



United Business Media

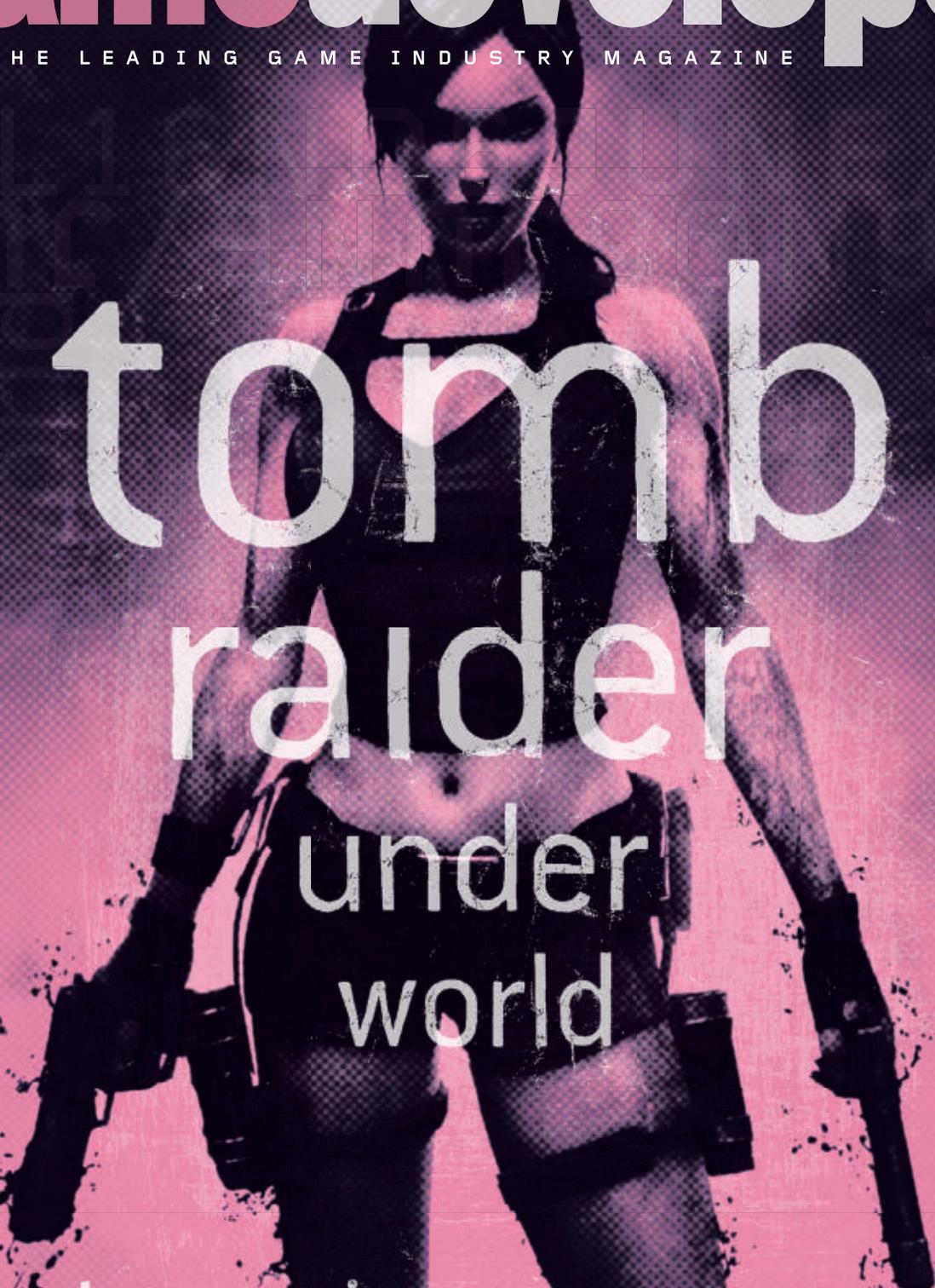
FEATURE: TOP 50 DEVELOPERS REPORT

VOL16NO6JUNE/JULY 2009

gamedeveloper

THE LEADING GAME INDUSTRY MAGAZINE

tomb raider under world



+ truly massive the finish line AN ARGUMENT FOR A SINGLE-SHARDED MMO ENCOURAGE YOUR PLAYERS TO COMPLETE YOUR GAMES

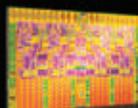


Proton Pact

Mark Randel, CEO Terminal Reality is putting on his best threads with Intel® and crossing streams with confidence.

"Spectrally speaking, teaming up with Intel on the development of *Ghostbusters: The Video Game* was the ecto-logical choice. I can't wait until everybody has fully threaded games on their desktop, because what's possible and what's going to happen in the game environment is going to blow peoples' minds."

Stay current on the latest software developments in visual computing at:
www.intel.com/software/visualadrenaline



©2009, Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries. *Other names and brands are the property of their respective owners. Image courtesy of Terminal Reality and Sony. Ghostbusters is a registered trademark of Sony Pictures LTD in the United States and/or other countries. The ratings icon is a registered trademark of the Entertainment Software Association.

POST MORTEM

- 34 CRYSTAL DYNAMICS' TOMB RAIDER: UNDERWORLD**
TOMB RAIDER: UNDERWORLD was originally planned as an "easy" sequel to the previous game in the series, using the same toolchain—but it never quite works out that way. Eric Lindstrom describes the process, highlighting instances in which the team wasn't able to avoid known problems. *By Eric Lindstrom*

FEATURES

- 6 TOP 50 DEVELOPERS**
Our second annual Top 50 Developers report uses metrics such as review scores, revenues, and reputation surveys to determine the best and brightest developers that released games in the past year. *By Trevor Wilson*
- 15 INFINITE SPACE**
Most MