) = P(u)'_u = 3[Q_0(1-u)^2 + 2Q_1(1-u)u + Q_2u^2]$$
where $Q_i = P(u)_{i+1} - P(u)_i$

And when using the GPU (cheaper matrix notation):

$$P(u) = [u^3 u^2 u 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

The coefficients of each term are called the *Bernstein Polynomials* and they can be summarized:

$B_{i,n}(u) = \binom{n}{i} u^i(1-u)^{n-i}$ where n is the polynomial degree.

Since u and v are independent, each point and tangent along the surface can now be written as:

$$P(u,v) = \Sigma_{i=0}^{3}\Sigma_{j=0}^{3} P_{i,j}B_i(u)B_j(v)$$

OK, IT'S A SURFACE AND IT ANIMATES, BUT IS IT A WAVE?

In order to actually make it look like a wave, we need to add many features. The following are a number of them:

Foam. Unlike the foam generated by colliding ripples, wave lip foam behaves and looks differently. When the lip starts to break, it also generates water particles. While the particles were added on a different pass triggered by animation, the foam was added as extra texture which changed oscillation and transparency according to the wave lip.

Turbulence. In most games, the wave lip looks unnaturally smooth. To avoid that I applied a scaled ripples function at the thin lip area. Although very cheap, it gave the natural look we aimed for.

Inner Tunnel Color. This was another trick that allowed our artists to control the color of the front side of the wave where the tunnel

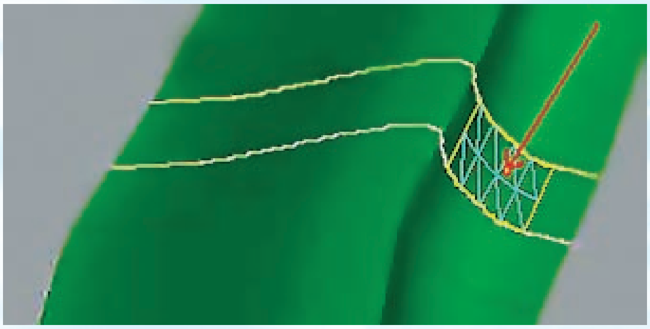


FIGURE 12 Collision detection: narrowing the search area according to the ray bounding box, tessellating the area, and finding the most relevant intersection point.

was forming. The idea is to apply the desired color according to the angle between the normal and a predefined direction:

```
Tunnel Color = Linear Interpolation (Wave Front Color,
Closed Tunnel Color, Tunnel Closure);
Tunnel Coefficient = Dot Product (Surface Normal,
Defined Tunnel Direction);
Vertex Color = Linear Interpolation (Water Color,
Tunnel Color, Tunnel Coefficient);
```

As the tunnel closes it turns darker while keeping the front side almost as light as the main water color, and nicely affected by the Fresnel equation.

Wave Transparency. When a big wave is passing, it generates turbulence that blends sand from the ocean floor. Therefore, starting from a defined point along each cut, our waves fade their reflection and opacity—this looks quite natural and helps deal with transparency sorting problems for a concave wave profile.

Wave Blend. Without a gradual blend, the ripples either cover the wave resulting in an unnatural look, or the smooth wave is grasped as if it is stitched to the water surface. We used a simple tuneable sigmoid function to determine the blend of transparency, reflection, ripples, and normal map on top of the wave.

OPTIMIZATIONS

The above scheme—while complete—can be highly optimized. Assuming wave animation of 10 cuts with 21 control points each, at 15 fps over 180 seconds the amount of stored data is too big (about 7 MB). We applied a number of useful optimizations. If you save only key frames, this reduces size to about 50 kb but takes much more time to compute at every frame (non linear interpolation). If you store only a single profile's animation, then a cut key frame is now actually a time reference to this animation. The cost was reduced to less than 10 kb with the above notion.

With data size issue solved, we could now bake the key frames for the reference animation (we baked 100 frames). At runtime we linearly interpolate between the baked key frames. The average wave animation data became around 30 kb per wave. If the wave animation doesn't change dramatically from cut to cut, we can assume the tangents between them to be horizontal, therefore diminishing the tangent equations as follows:

$$T(s) = P_0 s + P_3(1 - s) \text{ Where } s= (3 - 2u)u^2$$

If a control point can't move sideward, the side axis coordinate can be omitted and each vector passed to the GPU can store two control points. Calculating the cross product between the two partial derivatives, a point's normal can be represented as a closed form. For the PS2 implementation, both Edouard Poutot and Christian Desautels on our team assumed the tangents of a control point to be of same size and direction, hence omitting every third control point along a cut.

WAVE COLLISIONS

Having the wave generated as a piecewise Bezier surface helps when it comes to collision detection. Finding the mathematic intersection with a ray can have two different forms—for both, we optimized using the fact that all cuts are parallel and straight towards the forward axis. Once we found the relevant cuts, here are the steps for finding point of collisions.

Go over all relevant segments, and for each, either find nearest intersection point with the line (the analytical approach) or with coarse tessellation, find collisions (the geometric approach). If collisions exist, refine the result.

Then choose the nearest collision point. Blend with the collision point taken from the water plane according to location on the wave.

Because the wave is not convex, finding only a single collision segment might not be enough. Nevertheless, using the ray's bounding box we can isolate the exact cut location and test only a few of the profile's length segments.

It is important to take into account the fact that water ripples are blended along the wave. Collisions with the ripples should therefore be blended, otherwise the characters might seem to float above or go under the displayed water surface.

WAVE GOODBYE

There are many more aspects to water effects when creating a surfing game, but due to lack of space I will leave them to other articles, hoping that this one gave the reader a good understanding of how to approach such a problem.

As part of my conclusions while developing this technology I think that while the physics approach to generating big waves is very appealing, games that require fine control over these waves cannot rely on such an approach. However, I still believe that for games that contain shore waves with visual appeal only, the physics approach can be a viable solution rather than using animated waves (either through animation data, or texture passed to the GPU).

In our game the wave was too big and so I discarded the usage of projected grid and used the LOD method that was presented previously. The water, wave creation, and rendering were run on top seven different platforms successfully. On the Xbox 360, PC and PS3, the entire water process was implemented using the GPU solely. It took 3–4.5ms on top of the Xbox 360's GPU, and less than 5.5ms on the NVidia 7900GS GPU over a 1024x768 screen resolution.

The author would like to thank Jean-Francois Prevost for helping with this article, and Dominique Roussy and Eric Langlois for giving him the opportunity to develop the water technology for SURF'S UP.

The author also acknowledges Edouard Poutot for developing the water for the PS2, with additional thanks to Christian Desautels, Johan Verwey, Ned Mansur and the 3D team on SURF'S UP.

RESOURCES

Hinsinger, Damien, Fabrice Neyret, and Marie-Paule Cani, "Interactive Animation of Ocean Waves," ACM-SIGGRAPH/EG Symposium on Computer Animation, July 2002.

Hoppe, Hugues and Arul Asirvatham, "Terrain Rendering Using GPU-Based Geometry Clipmaps," *GPU Gems 2*, Chapter 2, pg. 27.

Jensen, Lasse, "Deep-Water Animation and Rendering," GDC & Gamasutra.com, September 26, 2001.

Jeschke, Stefan, Hermann Birkholz, and Heidrum Schmann, "A Procedural Model for Interactive Animation of Breaking Ocean Waves," WSCG 2003 posters proceedings, February 3–7, 2003.

Johanson, Claes, "Real Time Water Rendering—Introducing the Projected Grid Concept," Master of Science Thesis, Lund University, March 2004.

Kass, Michael and Gavin Miller, "Rapid, Stable Fluid Dynamics for Computer Graphics," *Computer Graphics*, Volume 24, Number 4, August 1990, pg. 49–57.

Knowles, Dr. C. Ernest, *Introductory Oceanography*, Chapter 10—Ocean Waves, NCSU 1997, **http://www4.ncsu.edu/eos/users/c/ceknowle/public/chapter10**.

Kryachko, Yuri, "Using Vertex Texture Displacement for Realistic Water Rendering," *GPU Gems 2*, Chapter 18, pg. 283.

Tessendorf, Jerry, "Simulating Ocean Water," Siggraph 2001 course notes.

Watt, Alan, *3D Computer Graphics*, Addison-Wesley, Second Edition, Chapter 6, pg. 163.

Watt, Alan and Fabio Policarpo, *3D Games—Real Time Rendering and Software Technology*, Volume One, Addison-Wesley, Chapter 5, pg. 82.

Wikipedia, "Ocean Surface Wave," **http://en.wikipedia.org/wiki/Ocean_surface_wave**.

Wikipedia, "Waves and shallow water," **http://en.wikipedia.org/wiki/Waves_and_shallow_water**.

THE NEW USER INTERFACE
EXPERIENCE

Scaleform® GFx

Artist-based **Flash**® UI development

Scalable image and vector graphic animation

High quality rich HTML text output

Efficient Asian font and IME text input

Available on all PC and Console platforms

*“Using our new state of the art CryENGINE 2 and Scaleform GFx,
Crysis will deliver an unrivalled cinematic experience.”*

Cevat Yerli, CEO & President of Crytek

*“Mass Effect is only the beginning of a long-term commitment by
BioWare to deliver an incredibly immersive cinematic UI experience.”*

Greg Zeschuk, President of BioWare

www.scaleform.com

Scaleform®

GDC ATTENDEES!
Visit Killer Tracks at GDC in booth 5338 (North Hall) to receive a complimentary copy of our new Video Game Music Sampler and more!

Perfect For:

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

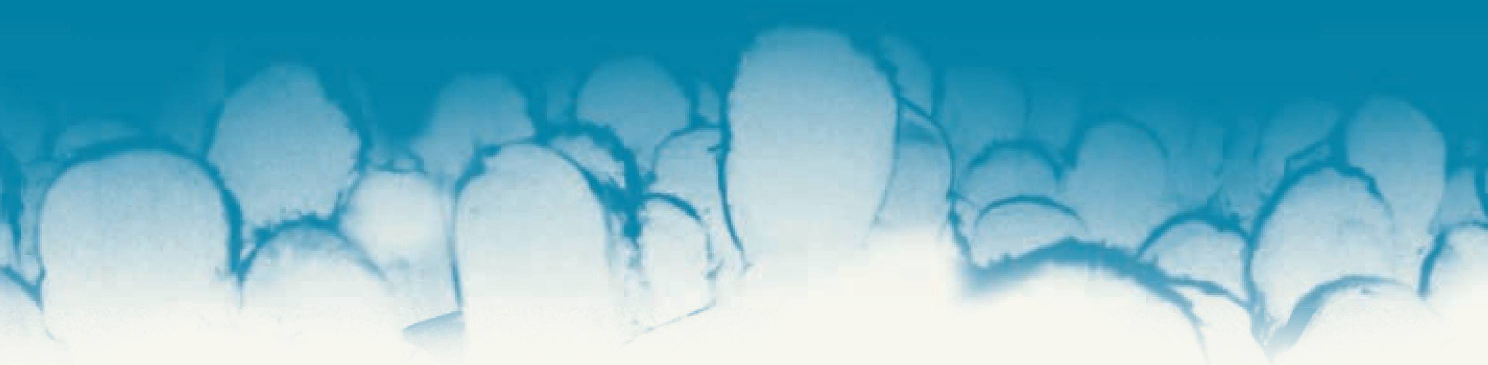
5.1 SPLITS AND LOOPABLE TRACKS AVAILABLE!

ASK ABOUT OUR **CUSTOM PRODUCTION SERVICES.**

Web: www.killertracks.com • **Email:** sales@killertracks.com
Phone: 1.800.4.KILLER • 8750 Wilshire Blvd Beverly Hills, CA 90211

a discussion with April Burba, Christian Schuett, Jonathan Hanna, Richard Weil, Sean Dahlberg, & Victor Wachter

COMMUNITY MANAGEMENT



COMMUNITY MANAGERS FOR MMOS ARE BECOMING MORE AND more common in online game teams. But best practices are few and far between. How should they interact with the rest of the team? To whom should they report? What do they even do? If you've ever asked yourself any of these questions, this article is for you.

***HOW HAS COMMUNITY MANAGEMENT EVOLVED IN THE GAME** industry over the last 10 years—what are the best and worst things that have happened to evolve the role?*

JONATHAN HANNA: The best I've seen is that community managers are being hired earlier and earlier in the development process. Ten years ago, they were hired when the game went live or right before. Soon after that they were hired right before beta started. Later they were hired about a year or more before beta. And now in some cases they are hired as some of the earliest members of the team. It's great to see developers realize the importance of having a community manager, even before they have a community. Planning takes time, and how you develop your community in the earliest days sets the tone for every following milestone. Community relations still needs to be involved more in the process, but this trend is a great sign.

The worst I've seen is the way community has moved away from taking responsibility for acquisition, or rather, they're asked less and less to focus on it. I've often heard people say that community is responsible for retention, and marketing and PR are responsible for acquisition. This is a mistake. The most influential person in a potential customer's decision to purchase a game is a friend who is currently playing the game, and nobody has more influence with those playing the game than the community team. Asking your community team to only be responsible for retention is asking them to only do half their job. They should work closely with marketing and be equally responsible for acquisition.

RICHARD WEIL: In terms of the good, some best practices are being developed, as the community professionals themselves have begun to view their roles as something specific and definable in corporate structures. Increasingly, community managers are performing vital and valuable tasks outside of the liaison role, such as beta management, event planning and execution and grassroots or viral awareness campaigns.

Some of the worst things include community relations departments having significant identity problems within most companies. Anytime you have to continually and repeatedly explain what you do, there are going to be issues!

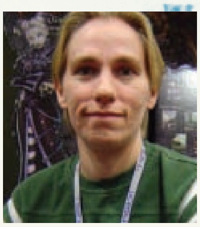
Community managers still don't, for the most part, cover what their peers do outside the game industry, such as organizing local outreach, charity participation by the company, and other elements of actual community relations.

Community managers tend to lack significant, dedicated representation in the upper tiers of company management, especially in larger organizations. This lack of representation can be detrimental to the mission of community relations when controversial or wide-ranging initiatives are formulated and/or implemented.

Initiatives that are formulated inside community relations departments sometimes have significant difficulties being implemented due to lack of "hard" resources, such as web development and graphics capabilities. The lack of executive champions can compound this problem, leaving good ideas and initiatives on the table for lack of implementation.

SEAN DAHLBERG *is creator of the Community Managers Group—a private association composed of professionals in the gaming industry that manage their respective communities. To learn more about the CMG, please visit www.communitymanagersgroup.com. Send comments about this article to* **editors@gdmag.com**.

COMMUNITY MANAGEMENT



APRIL BURBA *is community manager for NCsoft.*

CHRISTIAN SCHUETT *is community manager for Reakktor Media.*

JONATHAN HANNA *is a producer for Kaneva.*

RICHARD WEIL *is community director for Cartoon Network.*

SEAN DAHLBERG *is a community manager and designer at Stray Bullet Games.*

VICTOR WACHTER *is online community manager for Cryptic Studios.*

SEAN DAHLBERG: I've been working in the community management field for over seven years now and it's astounding to see the changes within in it during this short span of time. The field has been around for a long time outside of the game industry but still within the last decade. Where once it mostly involved forum administration and community management, it has now evolved into a liaison between the player base and development teams, which develops strategies for strengthening and building communities, tracks feedback, manages functions, and even creates assets such as interviews, podcasts, and product updates and sometimes we administer forums.

One of the issues still being dealt with to this day is that community is such a new position in game development and it is continually evolving and maturing. A lot of what has been and is still being learned within the field is through trial and error. Also, when compared to designers, programmers, or basically any other section of the development team, community management practices, procedures, and responsibilities and titles differ far more from company to company, creating something of an identity crisis.

VICTOR WACHTER: The best thing is that the worst things have already happened. Colossal mistakes have been made in the past, and we've learned, developers and publishers have learned, and communities have learned. Sure, that's a big blanket statement, but it's important. Most of us had little or no precedent to draw upon when we started in the field. We're through that learning phase, and community is much more effective for it. Phase three is profit!

On the other hand, our progress has been slow in terms of defining our field and turning it into a recognizable career discipline. We still don't have easily understood or descriptive titles and roles; for example, "community manager" often doesn't actually entail management functions, and I am still waiting for somebody to explain to me what "community specialist" is supposed to mean. It's an obstacle to doing your job effectively and to getting reasonable and realistic goals from your management.

What are the top three things a community manager (CM) is employed to do, in your opinion?

APRIL BURBA: *Be the liaison between the developer/publisher and the players.* This means helping developers and publishers make wise community decisions and being able to tell them if there are problems in a tactful way. The flip side is being able to take information that may be seen as a negative to a player and knowing how to make the best out of it or diffuse it rapidly when reporting it to the players.

Encourage and facilitate community between everyone involved in the game, such as developers, players, fansites, press, and third-party vendors. Help players create their own events, fansites, contests, radio stations, and wikis. Give them information, swag, and pimp their activities on your site. Attend their events. This gets them involved in your product and creates a sense of ownership and citizenship in your game. This also gets development involved with the playerbase and helps stop the "us vs. them" mentality.

Care for and feed super-users (the players who buy anything/everything associated with your product). Encourage them to become product evangelists by making sure they are fed information, listened to, encouraged, and generally feel good about your product.

> "Care for and feed the super-users. Encourage them to become product evangelists."
>
> —*April Burba*

HANNA: *Retention.* This is what most community teams are known for. Whether it's done through message board posts, news story updates, Q&As, developer chats, or helping a player with a problem, the community team should always be working to keep players informed, interested, and involved (maybe these are the three "I"s of community?).

Keeping the players informed helps them look forward to the future and feel less frustrated when things go awry. Keeping players' interest is all about giving them reasons to be excited and come back for more. The more the players are involved, the more sense of ownership they will feel—even if they don't agree with every decision you make. The better a community team is at achieving these the more likely players are to stick around.

Acquisition. An often overlooked responsibility for community relations is player acquisition. Community teams are in an excellent position to empower the player base to grow the game. Players are passionate about the games they enjoy and they want others to experience the game too. The community team should not only foster this, but create ways to facilitate players to be evangelists and spokespeople for the game. Good player retention practices can lead to player acquisition opportunities and the community team should take advantage of that.

Messaging. Just about everything the company does that affects the game or service needs messaging. When updating the game or changing a service policy there should always be a plan for how to communicate these issues to the community and those beyond the community. It's important for the community team to work with marketing and PR to create a consistent message and then communicate that message plan to anyone who might be talking to customers. Community professionals must often react to situations as they unfold in near real time and having good messaging practices helps make doing that easier.

How is being a community manager different from working in marketing or being a customer service representative (CSR)?

CHRISTIAN SCHUETT: I haven't been much into marketing yet but I have been working in customer support for quite some time.

In comparison the CSR is in direct contact with the customer like the CM is but in a more businesslike capacity. The approach toward the customer is not that personal and strictly focused on offering assistance to the customer's problem, be it account-, technical-, or gameplay-related. In my work as a CM I always try to build some kind of personal bridge to the community to give them the feeling that I am one of them, which often proves useful if you have to broach some bad news to them.

HANNA: The biggest difference between marketing and community is that for the most part, marketing is a one-way communication, and community is two-way. Marketing may create an ad or put out a press release, but they are rarely asked to respond to the players' reactions. The community team generally reacts in real time to player questions and concerns in addition to creating announcements.

That said, I've always felt community and marketing have more commonalities than they do differences. They're both responsible for communicating about the game, and more importantly, they're both ultimately responsible for acquisition and retention. They both talk to the community, they both talk to the media (or at least they both should), and both are responsible for creating the messaging that forms the basis for communicating with customers and potential customers about the game and service. In many ways, to be effective, community and marketing should operate as two sides of the same coin, supporting and working with each other as much as possible.

DAHLBERG: While marketing and community share many of the same goals and community can, at times, overlap with marketing, they are quite different. The main focus of marketing is traditionally the recruitment of new customers (acquisition) while the focal point for community is in the retention of those customers. This is not to say that community does not help acquire new customers because we do in more grassroots and viral marketing stratagems. In the end, both are working towards the betterment of the product but through means that are normally quite different.

You can say this about most departments within a company in relation to the community team. In fact, our former assistant community manager for SHADOWBANE gave a great visual example when discussing this very topic—The community team should be the hub of a wheel—attached to it are the spokes that lead to all the other departments. Information flows throughout the hub to all the other parts and without it, the wheel does not turn. That is how important

it is for all departments to work together as a team, and to understand and support each other.

What are advantages and disadvantages of a community manager being employed directly by a developer, being employed by a publisher directly, or being employed by that publisher's marketing department? Should a community manager's role be marketing- or editorial-led?

SCHUETT: In my opinion the CM's role is mainly editorial-led but touches marketing quite often. As the CM you are directly at the player base, gaining insight into what players think and what their needs are, which can be very useful for the marketing department. So I think this is a big advantage for a CM working directly for the developer instead of working for the publisher. On the other hand, it can also be a disadvantage if you are too closely involved with the project as it tends to blind you from the wider picture.

> "Marketing is a one-way communication, and community is two-way."
>
> *—Jonathan Hanna*

WEIL: A community manager or community relations department can "live" almost anywhere in the corporate structure, and there are benefits and drawbacks to just about any given situation. However, it is crucial that the community manager have strong ties to the development team.

The nature of the company, be it developer or publisher, is also an important factor in this question. Is it a multi-product company? In that case, there are significant advantages to a unified community relations department, though it is still important to keep community managers in close contact with the development team. I call this a "Federal" model, where community managers report to a community director (or higher) on the marketing side, but also are strongly integrated into their relevant product team. As an aside, the community team must always maintain close ties to the web team, if they are not one and the same.

WACHTER: The advantages and disadvantages are entirely circumstantial. The roles of publisher and developer across the industry are about as standard as the role of community manager is (not very). Naturally, it is easier to address those topics related to those functions carried out under the same roof you work under, but I don't think that says anything especially interesting about community relations. I think that most attempts to label community relations are artificial and counterproductive.

How should the community manager and a game's development staff interact in terms of feedback loops? What has and hasn't worked for you?

SCHUETT: Working as a CM developer, I can say that it's extremely useful to be able to take part in the daily stand-up meetings, plus the review and kick-off meetings in which the previous development progress is presented and discussed and the next tasks are determined. Aside from that, any upcoming matters can be easily discussed in standard meetings.

HANNA: The community team should be involved with just about every decision that will affect customers. This includes design decisions, customer support, service changes, and a host of other things. That's not to say community drives these decisions. They don't. But they're on the front lines of the service and they know the players and the issues they may have with decisions better than anyone else in the company. Having the community team in these meetings is a great way to plan for player responses to changes in the game or service, make changes to decisions before they cause a problem, and work on messaging for issues that may be controversial.

The earlier community team members are involved in the decisionmaking process (even if only in an observational capacity) the more effective they will be in communicating to the players and the press. If your community team understands the reasons behind a decision, the alternative ideas that were suggested but dismissed, and other aspects of the process, they will be well-equipped to create and execute an effective communication plan. There is no better way to achieve that than to have them present during the process.

If the community team isn't in these meetings, you're missing a huge opportunity and opening the door to negative reactions that many times can be avoided. It takes a lot more work to put out a fire than it does to prevent it in the first place and likewise, a good communication plan can turn a good decision into a great one.

WEIL: In addition to including the community team in decisions that affect customers, it is important that the community manager establish regular and productive reporting processes that give the development team and the rest of the company an accurate picture of what the state of the community is. This should be done at least once a week, preferably more, and could be in a variety of formats. It's crucial to include as many metrics as possible, such as topics of interest, number of forum posts, and any relevant media or fansite initiatives.

DAHLBERG: The community team is one of the best sources for knowing how the game is actually being played as opposed as how it was designed to be played. When the development team creates designs, programs features, or creates in-game assets, they have a goal as to how these will be used. The moment it is in the hands of the player, though, these can be used in various ways that were not imagined or intended by the development staff. Being in touch with the player base on the level it is, the community team is privy to how the game is being played by various groups more so than any other section of the development team and can bring that wealth of knowledge to the rest of the team.

In addition, the community team is usually on the "front line" and will know of players' concerns and issues before most other members. These are the individuals with whom the player base as a whole comes to have a great connection with, and will share information with. ⁙

POSTMORTEM

RATCHET & CLANK FUTURE: TOOLS OF DESTRUCTION

JOHN FIORITO *has been making games since 1992. He joined Insomniac in 1996 and worked as an artist on* SPYRO THE DRAGON *(read about that in the September, 1999 issue). Since then he's worked as an art director, project manager, and is presently COO. Email him at* **jfiorito@gdmag.com**.

ONE OF THE FIRST FEATURES DEVELOPED FOR RATCHET & CLANK FUTURE: TOOLS OF DESTRUCTION was the "Groovitron"—a bomb that was part boom box and part laser light show. Throw the bomb, force your foe into spontaneous dance, and then kill him to a disco soundtrack. It was hilarious to watch and just as fun to play. But the music almost died when we realized that we'd have to give every character, enemy, and creature a set of unique dance moves. The Groovitron would require hundreds of animation cycles, special case programming, and extra work across the entire project. In many ways, this reflected the challenge of bringing our heroes onto new hardware; the complexity, sophistication, and power of working on a new platform meant that our team would need to perform at an equally complex and sophisticated level. We had our work cut out for us but we felt we had an opportunity to create the RATCHET & CLANK game we'd always imagined.

RATCHET & CLANK FUTURE: TOOLS OF DESTRUCTION (RCF) is Insomniac's fifth RATCHET & CLANK game. We needed to deliver an experience on par with its predecessors while making a fresh debut on the PlayStation 3. Our challenge

GAME DATA

Developer:
Insomniac Games

Publisher:
Sony Computer Entertainment

Release Date:
October 23rd, 2007

Number of Developers:
70 full-time, 30 shared, 25 contractors (music, voice acting, testing, localization).

Length of Development:
23 months: 11 months preproduction, 12 months production.

Platform:
PlayStation 3

Hardware Used:
Artist workstation: Dell Xeon 3.2G with Quadro FX3400.
Programmer workstation: Quadcore PC with as much memory crammed into it as possible.

Software Used:
Proprietary engine and development tools. **Artist software:** Maya, Photoshop, ZBrush, Turtle. **Asset management and revision control:** Perforce. **Design/gameplay scripting:** Lua. **Compiler:** custom GCC compiler provided by Sony. **Character lip synching:** Annisoft.

Number of Times Art Director Spoke Ill of Dragons:
Zero—for he is crunchy and good with ketchup.

Technologies Licensed:
SpeedTree, Bink, Microsoft Visual Studio, Anark (UI menus), DevTrack (bug database).

Project Size:
Game size: 22.5gb
Project size: 1000gb
Bugs: 16,000
Source Files: 4429
Lines of code: 980,184

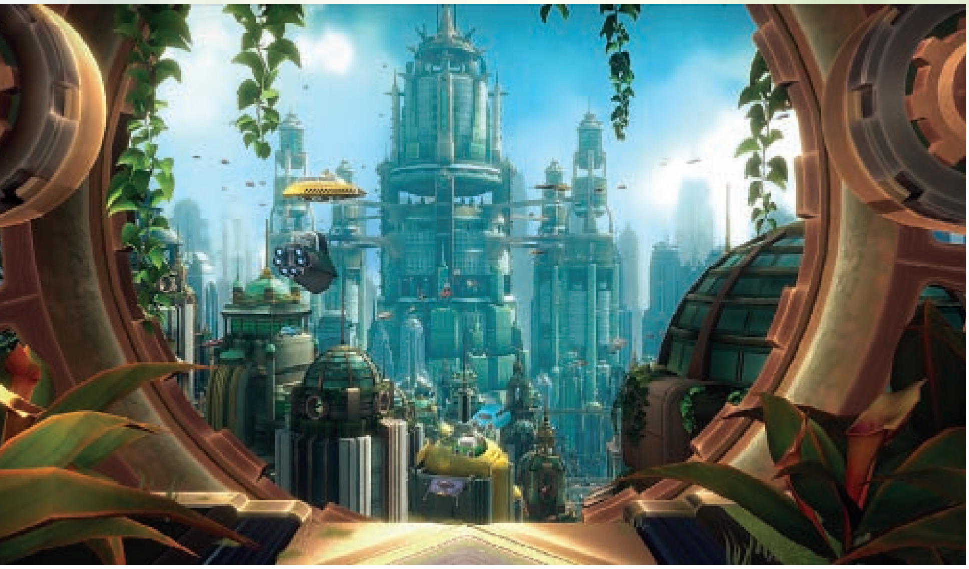


FIGURE 1 The opening view of Metropolis as we imagined it in 2005.

became even more daunting when we added to this a tight one-year development cycle (plus one year of preproduction with a skeleton crew), a production budget only half or two-thirds of similar games, and no experience working on the PS3. It was necessary to recreate the Ratchet & Clank universe from scratch for PS3, and what follows is an account of the many issues we faced, the decisions we made, and the methods we used to do this.

WHAT WENT RIGHT

1 **SET A VISION.** In late 2005 most of Insomniac was working mightily to get Resistance: Fall of Man up and running on early PS3 hardware (read the postmortem in the February, 2007 issue). Meanwhile, about a dozen of us were trying to figure out where to begin on a new Ratchet & Clank game. We were lacking hardware, an engine, game code, and even assets. We were truly at ground zero. To start, we decided to visualize the Ratchet & Clank universe PS3-style by recreating Metropolis, one of the iconic locations from our PS2 series. We did this by building a "diorama" of the city, adding vehicles, and sending a camera through it. We built our test city using the Resistance engine, stitched together a frame by frame camera fly through, and added audio effects to simulate the experience of being in Metropolis. We were pleased with the outcome, but we knew that this was a guess at best and revealed more about our hopes for the game without the memory, frame rate, and game design constraints of a real level.

The reaction from our producers at Sony and later from people who saw the Metropolis video at 2006 GDC was astonishing. We were being compared to feature film CGI, and there was great enthusiasm for the game. When Sony told us that future gameplay deliveries needed to "drop jaws" as Metropolis did we wondered if we could ever match the results in-game (see Figures 1 and 2). At this point we still did not know what lay ahead for RCF, but we knew one thing—we had a vision.

2 **PRE-PRODUCTION TIME.** Armed with our demo, we set about trying to plan the game. It was the beginning of 2006, and we had 10 months to get ourselves ready for production, which would begin in earnest when Resistance shipped and the full team joined us. We still had a lot of unanswered questions; What would Ratchet look like on PS3? How would we animate characters that were four to 10 times more complex than the previous generation? What constituted a "level?" How big was the game going to be, and how would we make it all work?

To start, we established the overarching goal of creating one fully-realized and playable level. Hopefully, this would give us two valuable pieces of information; first, an understanding of everything needed to get our game up and running, and second, a set of benchmarks to plan the content for the rest of the game. We also had three distinct production paths that we needed to keep aligned—asset creation, gameplay prototyping and implementation, and game design. To keep things organized, we set a series of escalating milestones. These included first prototype, first functional, and first playable builds that would eventually lead to a completed level. Our milestones were shared across the entire team so that we maintained unified production goals. In between, we divided specific assignments into one- to three-week blocks to keep the workload comprehensible to the members of our team. We met our goals, but wound up overshooting in terms of our design scope. See "what went wrong" for more on this.

3 **BALANCING CREATIVITY AND DEADLINES.** After our PS3 launch title Resistance had successfully shipped, our 15-person preproduction team began to swell to more than 100 people (70 full time and numerous "shared" resources). To carry out our game design, we needed to keep the entire team productive from day one. As much as possible, we needed a stable production environment with a constant flow of game code and game assets ready for whoever needed it next. Time was our biggest enemy and we needed to do whatever possible to use it wisely.

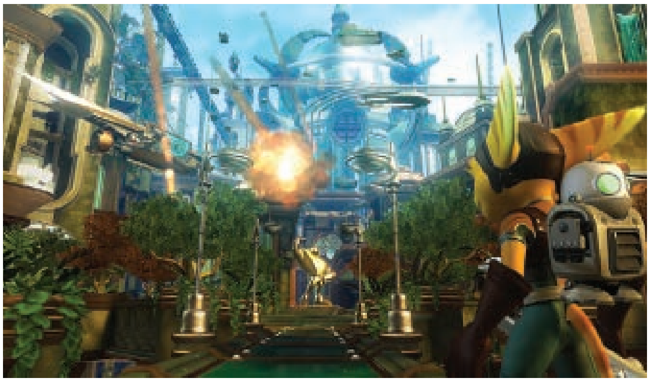


FIGURE 2 The in-game opening view of Metropolis in 2007.

We attempted to establish a production structure that rationalized the complexity and scope of our game but also gave everybody working on the project flexibility. One of the tenets of

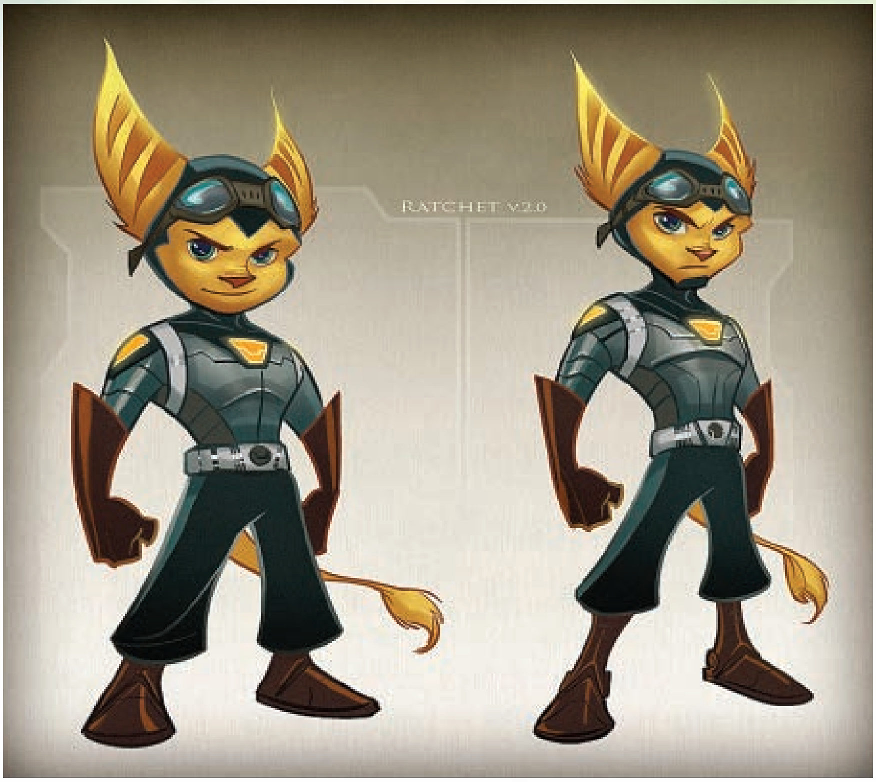


FIGURE 3 Early character studies for a next generation Ratchet.

Insomniac's culture is to allow creative contribution and control at all levels, and we had to balance this with another Insomniac truth—we never miss a deadline. Within our major milestones we established production cycles, eight-week blocks of time where our various teams created components of the game. At the end of each cycle, these pieces were brought together, often resulting in a play test, media presentation, or both. During the project, we were able to establish and track progress according to these recurring cycles while individuals or teams could fit their unique workflows into a cycle. At the same time we learned that creating functional gameplay rather than finished components not only accelerated progress but embraced iteration and change along the way. On past Insomniac productions we often tried to take features to completion before moving on, which complicated the inevitable changes and added unnecessary work.

As production intensified, we adopted additional strategies to keep up our momentum. Early on we instituted "Friday walkarounds." This was where the creative director and project manager would visit each production department (art, animation, design, gameplay, etc.) to see work in progress or evaluate completed assignments. We tried to keep this casual—usually we would view work at its creator's desk. However, by sticking to a regularly-scheduled routine we built a formal evaluation structure that focused the team on weekly, self-determined progress. Later, as the game was taking shape, we held Monday morning "war room" sessions where our designers and programmers would report the status of the game and what lay ahead for the coming week. Again, this took place in an informal setting, but ended up being the other bookend to our Friday walkarounds that organized RCF production into discreet one-week segments. Still later, we added a "Daily Load Test" report that our Q/A team sent out to everybody working on the project. This was a simple test that reported if a level would load, if it could be completed, and what issues stood in the way. It came at the time during the project where maintaining stability on a daily, or even hourly basis was essential, and gave everybody an accurate way to understand the most critical issues.

4 **LESSONS FROM THE PAST.** One of the biggest aids to RCF production was the experience gained on RESISTANCE. We were now working on a second-generation PS3 title, and we benefited greatly from working on a game engine that survived the rigors of launch title production. Meanwhile, our technology continued to develop. RCF included a new form of texture streaming that allowed us to add great variety and lushness to our surfaces and further optimizations meant we could target 60 frames per second instead of 30. By shipping a game on PS3 we proved to ourselves we could do it successfully and gained the kind of understanding that only comes by doing. We did not know everything to expect, but we had a good head start.

5 **FIRST PARTY RELATIONSHIP.** Another big help was our "first party" developer/producer relationship with Sony. On all fronts we received support that was immeasurable. At a time when PS3 development kits were scarce, we had 150 of them in our studio. Sony Europe and Japan provided localization support for 13 languages, and our game was tested throughout America, Europe, and Asia.

Creating a basic structure and organization of RCF's development didn't require any rocket science—though there's plenty of it in our code base. The most important lesson we learned was that using the above measures helped put our team in a position where we could succeed on a very ambitious production. Once we realized this, we believed that we would succeed, and achievement no longer became an assignment. It became our expectation.

CONTINUED ON PG 38

They're back to save the world.

Want In?

www.terminalreality.com/jobs

CONTINUED FROM PG 36

WHAT WENT WRONG

1 **NEW-GEN SCALE.** In general, we were unprepared for the natural challenges that came with working on a new-generation title, even during pre-production. Everything seemed to take longer than we wanted it to, and making changes was not as simple as it had once been. Establishing a design for Ratchet proved especially challenging as we considered redefining the character. He went through numerous iterations before we felt we got him right (see Figure 3). At the same time, we were still months away from the PS3 hardware launch, so technical stability was inconsistent at best. With these problems and the resulting lack of in-game feedback, it was especially difficult to lay out level designs. Progress was slow, but we were still making some. By the end of preproduction we had one semi-complete level that was functional but not very stable. This would eventually become "Kerchu City" (see Figure 4).

In addition to a level, we gained our first production experience that enabled us (or so we thought) to look at our overall macro design to determine if we had enough time and resources to complete it. We did not.

At the end of preproduction RCF's design called for 25 unique planets, 5 space combat missions, 1 hour of cinematic cut scenes, a hazily-defined co-op mode, and an even more ambiguous online component. We wanted the game experience to last about 15 hours—and we achieved this—but the final scope of our game was 16 planets, 3 space combat missions, 45 minutes of cinematic animation, and single player mode only. When we scaled back, it was not a popular decision but it represented a turning point in the project. We now had a design that the production team felt was achievable (though still ambitious), and we had removed content from the game before it ever went into production. Nothing was "cut," and during actual production no work was ever discarded. We had spent almost one year piecing together a single level of our game. With less than a year to go, and 18 levels left to create, we remained optimistic because we now had a plan that we believed in.

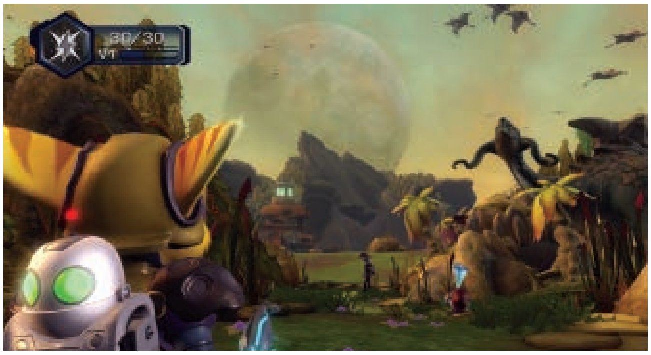


FIGURE 4 Kerchu City as it appeared at the end of preproduction, fall 2006.

2 **LAUNCH TITLES ARE NOT PRODUCTION BENCHMARKS.** As noted above, our knowledge gained from building a preproduction level and shipping RESISTANCE was an invaluable aid to our planning process. Without this data many of our production estimates would have been way off. What we failed to realize, and what hurt us at the end, was our lack of understanding in terms of what it took to finish a game of this scope. Our launch title experience did not (and could not)

Come and get it!

- Havok™ products will make your game-building experience better, faster and safer. Our products include the battle-tested **Havok Physics™**, **Havok Animation™** and **Havok Behavior™** for great characters and coming soon, **Havok Destruction™** for simulating rigid body destruction and **Havok Cloth™** for character garments and environmental cloth.

- Havok™ Products are built to use off-the-shelf to allow you to hit your deadlines and budgets. We offer full support for leading game platforms including Nintendo® Wii™, Sony® PLAYSTATION®3, Sony® PlayStation®2, PSP™, Microsoft® Xbox 360™ and the PC.

- And if you are pushing the envelope, we have a Professional Services team to help get the most of our products and help make your game the best it can be.

Havok™ Sponsored Sessions at GDC:

Havok Cloth™
Wednesday February 20th at Noon to 1pm
Speaker: Paco Vidal / Room 2010, West Hall

Havok Destruction™
Wednesday February 20th at 4pm - 5pm
Speaker: Oliver Strunk / Room 2010, West Hall

Havok Behavior™
Thursday February 21st at Noon - 1pm
Speaker: Jeff Yates / Room 3016, West Hall

License Havok Behavior™ between February 20 & March 31, 2008 & get Havok Animation™ for free!

Contact us today for details at gdcspecial@havok.com

havok™

Dublin *San Francisco* *Tokyo*

Offer applies to Havok version 5.5 only and not earlier releases. License must be purchased or ordered on or before March 31, 2008 and payment date is based upon agreed payment schedule. Offer is open to professional game developers only with appropriate rights to develop on the applicable licensed platform(s). A license to Havok Physics™ is required. A non-disclosure agreement and Havok standard software license agreement (if not already in place) are also required. Havok™: ©Copyright 2008. Havok.com Inc.(or its licensors). All Rights Reserved. See www.havok.com for details.

provide measurable production benchmarks. It was a sprint to the very end and involved many unknown variables. There really was not a true post production period as RESISTANCE development continued right up to its launch. Working on a simulated level during preproduction was also inaccurate since we did not account for many of the issues that need to be dealt with when shipping a real game (such as memory and frame rate optimizations, obscure crash bugs, and localization errors). We gave ourselves eight weeks from our project alpha date to "polish" our game and get it out the door. This was the same amount of time used on our PS2 titles, which had been loosely based on our PS1 model. Needless to say, it got a little crazy at the end of the project, and we've learned to add additional post production time to our future efforts.

3 **DIFFICULTY CREATING STABLE WORK BUILDS.** During RCF, our greatest periods of progress occurred when all of the pieces clicked, and the project seemed to take on a life of its own. The end result would be a stable working build of our game that could be play tested and further refined. Unfortunately, stability remained elusive throughout development and often required Herculean efforts from our tools, tech, and gameplay programming teams to achieve. There were times when we could not produce a clean build of our game for up to three weeks. Because of our compressed development cycle, we needed to continue creating assets and wiring the game but we had to do it blindly as our stabilization efforts took place. Once we fixed the problem(s), the resulting change lists submitted to the project were so massive that they invariably broke the game, which again required us to re-stabilize. This frustrating downtime hampered our ability to tune the game iteratively. This represents an ongoing struggle at Insomniac.

4 **DEVELOPING TWO GAMES SIMULTANEOUSLY.** In addition to working on new hardware, it was a new challenge to have two games in simultaneous development, and we felt our share of growing pains. Previously, we focused the collective effort of our studio on one title at a time. Any important issue could receive the attention of our entire company if necessary. But now, we had two games to think of, and needed to work with people who were considered shared resources and not always available. What proved more difficult than sharing people was hiring people. When RCF went into production we were understaffed, especially on our programming team. Yet, we did not adjust our plans and remained hopeful that we'd find the right person soon. We did not complete our hiring until three months before our release, and this meant that our understaffed departments had to take on extra work, which is not a sustainable strategy.

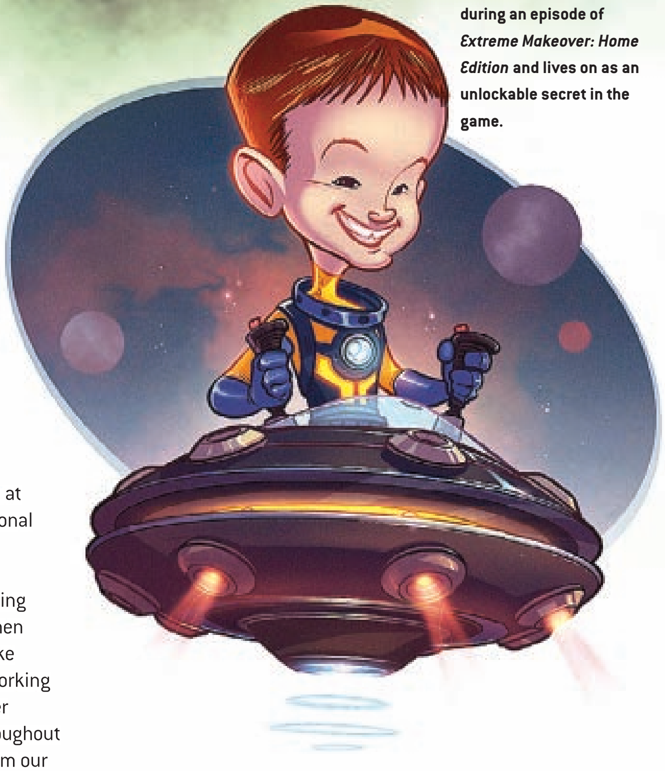


FIGURE 5 Captain James, who appeared in RCF during an episode of *Extreme Makeover: Home Edition* and lives on as an unlockable secret in the game.

5 **EXPECTING THE UNEXPECTED.** RCF also experienced its share of curveballs. We were given the opportunity to show our game as part of the TV show *Extreme Makeover: Home Edition*, provided we create a unique character to tie in with the episode (see Figure 5). This resulted in some good publicity and one happy boy in Oklahoma, but it cost us a week. We were running builds on PS3 devkits but Sony Q/A only worked on test kits, a miscommunication that forced us to scramble to fit our builds on disc earlier than planned. We prepared a downloadable demo of our game to coincide with E3 but it was an in-progress demo that only ran at 30fps. So we scrapped it and made the demo again as we were closer to finishing the game.

FUTURE TOOLS

I doubt any of these production problems are unique, and many of them have been solved by other studios. Our biggest challenge was to deal with them in the context of creating a game, which certainly raised the urgency and stressfulness of the situation. A lot of things did not work out for us, but a lot more of them did. We were able to stabilize many of our development techniques but know there will always be more ways to improve. And as we continue our mission to make great games, we've moved a step closer to a smoother production model.

TRINIGY

the perfect solution for advanced game development

VISION GAME ENGINE

GDC 2008

VISIT US AT BOOTH 6105 NH

WWW.TRINIGY.NET

Games for Windows XBOX 360™ XBOX LIVE arcade PLAYSTATION 3 PLAYSTATION® Network

NOW IN THE US:

TRINIGY, INC., 701 BRAZOS STREET, SUITE 500, AUSTIN, TEXAS 78701

dan boutros

CREATING HALO 3'S LEGENDARY MODE

A Q&A WITH GAMEPLAY DESIGNER FRANCOIS BOUCHER-GENESSE

CONTINUING THE DISCUSSION FROM THE DIFFICULTY ARTICLE (PG 6), *Boucher-Genesse discusses in detail the thought and process that went into the creation of HALO 3's Legendary difficulty mode.*

DAN BOUTROS: *What do you think are the issues surrounding difficult modes in games, and what do you think is behind some getting it right and others not?*

FRANCOIS BOUCHER-GENESSE: What players are usually looking for when playing on higher difficulty levels is a good challenge. But for a game to be challenging, you need to feel like it is fair, and that even if you failed you still had a chance at success. The more you reward creativity with success, the more you encourage players to think differently, and the more your game is going to present an interesting challenge to them.

This is done at the core gameplay level, for example by limiting players to two weapons only so that they have to figure out the best possible combo for the job. But it is also done on a mission level, by strategically placing enemy turrets in positions that can be turned against them.

An easy mistake to make when trying to create a challenging game is to make one which is entirely predictable. When an enemy always has the same behavior, and always appears behind that same rock, the challenge is pretty clear. But how fun is that? Players want to be surprised, and being suddenly put in extremely dangerous situations, like when a grunt decides to pull out two grenades and rush them. You just have to make sure that these situations have clear ways out.

DB: *The difference in difficult mode design between HALO 1 and 2 alienated some fans, as appears to be well-documented on particular fan forums (Neogaf as one example). What can you tell us about the changes made between HALO 1 and 2?*

FBG: I wasn't part of the team when they worked on HALO 2, but I did hear a lot of stories about its development. Wrapping the game up in the last months was really challenging for the team, and since difficulty tweaks are mostly made in that timeframe, it is safe to assume that they just had less time to examine these issues.

I think the playtests from the usability labs were also less frequent than they were for HALO 3, so that the designers had less feedback from players on which parts were difficult. It is really hard as a mission designer to take a step back and look at how difficult your mission really is, since you already know the clever things to do when you play them. Playtest results can be painful reality checks for designers, but they're still really useful.

DB: *What was changed from HALO 2 to 3 to restore that fan love for Legendary mode once again?*

FBG: The first obvious answer is that the development cycle was really different for HALO 3, and that we just had that much more time to polish the game, listen to feedback, and make it fun on every difficulty level.

I think you can understand how HALO 3 made a better game for Legendary fans by looking at its difficulty spikes. On normal, if you screw up, you can usually get out of the situation and reengage. When you play on Legendary though, if you get yourself in a bad situation, you're likely to die. It is really important in that context that players feel like they could have avoided this defeat, that they understand "the challenge" and replay it, confident that they can try a different approach. This requires encounters that are not totally scripted, and HALO is pretty good about presenting players with many options. If a vehicle is engaging you, the game is not just about shooting it. Should you run for the rocket launcher? Or wait and board it? What about stealing a plasma pistol from a Grunt, and powering the vehicle down? A lot of games try to offer you many options, but when you get to play them on higher difficulty levels you often feel like there is only one real way out of the challenge.

So on one hand you give the player many tools for the job, and on the other hand you make the player's job harder. As long as that player tries a different approach every time he or she dies, you know you've balanced the game properly. One of the interesting things we found out in the usability lab playtests is that sometimes, areas where most players die are still rated as the most fun. An encounter with a Scarab is a good example of this—I couldn't believe the high fun ratings when I saw how often players were dying in the first playtests we did with the Scarab.

The game is also really different on Easy/Normal than it is on Heroic/Legendary. You probably won't notice it at first, but there will be an additional turret denying you the easy way out of an encounter. Or a stealth Brute will be waiting for you on the other side of that cover which used to be safe. So it's not like we just cranked every enemy's health by 200% and called it Legendary—there was a good amount of custom changes that were made per mission as well. In that sense we encourage players with previous HALO experience to play at least on Heroic, since they get to see the game in its full scale.

I spoke with the guys that tweaked difficulty on the other HALOs, and the conclusion is that a really similar "formula" was used for every halo game. What did make a difference was the time spent tweaking and fixing issues to make the game fun on every difficulty level. All titles had more bad guys, stronger and more accurate enemies with faster projectiles. And they used similar numbers for each of these parameters.

New HALO titles added new player options (like vehicle boarding, equipment), but HALO 2 didn't allow players to use them as much as on HALO 3, since it lacked the polish time HALO 3 had.

You've got the looks. We've got the brains.

Kynapse A.I. Middleware

Kynogon's A.I. middleware, Kynapse, is the brains behind the work of some of the leading game developers out there. In fact, **6 of the top 10 game publishers** use Kynapse in developing their best-selling titles.

INTEGRATED UNREAL ENGINE PARTNER

www.kynogon.com

France ■ United Kingdom ■ Canada

kynogon

no mind no game

Softimage XSI 6.5

By Carey Chico

SOFTIMAGE

★★★★★

STATS

Softimage Co.
3510 Saint-Laurent Boulevard
Montreal, Quebec
H2X 2V2 Canada
www.softimage.com

PRICE:
Softimage XSI 6.5 Advanced: $4995
Softimage XSI 6.5 Essentials: $2995

SYSTEM REQUIREMENTS:
Workstation with AMD K8 or Intel Pentium IV of 1 GigHz or more.
OpenGL accelerated graphics card with 128 MB RAM or more.
1 GB RAM.
Windows XP Professional SP2 or x64 Edition.
Red Hat Enterprise Linux ver. 4, kernel 2.6.9.
Fedora Project Core 3, kernel 2.6.9-1.667smp.
Novell SUSE LINUX 9.3.
Default GNOME window manager or KDE.

PROS:
1. Brand new material engine with visual aides.
2. Awesome new Animation Layer system.
3. Brand new reference model system.

CONS:
1. Lackluster transform multi-object uv tool.
2. Still missing some industry-leading pelt and LSCM mapping algorithms.
3. Price point in 6.5 has increased.

ONE THING I'M REFRESHED ABOUT whenever I check out the latest version of Softimage's premier modeling and animation package—XSI—is how Softimage strives to keep the workflow and working philosophies intact. Softimage released 6.0 late last year and 6.5 to the public just recently, and the software engine shows no signs of the age you might expect to see at this stage of a product's life.

The Softimage 6.5 release is certainly a mid-year point release, which means it's not a tremendously new-feature-laden release. Overall, XSI 6.5 refreshes the 6.0 release, improving upon a software product with an already stellar workflow and adds high-end feature sets from the Advanced version like XSI Hair and Fur and Syflex Cloth to the lower-cost Softimage Essentials.

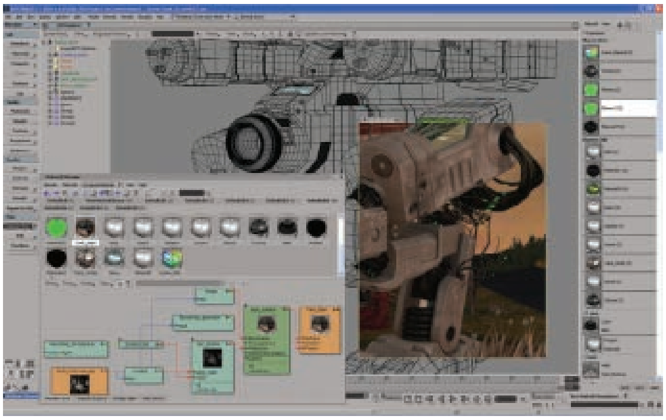
You would be looking to 6.0 for the huge new features that have gone in, including a redone Render Tree interface and feature set, a completely redone rendering engine based on Mental Ray, a brand spanking new Reference model system and—one of the coolest new features—animation layers.

Given the fact that XSI 6.0 was never reviewed here and since it's intimately tied to the 6.5 release, I will highlight some of the best new features that I was excited about from 6.0 plus comment on a few new things in the 6.5 release.

NEW RENDER TREE

In 6.0, Softimage completely revamped the Render Tree in order to streamline its operation and add some long desired features. The biggest new feature apart from the Material Manager and refined UI would be the new shaderballs which can appear as little thumbnails on each render node to indicate what the render would look like in that stage of the render process. It's been long awaited and Softimage doesn't just add quick features to fill bullet points for marketing but rather works in new tools as components to larger feature implementations. The shaderballs work wonderfully and you can also pop up larger shaderball versions in their own window if you wish.

Additionally, the Material Manager provides a streamlined interface for previewing scene Materials and creates a preview interface for preparing new materials. It directly interfaces with the existing material library, plus you can drag and drop to create new materials on the spot. It really helps with the workflow, especially for scenes juggling a high number of materials.



Softimage XSI's Material Manager.

To handle the newly revamped renderer functionality, Softimage introduced a new Render Manager Interface. At first glance, it's a bit convoluted due to the myriad of settings and options accessible to the user. But once you get around the huge amount of control options, you'll understand how they lay out and it will begin to make sense. It took me some time to figure out on the new interface where they put the old options that I had grown accustomed to.

REFERENCE MODEL SYSTEM AND TEXTURE SUPPORT

Coming out of a long research and development phase, Softimage revealed the new reference model system in 6.0 and updated and fixed some errant issues in 6.5. This new system now works with the concept of delta nodes. These are change nodes that exist on top of the original model. All changes can be stored in these delta nodes. You can also define an unlimited number of model resolutions and retain the connection in the scene. The system supports nested reference models and you can now import COLLADA files as referenced models. As a whole, I like the new reference system better as it enables more interaction with reference models than the old system previously allowed. It's also much easier to create and interact with reference models.

More tweaks were incorporated into the already excellent Texture Editor in the manner of a new "Tight" algorithm for the Unique UV Polypacker. This new algorithm is a leap over the earlier one that didn't optimize the UVs as well as it should have. I still think XSI really needs to get their own LSCM and Pelt solutions but I should add that Kim Aldis created a really easy to use plug-in that uses the Roadkill tool to add this functionality.

Something that was much needed for the game industry was the new support for Satellite rendering optimizations for the Rendermap function. This means that users can now get a speed increase when using additional satellite machines added to the .rayhosts file.

One new touted feature in 6.5 I find extremely lackluster and not very useful is what Softimage calls "Transform UV Property for multi-UV editing." At first, this sounded cool and I thought it was support for editing the UVs on multiple objects simultaneously. In games, we do this a lot because we are combining several objects' UVs onto a single texture page and we want to keep the objects separate. Once I realized what they actually included I was rather disappointed. What they created was a little UI that permitted the user to numerically transform the UV samples. You had to type in a number and press a key to actually do anything with multiple UVs.

This was definitely my biggest disappointment with 6.5 but I managed to speak with Softimage on this issue and was able to test a customized build made for Electronic Arts in which they added complete, real-time transform support for multi-object UV editing and it worked like a charm. One can hope that this revised feature will work its way into the next product release.

ANIMATION FEATURES

The new Animation Layers system has to be one of the best new features coming out of 6.0. This system allows you to non-destructively add animations on top of previously existing animation data—whether using action clips or animating directly on the FCurves. This has a profound workflow impact and I was immediately intrigued by all the possible ways I could use these layers—from organization to working with motion capture data that I wanted to adjust but not destruct. This new feature keeps Softimage in the running as one of the premier animation packages out there.

HAIR AND CLOTH

Specific to the 6.5 release, the addition of the Syflex cloth system into Essentials is a really big feature improvement for that mid-level version of XSI. Syflex has had its cloth simulation technology available for a while for Maya, Houdini, Lightwave, XSI, and 3ds Max and was written by some of the same team that wrote the cloth for *Final Fantasy: The Spirits Within*. It's super fast and super easy. Just as an example, I took a low-poly grid and set it up as cloth, and started running the simulation. While it was running, I selected the low-poly grid and subdivided it 3 times and all the while it kept animating and moving smoothly. I was impressed.

Hair is Softimage's own creation and it's been in the Advanced package for a while now. It's great to see this tool now available to a larger user base as part of XSI Essentials.

CONCLUSION

As far as 6.5 goes, you should buy it for the refinements, bug fixes, and improvements that Softimage implemented for one of their largest releases in 6.0. Apart from a few lackluster little features added to the mix, the big ones carry the day and are more than worth their weight in this fine package.

CAREY CHICO *has been in the game industry for over 11 years and is currently executive art director at Pandemic Studios. Email him at* ***cchico@gdmag.com.***

DECODA 1.07

By Brian A. Stone

There are lots of good reasons to build a scripting engine into your game. The usual arguments are data driven development, reusable engines, empowered designers, rapid prototyping and rapid iteration. But forget all that. Games are interactive by definition, or they would be movies. So when content is defined as interactive, treating game logic as data just makes good sense.

Sometimes designers are given the keys to the kingdom by turning them into script programmers with deep hooks available into the engine. In other cases visual tools constrain logic input carefully and actual script code is only written—or generated—in bite size chunks. Either way, chances are good that a full-featured language under the hood will give the most flexibility at the lowest cost.

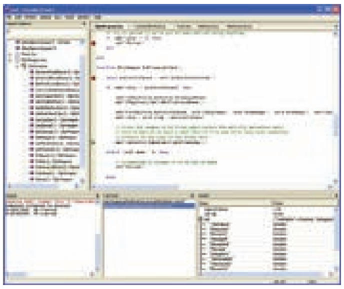
There are plenty of good scripting engines to choose from—enough that rolling your own doesn't really make sense any more. My favorite has always been Lua. Lua integrates easily into any engine, its syntax is simple yet deceptively flexible and powerful, and it is well supported in the industry. Most importantly for game development, the implementation is lightweight making it a reasonable choice for both PC and console development and full source is distributed free of charge without

restriction. Unfortunately, the out-of-the box debugging options are quite limited. This is where Decoda steps in.

Decoda is a full graphical IDE for debugging Lua script in your applications. Decoda looks something like a lightweight Microsoft Visual C++ (MSVC) for Lua. Straight from the feature list, it supports syntax highlighting, symbol browsing/filtering, configurable hotkeys and colors, custom tools, and more. It integrates with source control (SCC), plays well with alternative Lua distributions like LuaPlus, and even attaches to MSVC for debugging native code. When debugging Lua script you can set breakpoints, step-into or step-over functions, see the call stack, and keep watch lists.

Decoda does not require any code changes in your application to work. It just needs to be able to locate the Lua API functions within the application's executable. This works if your application dynamically links with Lua (lua.dll) or if Decoda can find the PDBs that match your executable. Modified Lua distributions will work as long as the original API is still supported.

As I put Decoda through its paces for this review, it mostly worked as advertised. I tried attaching to several sample applications, both linking to Lua dynamically and building it in statically. In both cases Decoda attached without problems. I also tried some alternative Lua distributions like LuaPlus. Here also, Decoda attached correctly. For a more real world test, I attached to a large in-development game project that makes heavy use of Lua built statically. This also worked without modifications. The only failure case I found was trying to attach to a heavily optimized release build. Decoda attached but failed to find some of the Lua API functions thus preventing debugging. If you have trouble attaching to your application, I found the support both through mail and on the forums quite responsive.



Decoda 1.07 Lua debugger.

Working with a sample LuaPlus application, I set breakpoints, stepped through code, and explored data in the watch window. Decoda performed well. The most complicated data structures I could devise expanded naturally in the watch list. I could even explore the global memory by simply adding "_G" to the watch window and expanding it out. Multiple virtual machines are also supported and as you switch between them, the watch window updates accordingly.

Unfortunately, Decoda does have some important limitations. I was very pleased with its performance on my sample applications. However, using it to debug my full-scale game development project left me wanting more.

The functionality of the project window is too simple when managing large numbers of files. The project window simply displays a flat-sorted list of open files with no way to organize them. For projects with lots of script files, especially if they share the same name and are only differentiated by location, this makes it extremely difficult to find things. I also started running into some bugs once large numbers of files were left open. For example, opening a Lua file into Decoda from the file explorer didn't work if a file of the same name was already open—even if it was actually a different file from another location. Furthermore, cleaning up the project window started failing—leaving temporary files from previous sessions open and no way to close them without restarting the IDE. Finally, if you use Decoda to stop an attached project, then not all the virtual machines will get destroyed, which again has the side effect of leaving open temporary files.

The debugging options were also just enough to get by but whetted my appetite for more. Noticeably lacking are a "locals" watch window and data breakpoints or some form of conditional breakpoints. And although it's wonderful to support multiple virtual machines, this could have been fleshed out further. The watch window updates when a virtual machine is selected, but sorting open project files by VM, setting breakpoints conditional by VM, and custom watch windows by VM would all have been welcome.

It wasn't an issue for me, but some may be bothered by the standalone nature of Decoda. The IDE is fully separate from MSVC instead of implementing itself as a plugin. You can simultaneously do native code debugging from MSVC and be attached with Decoda for Lua debugging, but it requires task switching between IDEs.

Finally, for us console developers; the most noticeably lacking feature is remote debugging. According to the developers at Unknown Worlds, this is a feature they are looking into for the future, but it is not available today. Decoda is only available for Windows.

Compared to the open source offerings for Lua debuggers Decoda is probably worth the reasonably modest cost. I've explored the free alternatives and although you can find debuggers with similar functionality, they are not as well supported or as robust. For example, LuaPlus—my usual recommendation for programmers looking for a Lua distribution—comes with a remote debugger. However, it is finicky, unsupported, and has the huge drawback of not supporting multiple VMs.

Given the amount of script code often running in real world game projects, a good debugger is essential. For this purpose, although somewhat bare bones, Decoda certainly gets the job done. I would recommend Decoda to anyone who uses Lua without an existing debugging solution and who doesn't need a remote debugger for consoles. However, I look forward with great anticipation to a future version of Decoda with full support for consoles and better file management for large projects.

BRIAN A. STONE *has been writing code for games both big and small, consoles and PCs for over a decade. He is currently a development manager in Microsoft Games Studios and helped ship some of the top Xbox 360 titles of the past year. Email him at* ***bstone@gdmag.com.***

CONTINUED ON PG 48

DECODA 1.07

★★★⯪

STATS

Unknown Worlds Entertainment, Inc.
959 Haight St. #12
San Francisco, CA 94117
www.unknownworlds.com/decoda

PRICE:
Indie: $50
Commercial: $200

SYSTEM REQUIREMENTS:
Windows 2000
10 MB RAM
10 MB Hard Disk space.
Works with applications utilizing Lua 5.0 and later. Supports wide character strings in LuaPlus.

PROS:
1. Attaches easily to an application without requiring any modifications. Literally within minutes, I was stepping through the Lua code of each game project I tested.
2. Familiar interface for anyone used to MSVC. Basic functionality all there—breakpoints, stepping in and out, call stacks, and watch window.
3. Full support for multiple virtual machines.

CONS:
1. No remote debugger—will only be useful for debugging Lua code in Windows applications.
2. No ability to organize files in project window makes working with large projects cumbersome.
3. No extended debugging capabilities—no locals watch window, no data or conditional breakpoints.

CONTINUED FROM PG 47

product news

Learning Edition For PFTrack 4.1

Pixel Farm

Pixel Farm has released PFTrack Personal Learning Edition, a free version of its geometry tracking, image-based modeling, z-depth extraction, and multiple-motion solving application.

The learning edition allows students, digital artists, and facilities to evaluate, learn, and master the latest version of PFTrack without a permanent or temporary software license. It includes PFTrack's complete toolset, but limits the ability to export camera and image data. Projects created in the learning edition of PFTrack cannot be opened in the retail version, limiting it to non-commercial use.

http://thepixelfarm.co.uk

AlienBrain 8.1 Expands OS Support

Softimage

The latest free upgrade for the Alienbrain digital asset management system has expanded OS and application support, adding further integration with other content creation applications.

The latest Alienbrain update adds more support for Microsoft Windows Vista and Apple Mac OS X Leopard. It also adds digital content creation software integrations with Softimage's latest XSI, Microsoft Visual Studio 2008, Adobe Photoshop C3, Autodesk AliasStudio 2008, Autodesk 3ds Max 2008 and Autodesk Maya 2008, and allows for preview of DirectX files.

www.softimage.com

New Middleware Chat SDK For MMOs

ChatBlade

ChatBlade Middleware Chat SDK For MMOs is now shipping as royalty-free C++ source code, and includes HTML documentation and sample apps.

The SDK is Unicode compliant, enabling chat in languages including traditional Chinese, Korean, Swedish, and English, and supports features for moods, in-game languages, "drunken speech," and chat filtering.

Additionally, the middleware recognizes in-game "slash commands"—those commands recognized by typing a "/" before entering them—from various MMOs, allowing users to work with the chat commands to which they have previously become accustomed.

www.chatblade.com

Mental Images Releases Standalone Shader Tool

Mental Images

A full-featured, stand-alone beta version of Mental Mill Artist Edition has been released. Previously bundled with Nvidia's FX Composer 2, Mental Mill lets artists develop, test, and maintain shaders without previous programming experience.

Shaders created with Mental Mill are platform and environment agnostic, so they can be built once to run on any configuration and need only be created once before being exported to Cg FX, HLSL, GLSL, or other formats.

Mental Mill features real-time feedback to facilitate graphical debugging, shader optimization, and provides a hardware-rendered preview that shows nodes at any point in the shader graph.

www.mentalimages.com

Hansoft Adds C and .NET Support To Integration SDK

Hansoft

Support for C and .NET Support has been added to the Integration SDK Project management and bug tracking software. Developer Hansoft announced the latest update of its tool chain integration SDK, adding new support for C as well as NET, with a new C++ layer soon to be released.

The Hansoft Integration SDK can access all data and functionality on the Hansoft server to do real-time integrations within the studio tool chain, or with other systems, such as an external bug database.

The SDK also lets users analyze and manage their research usage, and features a Find All function that can search multiple projects by various criteria. It also enables teams to delegate planning rights and features support for task scheduling and agile project setups side by side. Process support templates can be created, version-controlled and reused.

www.hansoft.se

Autodesk Maya Adds Muscle Functionality

Autodesk

For the latest release of Maya 2008 Extension 1 software, developer Autodesk has incorporated Maya Muscle functionality into the 3D modeling, animation, visual effects, and rendering solution.

Maya Muscle, which Autodesk recently acquired from Comet Digital, allows artists to create skin motion through features that let them direct muscle and skin behavior. The toolset integrates with the Maya architecture and overall workflow, enabling it to be used in isolation, interconnected with other Maya features, or customized and scaled as needed.

Tools include advanced muscle and skin sculpting and deformation tools, as well as jiggle and weighting options like slide, sticky, and wrinkle weights. It also incorporates tools such as automatic rigs, real-time jiggle tweaking, and file caching, designed specifically for secondary character motion.

http://usa.autodesk.com

Unreal Engine 3 Gets Illuminate Labs' Beast System

Illuminate Labs and Epic Games

Illuminate Labs announced a licensing agreement to incorporate its global illumination system, Beast, into Epic's Unreal Engine 3.

Through the agreement, Illuminate Labs joins Epic's Integrated Partners Program, making its technology available to any publishers or developers that license UE3.

Beast enables the addition of global illumination to development environments using the Unreal Editor. It is a platform-independent baker that utilizes the open-standard FBX file format, allowing users to add baking functionality to 3D asset creation tools.

Beast features include global illumination baking, skylights from HDR images, and baked soft shadows with transparency. As part of the partnership, future feature sets developed by Illuminate Labs will be incorporated into UE3.

www.illuminatelabs.com

MICK WEST

THE INNER PRODUCT

COLLABORATIVE GAME EDITING

GAME DEVELOPMENT TEAMS ARE getting quite large. The largest teams can have more than a hundred developers working on a game at any one point in the process. Yet many developers still use work practices that revolve around individuals taking the entire responsibility for large chunks of the game. For many things, such as the creation of the 3D models and textures for individual game objects, this is still quite reasonable. However, for the creation of large game levels and continuous game worlds, this can create problems and it is worth investigating methods of content creation that are more collaborative in nature. This article takes a look at some of the issues involved, and discusses a few technical problems from the tool programmer's perspective.

CODE CONFLICTS AND BOTTLENECKS

When multiple people are working on overlapping aspects of a game, there are two problems that arise: edit conflicts and edit bottlenecks. These problems are two sides of the same coin. Fixing edit conflicts can create edit bottlenecks, and vice versa.

An edit conflict happens when two people edit the same thing at the same time and their changes create two conflicting versions. The nature of this problem depends on how fine-grained the "things" are in the game being developed. Programmers have always had this problem when two of them work on the same area of code at the same time. If both programmers make changes to the same file, an edit conflict occurs, and it needs to be resolved.

Generally, programmers will not be working on the exact same line of code, but rather will work on separate tasks, which use code that overlaps in various files. When a programmer wants to make a change to a file, he or she will check it out from the version control system (VCS), edit it until the changes are working, and then check it back into VCS.

The problem of code edit conflicts can be handled by allowing only one programmer to check out a particular file at one time (this is typically a setting in the VCS software). Since only one programmer can edit the code at a given time, there can never be an edit conflict. However, we still get bounced over to the other side of the problem—the edit bottleneck.

An edit bottleneck occurs when two people want to edit the same thing at the same time, yet person B cannot edit it because person A is editing it. With code, this is particularly problematic when large sections of functionality are incorporated in key files (typically with rather overarching names, like: "player.cpp" or "globals.cpp"), which affect many areas of the game and need many changes.

The common solution is to allow for multiple checkouts. Thus, more than one person is allowed to check out the same file at the same time. They make their changes, and then check the file back in. If the changes conflict, then they have to be merged. This step can usually be done automatically, but occasionally it requires a little manual intervention. Sometimes programmers end up editing the exact same piece of code and have to actually talk to each other to figure out a solution.

Some programmers prefer to structure things in such a way that multiple checkouts are never required. Theoretically, we can achieve this solution by making individual code files as small as is practical, splitting up files that contain multiple functionality, and establishing programmer procedures for rapid iterations, minimizing the length of time files can be checked out.

In practice a combination of efficient code division and multiple checkouts is commonly used. While edit conflicts can occasionally create problems, these can generally be mitigated by keeping a reasonable amount of functional separation in your code organization, and ensuring programmers check in (or merge) their code reasonably often. The problem of edit bottlenecks can stop programmers in their tracks, or at least seriously cramp their programming options, and the occasional problem caused by an edit conflict is well worth the extra flexibility that simultaneous editing can provide.

FROM CODE TO DATA

Traditionally, level editing is done using some kind of standalone tool that is not part of the game engine. The level designer loads the level, makes changes to it, and exports it to see how it plays in the game. This level-editing tool might be a commercial product, such as 3ds Max, or it might be an engine-specific tool that comes with a third-party engine, or it might even be a custom in-house tool.

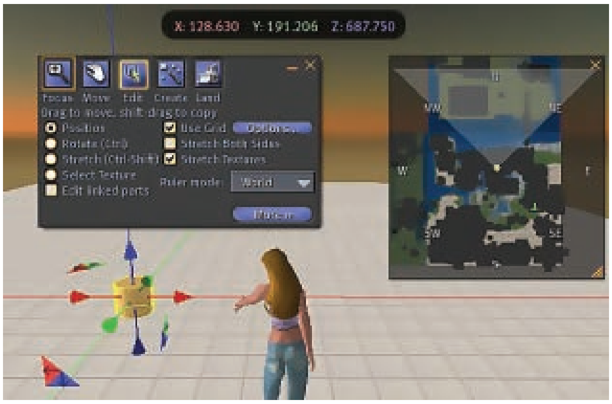
Level editors can be integrated with the game engine, allowing the designer to view and play the level as it's being edited. This is ideal from the point of view of rapid feedback, but the problems of edit conflicts and bottlenecks still remain. If two level designers want to edit the same part of the level, they usually will have to check out the entire level from the VCS.

With code, multiple checkouts of the same file are less problematic

MICK WEST was a co-founder of Neversoft Entertainment. He's been in the game industry for 17 years and currently works as a technical consultant. Email him at mwest@gdmag.com.

because it's relatively easy to merge text files automatically, as the changes are usually on well separated lines. Conflicts that cannot be resolved automatically are usually taken care of very easily manually, as the nature of the conflicting changes is readily apparent to the programmer who makes the merge.

Unfortunately, this solution does not work with level data, which is often stored in a binary format that's impossible to merge, especially if the level editing tool is some thing like 3ds Max. Even if the level is stored in a text-based format, such as XML, it's much harder to merge as there are many internal dependencies and automatically generated data that create a broader mesh of changes that are difficult for a human to read. What we get are more conflicts, which then are much harder to resolve without breaking something.



Objects in Second Life are edited in real time.

SPLITTING DATA

The difficulty in merging changes in level or world files would inevitably lead to bottlenecks if no steps were taken to mitigate it. Various solutions have arisen to handle the problem. The simplest is to break the level down into the smallest chunks possible, so that individual level designers can check out only the sections of the level they need. Using small chunks improves matters by reducing edit bottlenecks, but adds complication in how the level is split up and subsequently pieced together. Implementing high-level changes to large sections of a level also becomes difficult. Though bottlenecks are reduced, they're not eliminated, as there will still be merge work needed at the borders of the areas assigned to individual level designers. Edits here must still be manually coordinated. The large number of individual parts that make up the divided level now places an undue organizational burden on the level designer.

Sometimes more ad-hoc solutions are used. A level might still be stored in one large file, but when two people need to work on it at the same time, it is manually split into two sections. The work is done (with some discussion to avoid conflicts), and then the two sections are visually merged back together. Sometimes, special merge tools are written just for this purpose. This technique is fraught with problems, as the merging process is rarely simple, and can take some time.

Another problem is that using the VCS is inherently a manual process. When a designer is working on a level, she'll check it out, perform her work, and eventually check it back in. During that time, no one else can actually work on it. But the designer who has the level checked out is unlikely to be working on it every minute she has it checked out. There might be long periods of time when she's working on something related. If the designer has to keep track of all the pieces of the level she might want to edit, then she might be tempted to check out large chunks, just to make things easier. In other words, designers will start checking out sections they are not in fact editing.

If instead the check-in and check-out procedures are handled automatically, then in theory you could set up your tools so that as soon as the level designer started to edit a particular piece of the level, then that section would be checked out from the VCS; and as soon as the designer stops editing (for example, the file is saved or successfully viewed in the game), then that piece is checked in.

Various problems obviously arise from using this automated method. Most notably, you're going to end up with people checking in things that are broken, so perhaps it's better to still rely on a manual commit at some point. Still, checking out sections automatically can work well if the level is fine grained enough that conflicts are minimized. The level designer need not specifically be aware of the parts of the level that he has checked out—simply that he's made a set of changes and needs to check them in. The tools will keep track of what has been changed, allowing him to simply check in everything with a single click.

THE IDEAL PROCESS

In designing any toolset, it's useful to imagine the most desirable possible end result, regardless of how practical it seems. Such blue-sky designing can uncover possibilities that might not be considered with a more traditional incremental approach to adding features to an existing tool or process.

With our level-editing problem, we have a two sided problem: edit conflicts and edit bottlenecks. Fixing one problem can mean making the other problem worse, so we end up simply balancing them as best as possible, accepting the inevitable manual resolutions and underutilized resources that come from this.

But in an ideal world, there would be neither conflicts nor bottlenecks. Anyone would be able to edit in any area of the game at any time; nobody would have to wait to start working, and all changes would be merged automatically. In addition, all changes would be visible immediately in the game as they're made, with no delay and no "exporting" of files.

Sure, that would be great, but can we get there? If we can't, then how far can we get in that direction?

SECOND LIFE

In some respects, the online virtual world Second Life is already there. Players in the game exist in a large continuous world, which they can edit in real time. Multiple people can edit in the same area of the world at the same time, and their

YOU'VE LED US ON THE BATTLEFIELDS OF GETTYSBURG,

PILOTED PIRATE SHIPS ON THE HIGH SEAS,

FLOWN US TO THE STARS, AND INSPIRED CIVILIZATIONS.

CONGRATULATIONS, SID, ON YOUR LIFETIME ACHIEVEMENT AWARD.

FROM YOUR FRIENDS AT 2K GAMES

CMP
United Business Media

GDC returns to San Francisco in 2009

2009

GameDevelopers®
Conference

March 23 – 27, 2009
Moscone Center, San Francisco
www.GDConf.com

Visit **www.GDConf.com** for more information and GDC08 proceedings

Call for submissions
opens late summer 2008.

changes are visible instantly, both to themselves and to others. They can edit 3D models and scripts simultaneously, editing, testing, and playing all at the same time in a truly collaborative editing environment.

That does sound somewhat like our ideal, but if you've ever edited something in SECOND LIFE, you'll see it's not quite perfect. The editing process is based on you editing objects that exist in a remote database. But working over the internet introduces some annoying delays. The tools themselves are rather primitive, which is to be expected. But the most glaring omission from a game developer's perspective is the lack of version control. When you edit something, that's it. You've edited it. No rolling back to the version from yesterday (or five minutes ago) if you accidentally delete half your script or mess up your UV coordinates.

> **"The problem of edit bottlenecks can stop programmers in their tracks, or at least seriously cramp their programming options."**

VERSION UNCONTROL?

The problem of not having version control in an editing environment such as SECOND LIFE could obviously be addressed by incorporating some kind of VCS directly into the project. But this again raises the issues we had earlier of potential bottlenecks: how to divide things into files, and the manual burden of checking things out and back in again.

Take a step back and ask the sacrilegious question: Do we need version control?

The immediate and emphatic answer is, "Of course," especially if you remember the dark days of early game development when all code was manually merged on Fridays, nobody had the right version of anything, and programmers lost days of work if they accidentally overwrote a file. Version control is an invaluable part of the game development process that is impossible to discard.

But consider what you use version control for, especially with game assets. You use it to prevent edit conflicts, to distribute the latest versions, to back up files, and to create branched versions of the game.

Now on a daily basis, the vast majority of what a VCS is used for is the first two; preventing edit conflicts and distributing the latest version. Developers check out or "open for edit" some files, do their work, and then check in or "submit" those files. They then "get" or "sync" the latest version of the project and tools. The backup aspect is a necessary part of version control but is used relatively infrequently. The branching is used even more infrequently and is a high-level task performed by very few team members.

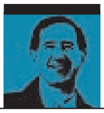
Consider now how this works in SECOND LIFE. We don't need to ever get the latest version, as we always have the latest version. We are in the latest version. We don't need to check things in or out, as we simply start editing them, and they get locked. Then we stop editing them, and they are unlocked. We don't actually need the conflict prevention and version distribution of a VCS.

But what about backups? You'll note that backups are what you really miss about the lack of a VCS in SECOND LIFE—and remember, the backup functionality is one of the least used functions in VCSs. We still obviously need it, but we might now want to perform a little paradigm split and separate the VCS into two separate function groups, one being the file ownership and distribution system and the other the file backup and history system.

Since they have been so closely linked in the past, it has been natural to treat them as being the same thing. When you check in a version of a file, it becomes the file that gets backed up. Yet the real utility of a backup system would be the ability to undo your changes to an arbitrary point. If you've been editing a file for a while without checking it in, then you might have made a lot of changes that you might want to undo. Designers often save multiple numbered iterations of a file locally before checking in. Programmers who keep a file checked out for a long time frequently get nervous and make local backups, or even check-in with changes (hopefully) stubbed out, just to have the security of that VCS backup. Here the act of checking in a file is a useful label in the backup process, but it should not be the sole method of backing up a file.

Edit conflicts, bottlenecks, file distribution, and backups are, to some degree, historical problems, forced upon us by the constraints of a simple file-based editing system. Files are large, which means you get edit conflicts or bottlenecks, and that it's expensive to back up every single change. But if we transition to a game editing process more like that found in SECOND LIFE, we are no longer editing files, but are editing much smaller objects within the world, which are stored on a shared database which needs no distribution. Here it would be quite possible to back up every single change and every single edit to each object, essentially giving the developers both infinite "undo" capability and the confidence to make bold changes knowing they can safely return. Branching is also quite possible within such a scheme.

I think it is inevitable that some aspects of game development will move to using this collaborative model. It will require a rethinking of the way version control is handled, and even the way version control is viewed—a perhaps violent division of the functionality of file locking, distribution, and backups, that will allow each separate area of functionality to develop unfettered with new and powerful tools and processes. ⁙



STEVE THEODORE

THE FREE TOTE BAG GIG

Why and How to Speak at a Professional Event

> A good talk combines technical or artistic command of a given subject with an understanding of production realities.

IT'S GDC TIME AGAIN, AND THERE'S A good chance that you're even reading this somewhere in the noisy halls of the Moscone Center. If you're not in San Francisco right now, allow me to make a shameless pitch on behalf of the show. (Disclosure: I hold a seat on the GDC advisory board, so it's a given that I think of the show as a good thing; in addition, the company that owns the GDC is the same one that owns *Game Developer*).

GDC is important because it is one of the few institutions devoted to spreading knowledge around instead of hoarding it. Older businesses have professional schools and academic wings that help keep them vibrant, and although the game industry is gradually evolving to have these sorts of things as well, on the whole we still have to do the spade work ourselves.

SPEAKER PRESTIGE

Being a GDC speaker offers a nice boost to your resume. It gives you some visibility among your peers and gives you a chance to demonstrate your chops in front of an audience that's likely to contain possible employers who are looking for skills like yours. If all that weren't enough, speaking at GDC earns you a free conference pass, a tote bag, and the chance to bump into Will Wright in the speakers' lounge. Not surprisingly, a lot of folks would like to win a coveted speaker's badge.

It's a good thing that the rewards are so steep (did I mention the tote bag?), because putting together a good talk is a serious undertaking. It requires rigorous planning; talk proposals are submitted six months or more before the show. It takes a lot of preparation: creating a slide deck, putting together example art, and doing enough practice runs to be a confident presenter. And it's highly competitive. Only a fraction of the talks that are submitted to the conference web site are approved for the show.

It helps to understand how the evaluation process works. Proposals are submitted to the GDC via the conference web site at GDConf.com, usually over the summer of the preceding year. The proposals are vetted by an advisory board of developers. Conference management recruits advisors from a number of studios for the various "tracks" or disciplines. The advisory board looks at all the proposals and ranks them on the appeal of the topic, the quality of the proposed treatment, and the track record of the speaker.

The board is also responsible for filtering out a large number of talk proposals that are basically sales pitches or user-group style sessions. While there's nothing inherently wrong with a talk on your fabulous new motion-capture camera or your amazing new paint package, the conference tries to steer clear of potential conflicts of interest by routing these through a separate process. You'll see these sorts of talks in the show catalog as paid-for sponsored sessions, as in "Sponsored by Autodesk." If you're thinking about a topic that involves a lot of very particular attention to one product, service, or vendor, you might want to get in touch with the conference staff before submitting your proposal in order to get a little more guidance on the rules for distinguishing between the regular conference sessions and the sponsored variety.

KEYS TO A GOOD TALK

When you start to develop your topic, you don't need to be too academic. That's how the programmers amuse themselves. GDC talks aren't classroom exercises. They are a chance to share practical knowledge. The unique value of the show is that it's oriented around a professional community. You can pick

STEVE THEODORE *has worked a modeler, animator, and technical artist for more than a dozen years and has spoken at GDC every year since 2003. His credits include* MECH COMMANDER, HALF-LIFE, TEAM FORTRESS, *and* COUNTER-STRIKE. *He's currently content-side technical director at Bungie Studios. Email him at* ***stheodore@gdmag.com***.

®
VICIOUS ENGINE
making games. easier. faster. better.

up the finer points of ZBrush at your local branch of Gnomon, and you can get a lot more help for your key framing skills from a few months of Animation Mentor than you'll ever get from a 50-minute lecture.

What you can't get anywhere else is the combination of technical information and rubber-meets-road experience that comes from other devs, so a good talk combines technical or artistic command of a given subject with an understanding of production realities.

THE TAKEAWAY

The first thing the advisory boards look for in a proposal is the "takeaway"—the key nugget of wisdom that the audience will (hopefully) get from the presentation. From a selfish standpoint, snappy takeaways are important because those are what you show your boss when trying to wrangle yourself a junket to the show. Good takeaways make for more conference goers.

> "Just as in high school English class, coming up with a strong outline is the key to success both in the submissions process and in crafting a popular talk."

More importantly, though, the takeaway is the acid test of what you're trying to say in your talk. If you can't figure out the two- or three-sentence distillation of what you want to get across, the audience and the reviewers won't either.

A good takeaway is concise and straightforward, for example: "Attendees will learn the pros and cons of the major normal mapping techniques, with particular attention to choosing the right technique for your game genre." A bad example might read something like this: "Attendees will learn the coolest normal mapping tricks EVAR!"

SCOPE

As I've said before, the unique value of a show like GDC is the combination of technical and artistic knowledge with real-world experience. You can learn as much, or more, from your presenters' accounts of hassles and failures as from the theory they are trying to explicate. Thus this year's talk on "The Illustrative World of Team Fortress 2" could teach a lot about dealing with a multiplayer-only title or working with the Source engine material editor, as well as offering some general wisdom about stylized character art.

The best talks teach general principles using real-world production as vivid examples, striking a balance between simple postmortems on the one hand and pure theory on the other.

Trying to go very deep is tough in the typical hour-long lecture format. If you focus too closely on the precise specifics of a particular technique or technology you'll probably lose many of your listeners in the thickets of details. And if you can't provide a larger takeaway, you won't be able to reach audience members who work in other genres or on other platforms. A talk about how to use non-linear animation tools to crank out a lot of animations for multiplayer games, using your online shooter as a case in point is going to work a lot better than a talk about the six months you spent getting everybody to learn the trax editor in Maya.

However, overly broad talks are weak, too. It's a notorious truth, for example, that every year a large number of talks appear about art direction. Most of these are good proposals from serious professionals, but the more broadly these talks are pitched, the more they tend to overlap, so inevitably a large number of proposals is whittled down to only one or two selected talks. There's only so much room in the program for "Grand Unified Theories of Art," no matter how worthwhile. On the other hand, a talk about creating art direction for international products, with special reference to porting a popular Korean game for the U.S., a primer on doing historical costume research as shown in a Renaissance-themed adventure game, and a third talk about the special challenges of creating a visual direction for DS games can all coexist happily.

CONTINUED ON PG 60

CONTINUED ON PG 60

FOCUS

Working in an esoteric business like ours can be lonely. Every veteran game artist has a closet full of favorite tactics and war stories to share, and performing before an audience full of the only people in the world who have any idea what you're talking about is a standing temptation to cram every tip and trick of your career into your PowerPoint presentation. Unfortunately, that grab-bag approach may make good fodder for a bull session on the suite party circuit (or maybe not, as many GDC after-party vets can attest), but it's murder for a GDC talk.

Verbal presentation is a much less efficient medium than print or the web. To reach an audience effectively, you need to make sure your points are clearly marshaled and mutually reinforcing. Your talk outline needs to be a clearly structured, well thought-out presentation of your points and not a laundry list. Talk proposals with good topics and interesting speakers often fail to make the grade if they come with fuzzy or incomplete outlines. Just as in high school English class, coming up with a strong outline is the key to success both in the submissions process and in crafting a popular talk. Fortunately, a good takeaway and strong hook will both nudge the proposal towards a clear and well-focused structure.

PHOTO BY VINCENT DIAMANTE

CASTLEVANIA series producer Koji Igarashi at GDC 2007.

CREDENTIALS

If you haven't given a GDC talk before, landing a speaking gig is challenging. Newcomers confront the same catch-22 that maddens first-time job seekers: without any experience, it's hard to convince people to let you earn experience.

Audiences and reviewers are attracted to well-known names and high-profile titles. If you've had the good luck to work on a high-profile franchise—or better yet, a big hit from last year—your proposal will have a leg up over an equally good submission from an eager but obscure competitor. This probably means some good talks don't get accepted, but it does reflect the preferences of the audience.

Naturally, if you've given a successful talk at GDC before, it's much easier to get another chance. The conference organizers carefully track audience responses, so if you've previously given a talk that was unimpressive, you might not be at such an advantage. The ratings are done on a five-point scale; if a previous session averaged three or lower, it'll be pretty tough to win a second chance.

If your resume isn't studded with million sellers, you should make sure to burnish up any other credentials you may have. Presentations at other industry shows certainly help, as does experience as a teacher or writer. If you fear you're at a disadvantage in the credentials department, you should think about developing your presentation skills and name recognition. A semester or two teaching at your local art school or community college game design program can be helpful, as can a guest spot in *Game Developer* or other industry publications and web sites. In any event, if you're worried about breaking into the hallowed ranks of GDC presenters, you have all the more incentive to really hone your proposal into an irresistible pitch.

WHY BOTHER?

If this sounds like a lot of work, it is. Landing a speakers badge at GDC is an accomplishment in itself. You'll need a well crafted topic, some hard won personal expertise, and the willingness to get up in front of an audience of smart and skeptical peers who won't hesitate to call BS. Is it really worth it?

Well, the practical benefits for your career are obvious. If you make a name for yourself as a presenter, you'll have an advantage in future job searches. As a side benefit, you'll become a Jedi master of whatever subject you set out to speak on. No matter how well you knew it when you clicked the "send" button on your proposal, a few months of slide-crafting and run-throughs will force you to know the subject far better than you thought possible.

Even with all purely personal benefits, pinning on the lav mic and facing the crowd is also a real service to the community. Older professions have a pretty good sense of what it means to be a member of the club. If you were a lawyer or an accountant, you would have a pretty clear idea of how you can expect your career to go. In our young (not to say "infantile") business, those expectations are much more fluid and are still evolving. Stepping into the spotlight for a few minutes is helping to define those expectations for yourself and your peers. What does it mean to be a "game artist?" You're helping us all to figure it out. If that comes with a tote bag, so much the better. ⁙

Austin Game Developers Conference

SAVE THE DATE!

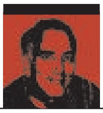
Austin Game Developers Conference Returns

September 15-17, 2008
Austin Convention Center, Austin TX

Call for Submissions Opens Mid - March 2008

Visit **AustinGDC.net** for details and to download 2007 proceedings

CMP
United Business Media

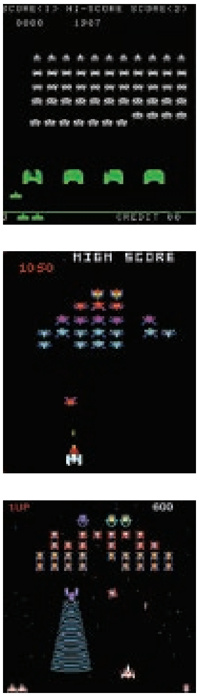


NOAH FALSTEIN

GAME SHUI

MULTIPLE VARIATIONS ON A SINGLE THEME

Taking the Fifth



Variations on a theme: SPACE INVADERS, GALAXIAN, and GALAGA.

I OFTEN LISTEN TO MUSIC WHEN I'M doing creative work. When I have a tough game design challenge, one favorite piece is Beethoven's Fifth—not his *Symphony No. 5*, but his *Piano Concerto No. 5*. One thing many classical pieces have in common is the form of stating a simple theme and then playing with it in multiple variations.

Many good games do that too. Casual games in particular often take a basic concept like match-three or finding words or moving blocks, and then present level after level that gradually leads the player through dozens of variations. The best of these use a few simple building blocks of game elements and find dozens or even hundreds of intriguing ways to recombine them and make them seem fresh.

Last month I praised PORTAL for doing just that. That game is masterful in the way it gradually and elegantly introduces just a few new elements, from actions you can take, to lock-and-key puzzles, to hazards, and then lets you get used to them slowly before escalating. It amuses me that people still argue about whether video games are art when there are examples as sublime as PORTAL around.

LEARNING FROM HISTORY

But the idea of theme and variations is not new even to video games. Classic arcade game themes sometimes took a few iterations to refine the elements and get the balance just right. SPACE INVADERS established the idea of an array of aliens moving down a screen, dropping bombs on a player ship that could only move side-to-side and shoot up. GALAXIAN refined it, making the aliens move more freely in interesting patterns. And then GALAGA got the balance just about right, and now, 26 years after its release, I still find that game in movie theater lobbies, where people avidly play it.

Each level is of course a variation on a basic theme with quite simple and elegant improvisations layered on.

Learning is key. This very principle of establishing a basic task for the player to master and then presenting one variation after another with gradually increasing difficulty is at the heart of nearly every video game. It's at the heart of most effective pedagogy, too. James Gee's book, *What Videogames Have to Teach Us About Learning and Literacy*, systematically analyzes just how efficiently games use proven teaching methodology. In the commentary section of PORTAL (have I mentioned how much I liked PORTAL?) I was intrigued to note that the vast majority of the comments from the developers referenced modifying the game to make the learning curve more accessible.

DO IT YOURSELF

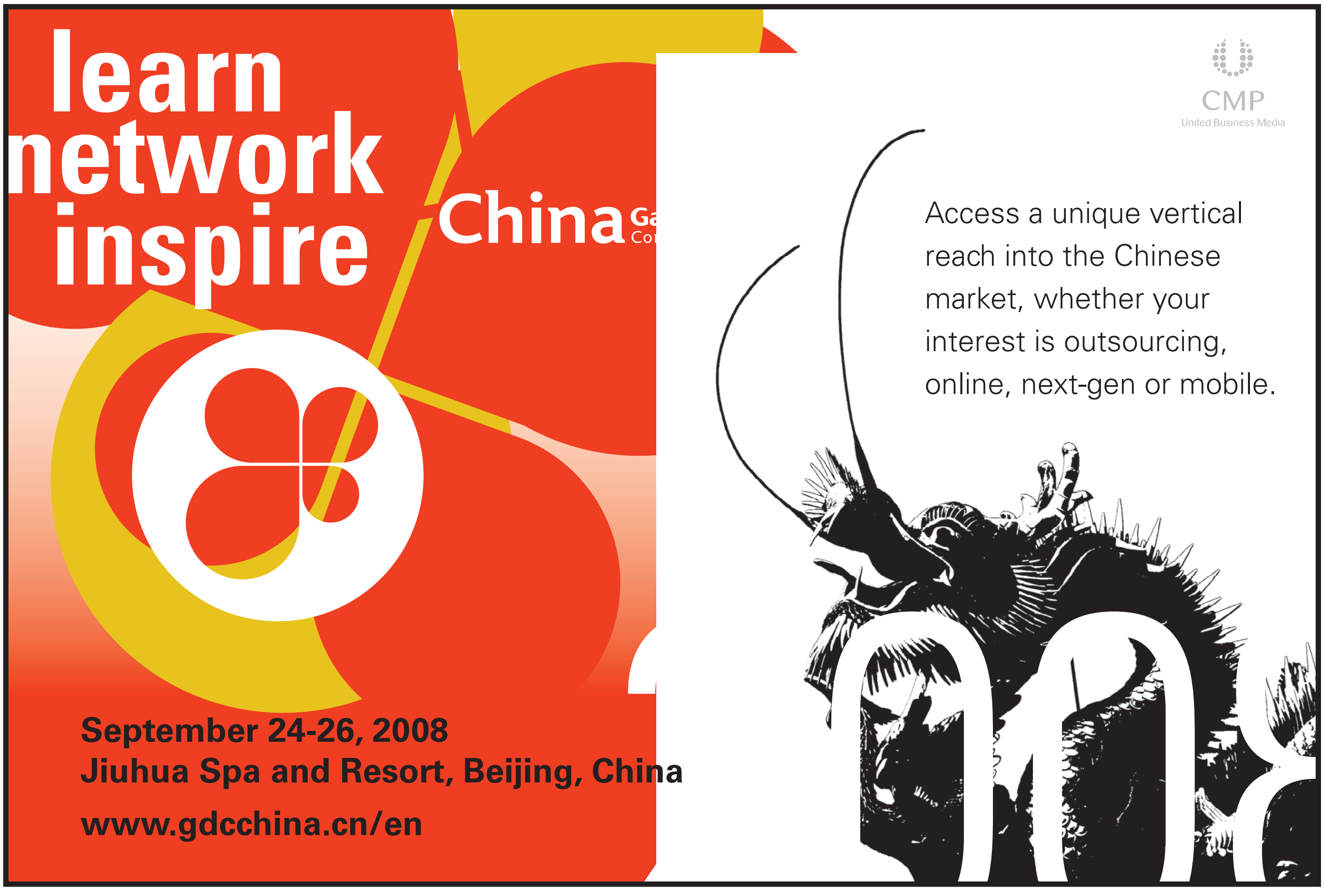
How can you know when you've found the right combination of player actions and abilities, enemies and challenges, and story or theme? One sign is in the brainstorming process. If you have to labor for hours to find new variations, or if every variation you come up with demands new use-once game dynamics or unique inventory items or specialized enemies, you're not there yet. When the ideas start coming faster than you can write them down, and reuse existing actions, items, and obstacles in novel combinations, you're on the right track. During production, you should be prepared to go through a series of expansions and contractions of those different game elements as well. As new levels or challenges are introduced, they may at first seem to require totally new weapons or tools or enemies—but keep looking for ways to simplify, and soon, if you're on the right track, you'll also find ways to combine and streamline until you just can't trim any more. That's another sign you're getting close. But ultimately, you also need to confirm your team's judgment by bringing in fresh eyes to try your game.

GREATER GIFT GIVING THROUGH GAMING

In the theme of refining a simple game, I'd like to conclude with a reader design challenge. I'm writing this in December when those holiday gift exchange parties are popular. Everyone brings a wrapped gift and draws a number from a hat, then in order each person either picks an unopened gift or takes one that someone already opened (who then picks a new one).

When I've played this gift exchange game at development studios, the game designers always propose variations and limits to make the process more fair or at least more fun and to prevent infinite loops. I'm sure many of you have your own ideas about this. Rather than suggest my own, I'd like to hear yours, while the gift-giving experience is still fresh. Email me with your suggestions and I'll publish the best.

NOAH FALSTEIN *has been a professional game developer since 1980. His web site, www.theinspiracy.com, has a description of The 400 Project, the basis for these columns. Also at that site is a list of the game design rules collected so far and tips on how to use them. Email him at* ***nfalstein@gdmag.com***.



JESSE HARLIN

BEYOND THE BUTTON PRESS

THE PALETTE FOR USER INPUT WITHIN games grows every year. From drum sets to styluses, nunchuks, bongo controllers and beyond, the past few years have seen an explosion of bundled hardware geared toward changing how players interact with virtual worlds. Many of these technologies are one-off gimmicks. Others, however, have become so ubiquitous as to become either standardized accessories to particular genres or console manufacturer-mandated pieces of hardware.

While gameplay designers continue to utilize these new avenues for player interaction, audio designers are largely stuck in the decades-old mindset that players make noise in-game by pressing buttons on their controllers. Quite to the contrary, never before in our industry's history have audio designers had more opportunity for creative interaction with players through hardware and software that is readily available and simply waiting for them to utilize.

TELL ME MORE

Sit down with THE LEGEND OF ZELDA: THE PHANTOM HOURGLASS and it won't be long before gameplay mandates that you yell at your DS. Surprisingly, this kind of user-created audio input is something that the DS does extremely well, which anyone who has taught their Nintendog to "roll over" or "sit" can attest to. At the heart of the gameplay mechanic is the DS's internal microphone. Yet, despite being available for use in every DS game by the very nature of its hardware, very few titles incorporate the microphone as a means of interactivity.

Similarly, first person shooters and networked gameplay over PCs and Xbox Live have made headset microphones a must-have for online multiplayer. However, utilizing these nearly-omnipresent audio input devices as part of the single player experience is extremely rare. A handful of previous games have flirted with voice recognition and speech interaction, such as Konami's LIFELINE and various tactical shooters like SWAT: GLOBAL STRIKE TEAM, but few AAA games have managed to make player voice interaction a compelling part of their game's soundtrack.

The gameplay potential for this kind of audio interaction is huge, however. Voice print identification and encryption can find its way into espionage games. Imagine calling plays by physically *calling plays* within sports games. Consider the gameplay potential in using the range of human speech from whispering to yelling in order to frighten or coerce enemies within platform games like JAK & DAXTER or LOCO ROCO. As audio designers, we can be doing more to encourage user-generated audio input as part of our design.

WHAT'S IN STORE

Over a decade of CD- and DVD-based console development has largely created an environment in which audio designers see their work as relegated to the inflexible confines of a locked disc. However, hard drives and memory devices are now either standard issue or readily available for every gaming platform, though some are more practical than others. Every PC and PS3 game has access to a built-in hard drive. While PC gamers are used to patches and publishers that push new content into their games, the concept is relatively new to console games. Each major gaming console has its own online store with a variety of demos, games, and additional content such as added levels and extra songs for music games.

The potential for broadening the sound experience of a game after it's been purchased is broader than updated playlists or a handful of new voice lines. Imagine an adventure game in which the most powerful weapons or treasure were hidden in-game and the only clues to their whereabouts were garbled pirate radio transmissions that could only be purchased and implemented via downloadable content. Imagine edutainment games for children with continually expandable vocabulary packs or a music game like MTV MUSIC GENERATOR that allowed for uploadable and downloadable user-created collaboration. The shipped disc isn't the end of the game anymore and audio designers should be considering the gameplay potential of hard drives, storage devices, and online delivery channels.

LISTEN TO THIS

Speaking of music, with the Xbox 360, Microsoft now mandates that any music occurring during interactive gameplay be replaceable with the user's own local MP3 collection. The text of the actual Microsoft technical certification requirement (TCR) says little more than that. In practice, however, the Custom Soundtrack option has proven to be clunky and artless in its implementation. This isn't, however, in any way a reflection on the requirement itself. Nothing in the wording of the TCR says that voice ducking or in-game DSP filtering cannot affect the user's music. Theoretically, a game could read the metadata of a user's mp3s and reassign slower ambient music and faster rock or hip-hop tracks accordingly so as to react with an interactive music engine.

Additionally, the Xbox 360 isn't the only platform with the potential for custom soundtracks. PCs, the PS3, and the PSP all contain the necessary hardware to access a ready bank of user-selected music files. Surprisingly, even Apple's MP3-playing iPod—which recently began marketing itself as a gaming platform—lacks a custom soundtrack option in the majority of its titles.

Audio design potential has extended beyond the confines of disc burns and button presses. As audio designers, it's up to us to make sure that our creativity keeps pace. ⁙

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**.

WE ARE HIRING!

Located in France, **EDEN GAMES** is creating innovative games
(see our references on: www.edengames.com).

For the development of consequent games on next-gen consoles, we are currently looking for talented:

**Game/Level Designers, Senior Artists, Next-Gen Console Engineers,
Lead Sound Designer, Senior/Executive Producers...**

If you share our passion, come join the award winning team of **Test Drive Unlimited** and **Alone In The Dark** to work in a dynamic, exciting and creative environment using high technology and making AAA games.

To apply for any challenging position, please send us your résumé and work examples to:

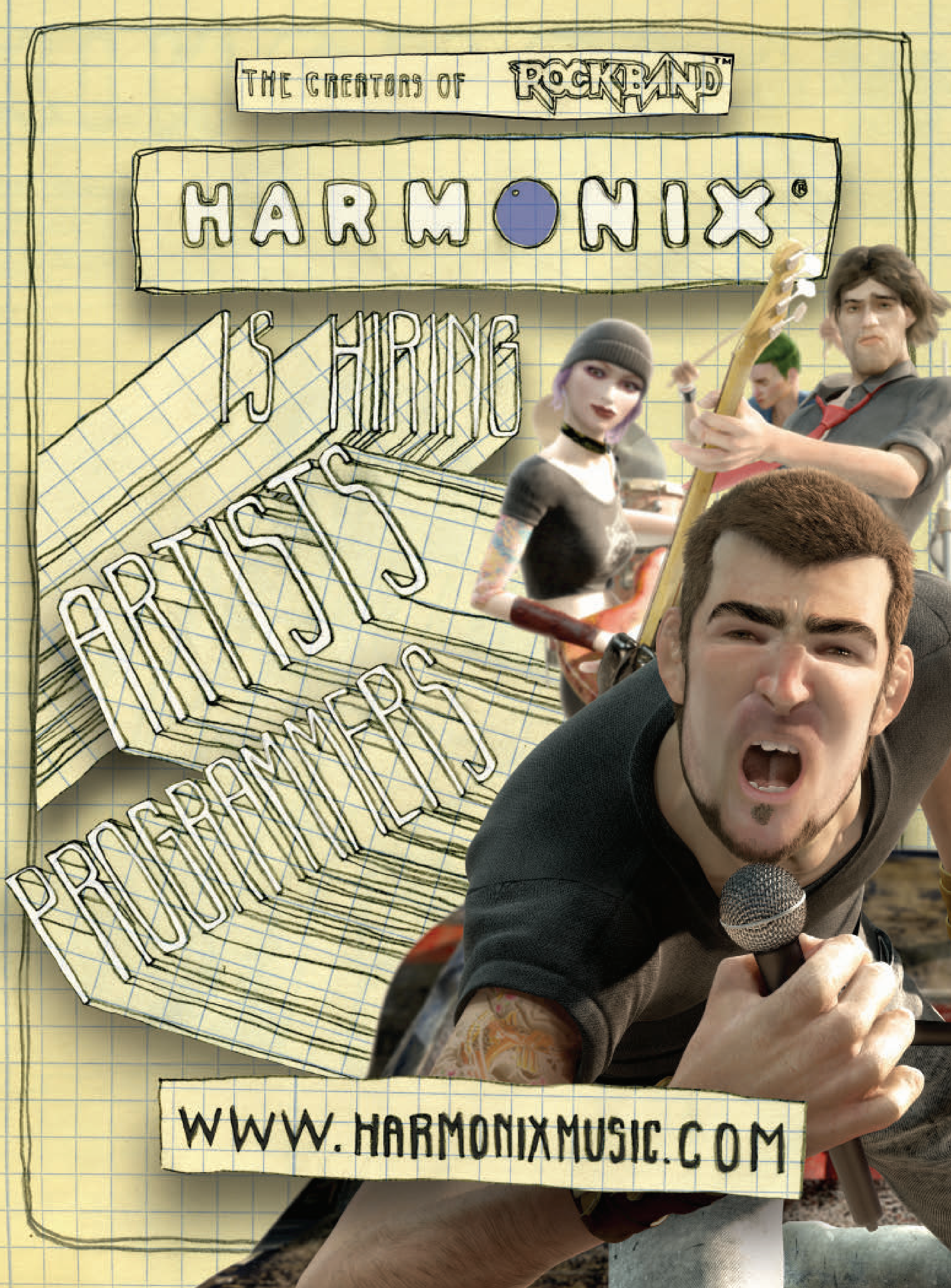
job@edengames.com

CUSTOM SLIDE
EL MONTE CA
AUTOMATIC
CALIBER .45

GUITAR HERO III
LEGENDS of ROCK
NEVERSOFT
WWW.NEVERSOFT.COM

GAME CAREER GUIDE
FEATURED SCHOOLS
GUILDHALL
Student Day in the Life:
Savannah College's Justin Popper
GETTING STARTED
INDUSTRY NEWS
FEATURES
Serious Games Summit GDC
CALL FOR SUBMISSIONS

Are you
GAME
for
Singapore?


THE CREATORS OF ROCKBAND™
HARMONIX®
IS HIRING
ARTISTS
PROGRAMMERS
WWW.HARMONIXMUSIC.COM

MAX YOUR EDGE WORKING AT WMS

- Leading the creation, design and manufacture of video and reel-spinning slot machines
- Extraordinarily successful, well-developed product lineup.
- Innovative designs and games that will lead the gaming industry into the future.
- Use the latest technology
- Join a company experiencing exponential growth
- Fun work environment allowing you to continually learn and grow

NOW HIRING

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

WMS™ MAX YOUR EDGE

COME VISIT OUR BOOTH CP 330

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

Got Game? Prove it!

Take your game career to the next level.

Nelson Staffing's Video Games Group is looking for energetic, creative gamers who want to play a role in developing next generation products. Don't miss your chance to be part of a unique group working in a fun, collaborative environment!

Nelson's Video Games Group provides game experts and enthusiasts to some of the most well-known names in the computer entertainment industry.

Hiring Now!

Associate Producers
2D & 3D Character/
Concept Artists
3D Animators
Environment Artists
Game Designers
Game Testers
J2ME/BREW Engineers
Level Designers
Product Marketing Managers
Senior Game Programmers

To find out more or to submit your resume or portfolio email **ralmeda@nelsonstaffing.com** or **mphillips@nelsonstaffing.com**, or visit us online at **www.nelsonjobs.com**.

NELSON STAFFING
Full Service Recruiting Solutions
Video Games Group

Great Jobs with Great Companies for Great People! **www.nelsonstaffing.com**

Join the PLAYSTATION® Team!

Sony Computer Entertainment America, Inc. (SCEA) markets the PlayStation family of products and develops, publishes, markets, and distributes software for the **PlayStation Portable**®, **PlayStation**®**2** and **PlayStation**®**3** computer entertainment systems, for the North American market.

We are currently looking for talented and experienced game development professionals for the following areas:

Foster City • California

Software Engineer
Product Marketing
Technical Manager
Programmer
Producer

San Diego • California

Web Developer
Software Engineer
Animator
Technical Support

Santa Monica • California

Designer

At **Sony Computer Entertainment America, Inc.**, we offer our employees exceptional benefits, an open and creative working environment, and the opportunity to be apart of a team that performs at **1.8 trillion calculations per second.**

For additional information, ability to send your resume, add updates, and establish a new business relationship, please visit our website at:

www.us.playstation.com/jobs

PLAYSTATION® 3

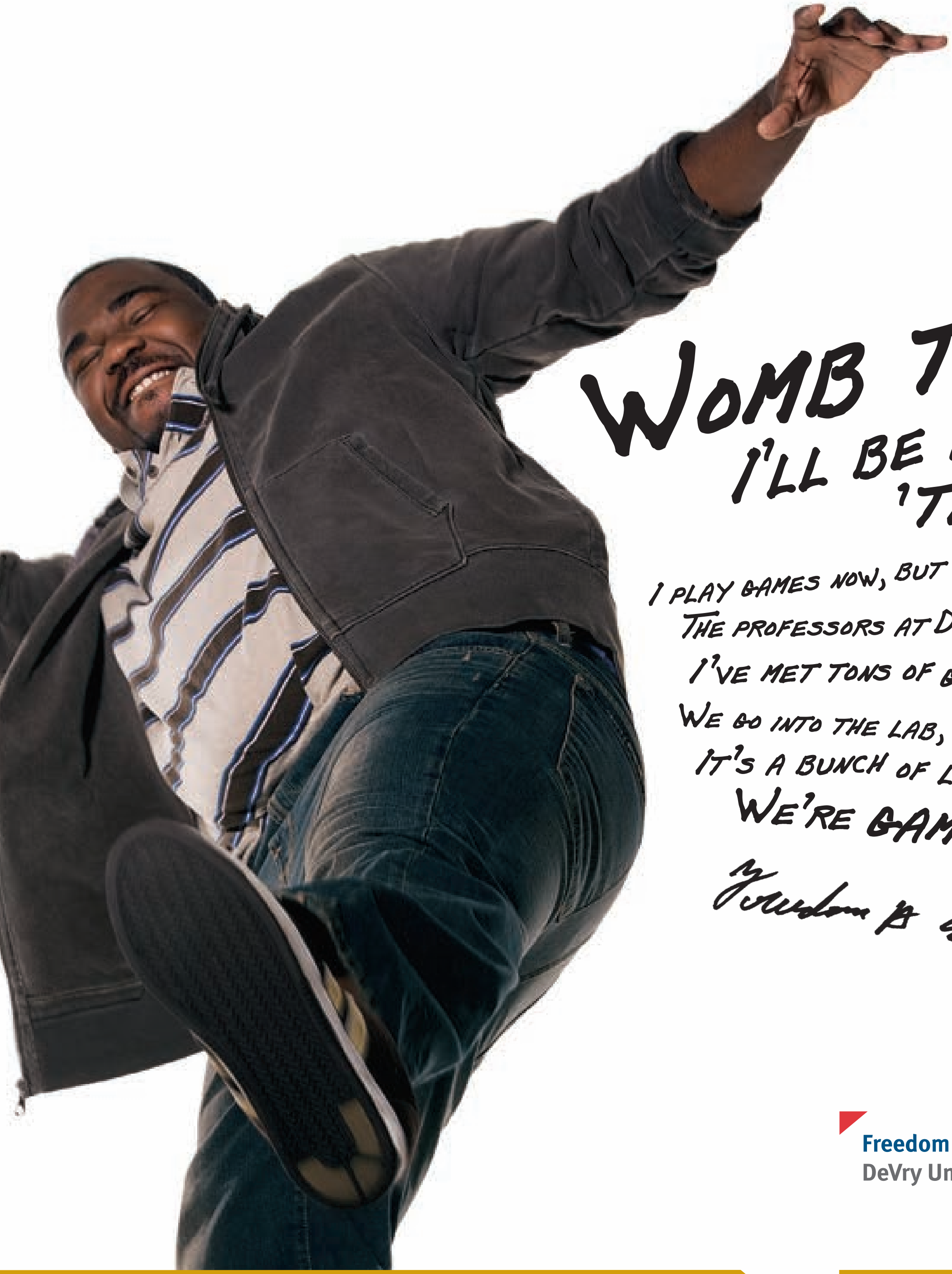
PlayStation®2

PSP®
PlayStation Portable

SONY
COMPUTER ENTERTAINMENT ®

"PlayStation" and the "PS" Family logo are registered trademarks of Sony Computer Entertainment Inc.
The Sony Computer Entertainment logo is a registered trademark of Sony Corporation. Equal Opportunity Employer

Discover your future
Wanted
Programmers
Artists
Designers
Producers
www.eidosinteractive.com



WOMB TO TOMB,
I'LL BE A GAMER
'TIL I DIE.

I PLAY GAMES NOW, BUT ONE DAY I'LL MAKE THEM.
THE PROFESSORS AT DEVRY ARE VERY IN-THE-KNOW.
I'VE MET TONS OF GAMERS AT DEVRY.
WE GO INTO THE LAB, TALK ABOUT OUR IDEAS.
IT'S A BUNCH OF LIKE-MINDED PEOPLE.
WE'RE GAMERS FOR LIFE.

Freedom A. Evans
DeVry University '09

Since 1975, 227,605 undergraduate students system-wide have graduated from DeVry and 90% of those in the active job market were employed in career-related positions within six months of graduation. Classes start soon, so find out about the DeVry program that is helping Freedom find his game plan for success.

Call 1.866.529.1357 or log on to devry2.com/gdc

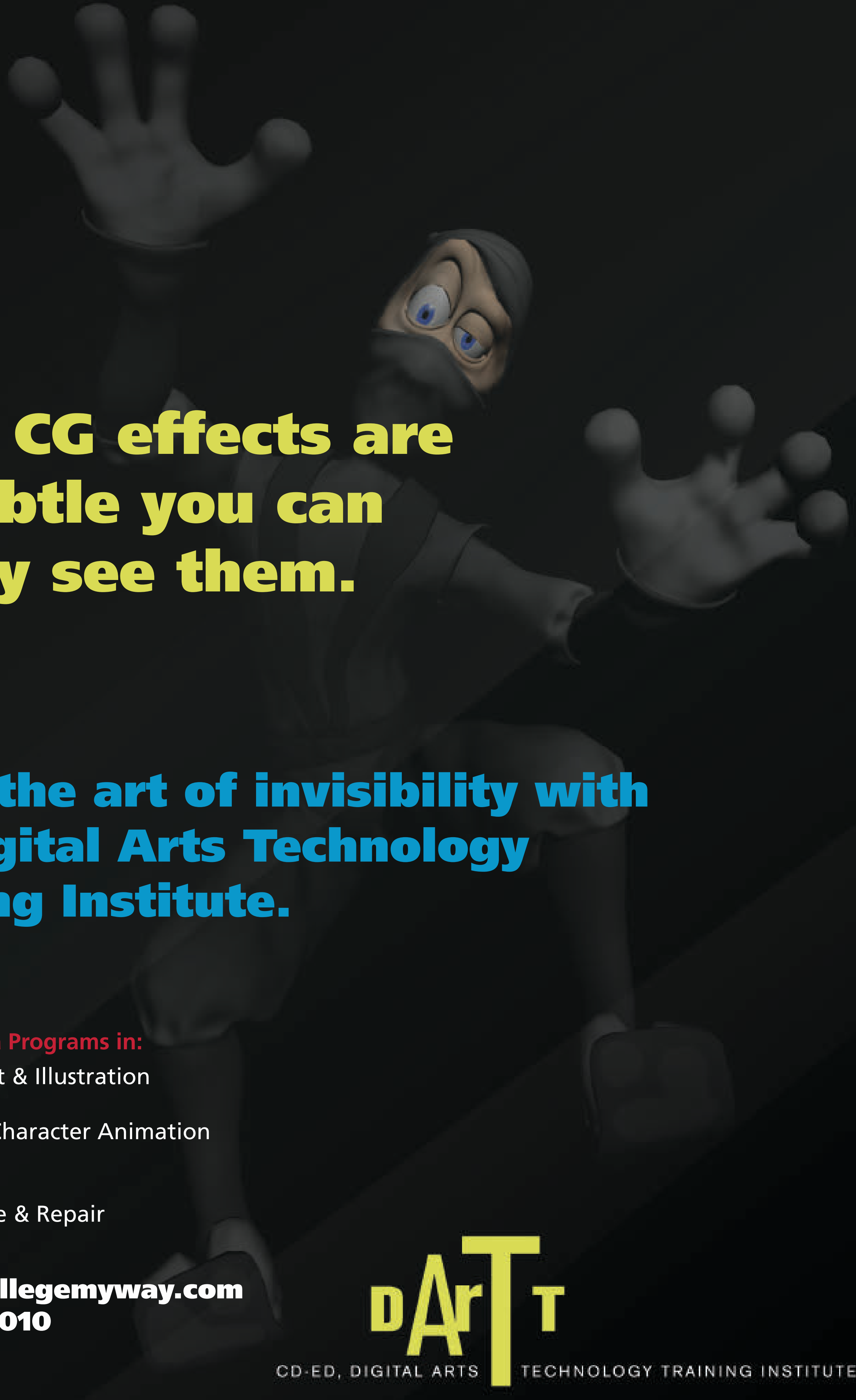
DeVry University

We major in careers.™

Accredited by The Higher Learning Commission of the North Central Association (NCA), www.ncahlc.org.

Program availability varies by location.

DeVry University operates as DeVry Institute of Technology in New York.



DIGIPEN
INSTITUTE OF TECHNOLOGY

a serious education for a serious career

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' theses focus on 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.

Learn more at WWW.DIGIPEN.EDU...

And stop by GDC Booth 5427!

5001 150th Ave NE | Redmond, WA USA 98052 | Phone (866) 478-5236 | FAX (425) 558-0378 | **www.digipen.edu**

COLLINS COLLEGE

Your passion. Our profession.

Interior Desgin

Information Technology

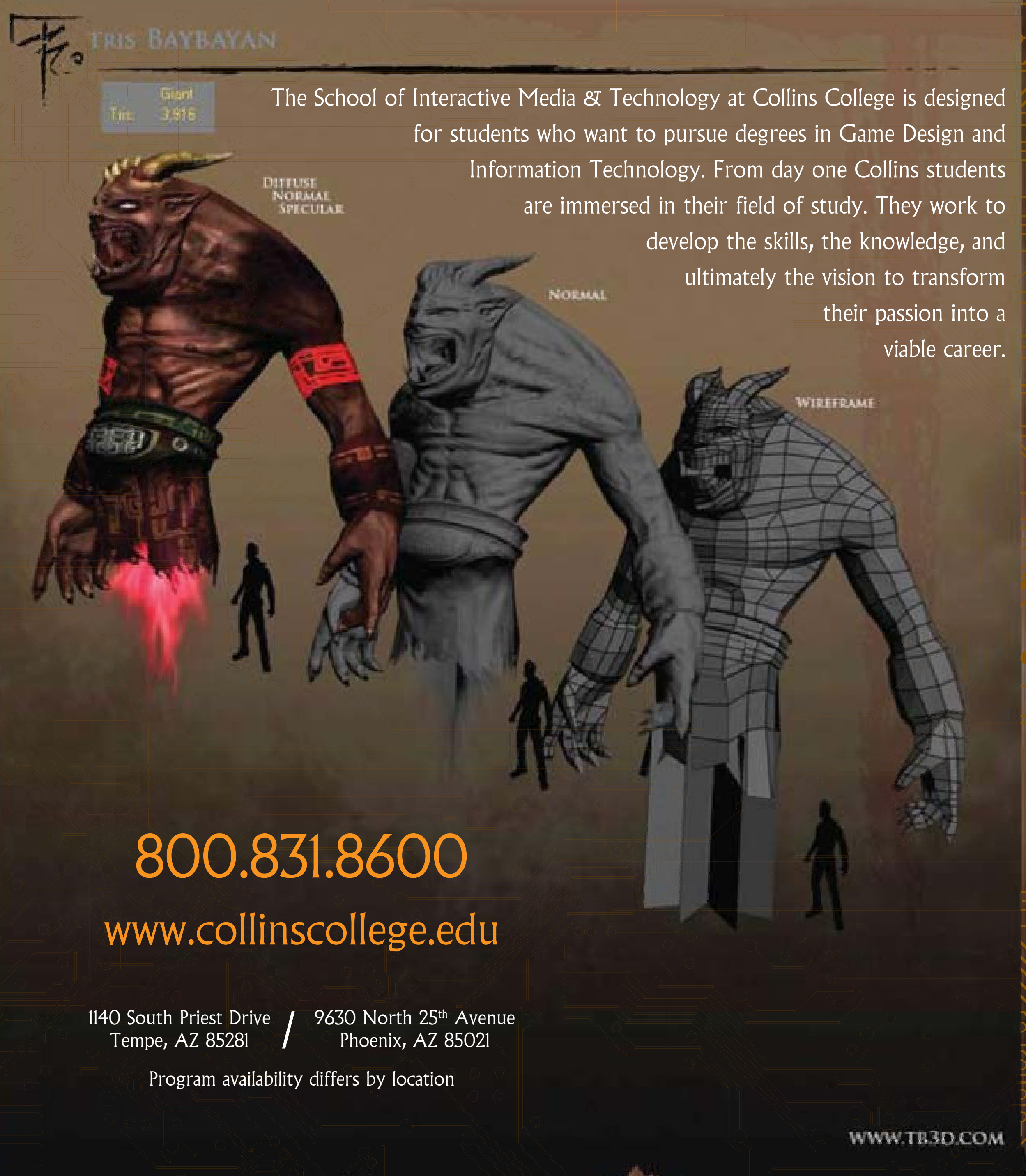
Digital Video Production

Film & Video Production

Visual Arts Game Art

Game Design

Graphic Design



The School of Interactive Media & Technology at Collins College is designed for students who want to pursue degrees in Game Design and Information Technology. From day one Collins students are immersed in their field of study. They work to develop the skills, the knowledge, and ultimately the vision to transform their passion into a viable career.

800.831.8600

www.collinscollege.edu

1140 South Priest Drive
Tempe, AZ 85281

9630 North 25th Avenue
Phoenix, AZ 85021

Program availability differs by location

GAMES ACADEMY™

founded in Berlin 2000

Teach or learn in the heart of Europe!

Education in: Game Producing > Game Design > Game Art & Animation > 3D Programming

North Hall,
Booth#6105
Visit our booth at GDC

GAMES ACADEMY™ GmbH
State approved vocational school

Managing Partner:
Thomas Dlugaiczyk, Andreas Stock

Courses available in Berlin &
Frankfurt am Main
Administration and Management:
Rungestrasse 20
D-10179 Berlin, Germany

Call: 011 - 49 - 30 - 29779120

or visit

www.games-academy.com

OUR MASTER'S DEGREE HAS ONE THING ON ITS MIND.

UCF School of Film and Digital Media

LEARN:

DIGITAL ANIMATION
DIGITAL ART AND DESIGN
DIGITAL VIDEO
GAME DESIGN
ARTIFICIAL LIFE PROGRAMMING
COMPUTER FORENSICS

GAME PROGRAMMING
NETWORK ENGINEERING
NETWORK SECURITY
SOFTWARE ENGINEERING
WEB ARCHITECTURE
ROBOTICS

University of Advancing Technology
UAT
Learn. Experience. Innovate.

You can talk the talk. Can you walk the walk? Here's a chance to prove it. Please geek responsibly. **www.uat.edu >** 877.UAT.GEEK
877.828.4335

VFS Student work by Alvaro Buendia

GNOMON

YOUR PIPELINE INTO THE INDUSTRY

GNOMON SCHOOL OF VISUAL EFFECTS

Gnomon School of Visual Effects specializes in training individuals for careers in high-end computer graphics for the entertainment industries.

HIGH-END CG
CERTIFICATE
PROGRAM

PROFESSIONAL
STUDIES
PROGRAM

MAYA
FAST TRACK
PROGRAM

5 & 10 WEEK DIGITAL CLASSES OFFERED
ON A QUARTERLY BASIS

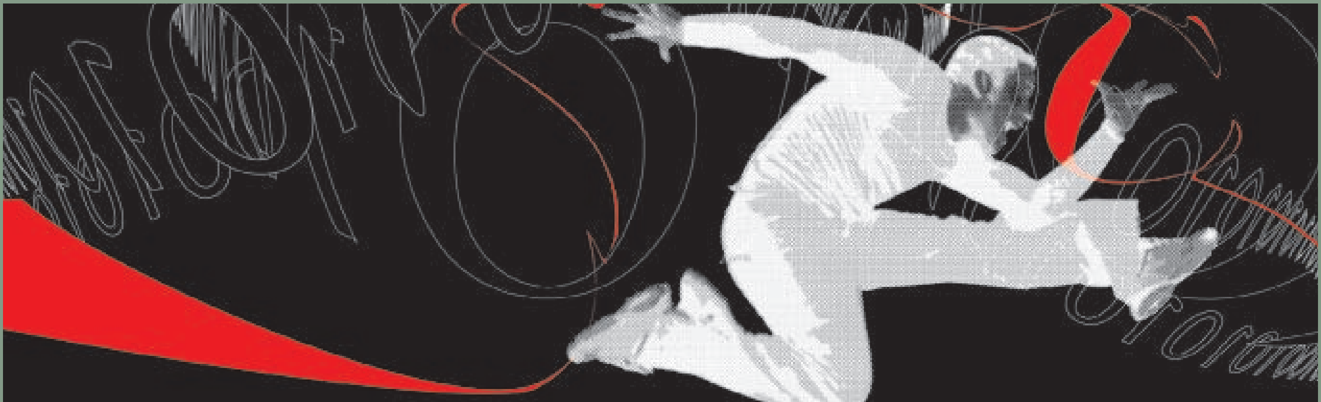
GNOMON ONLINE

Gnomon Online offers hours of web based training in high-end computer graphics to help artists become more productive, regardless of their physical location.

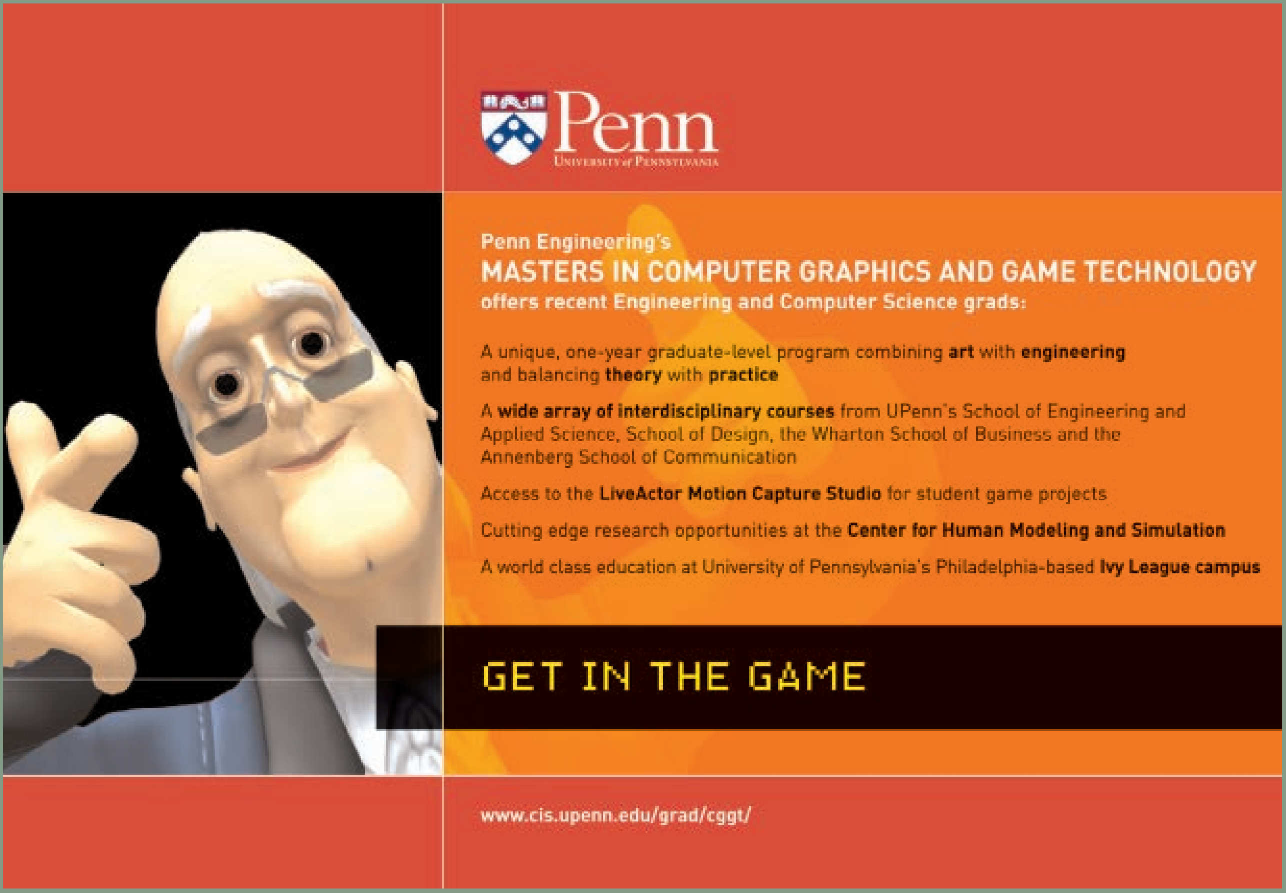
Join us at GDC 2008 - Career Pavilion Booth #209

www.GNOMONSCHOOL.com | www.GNOMONONLINE.com

For more information about GNOMON, email us at admin@gnomon3d.com or call 323-466-6663. 1015 N. Cahuenga Blvd - Hollywood, CA 90038



PRODUCT REVIEWS SIDE EFFECTS' HOUDINI 9
gamedeveloper
NOVEMBER 2007
THE LEADING GAME INDUSTRY MAGAZINE
PORTING WITH POWER BRINGING A GAME ENGINE FROM PC TO NINTENDO DS
AURALFIXATION AUDIO LOCALIZATION AROUND THE GLOBE
Q/A'S 10 COMMANDMENTS HOW TO BEST UTILIZE YOUR TESTING FRIENDS
POSTMORTEM: 2K GAMES' BIOSHOCK



ADVERTISER INDEX

Game Developer (ISSN 1073-922X) is published monthly by CMP Media LLC, 600 Harrison St., 6th Fl., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as CMP Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is $49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: $69.95; all other countries: $99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to Game Developer, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 1601 W. 23rd St. Ste. 200, Lawrence, KS 66046-2703. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 841-2624. 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/CMP Media, LLC, unless otherwise indicated. Don't steal any of it.

MARTY POULIN

THE NEXT INTERNET GAMING WAVE

IF WE TAKE A LOOK AT THE PETRI DISH WE call the Internet we notice something, it evolves pretty quickly. Last month's killer application is old news the day the next start-up launches. The companies that grow either have something intrinsically valuable, or can evolve faster than their peers. We are learning that the only true constant is ever accelerating change.

JACKING UP

Of course this isn't just happening in the Internet, it's bleeding into all facets of technology and content. While we may have previously measured it in the hardcore game industry in terms of rendered triangles and frames per second, we now measure it in development time, content footprint, and ever increasing budgets. An industry that was previously constrained by hardware now can't keep up with the event horizon of consumer expectation driven by Moore's law.

In many ways this high profile segment of the game industry has close parallels in film. Blockbusters are high risk endeavors where risk is mitigated with large marketing budgets, sequels and emulating past success.

At the beginning of the silent film era, production costs were dictated by technology rather than content and as technology, expectations and completion increased so did the costs of content.

As in film, blockbuster games are also just the tip of the iceberg. The combined markets of virtual worlds, niche MMOs, advanced casual (as is found on the Wii, Xbox Live Arcade, and others), handheld (DS, PSP), mobile, and casual games dwarf the high profile blockbusters like Halo and World of Warcraft at significantly lower costs.

Each of these segments is experiencing exceptional growth in the range of 15–40% annually with revenue from games already set to eclipse movie box-office. In contrast, movies, DVD, and television are stagnant or declining slightly across the board. The consistent growth of the game industry exemplifies a shift in the entertainment consuming habits away from non-interactive broadcast content to engaging interactive and social experiences.

I do not believe that big budget or console gaming is a dinosaur waiting to become extinct. The 39% growth in 2007 for console games should cement the fact that the industry is booming. As the number of next gen handhelds and consoles multiply, so do game sales—and the market is nowhere near saturation.

INTERESTING CURRENT

On the other hand, what I do find interesting is that almost 30% of Internet users worldwide play Internet games (casual and MMO) with revenues exceeding $3 billion. That market can be expected to double in the next 4 years producing a global market of around 500 million Internet game players.

While this in no way is detrimental to consoles, it does illustrate the continued shift and adoption of the Internet as the new multimedia operating system. The Internet already represents over 25% of total media consumption with growth rates accelerating this last year. In addition, online ad spending is expected to increase at an annual rate of 20% over the next 4 years.

HERE'S THE FUN PART

When we look for radical changes that can energize a market segment we look for unmet needs. An example of this can be seen with Nintendo's Wii and DS. Nintendo set out not to simply compete with Microsoft and Sony, but instead to focus on a market that wasn't being met. The Wii as an example made games not only physical, but intuitive. The emotional friction of having to learn a game was reduced. We all know how to bowl, and play tennis. Without that friction a whole new demographic of gamers jumped in with both feet.

Internet gaming is growing by leaps and bounds, but is currently polarized around Casual and MMO. This is our first clue that something is up. Like most things in nature the Internet abhors a vacuum and the lack of a strong segment of advanced casual games sticks out. Part of the reason for this vacuum is the lack of common platforms and delivery. While casual is mainly Flash and AJAX, advanced casual is typically embodied in 3D. Releasing a custom client only adds friction for the user, greatly reducing adoption.

CHANGE IS COMING

Adobe is reportedly adding 3D to its Flash plugin, and I would expect Microsoft to attempt to up the ante with Silverlight's features. At the same time the rise of social gaming on Facebook and other platforms is changing how we virally spread games and community. Breaking down the walls has created instant customer bases for the right content.

These trends haven't gone unnoticed. Silicon Valley and big media companies like Disney, Nickelodeon, MTV, and IBM have committed to spend over $1 billion in this area in the last year and have been putting a lot of fire under new start-ups as well as initiatives in virtual worlds and casual games.

This is all calculated to take advantage not only of a growing market, but new opportunities and untapped demographics. Over the next couple years we will see a big swing in the general quality of casual games on the Internet and a much greater range of game intensity. While true casual will remain king, the expectation of consumers will rise. At the same time an advanced casual segment will emerge as well as new mid complexity virtual worlds and MMOs.

It is always exciting to see a wave coming in and know that it's time to hop on. This one is on us and it's time to start paddling. If online is your passion, wade on in and join the fun. It's going to be one hell of a ride! ⁙

MARTY POULIN *founder of SocialSage, a company working to extend and redefine social media and games. He has worked previously at Sony, as director of online at Disney's Interactive Studios, and as CTO of the MMOG WORLD WAR II ONLINE. Email him at* ***mpoulin@gdmag.com****.*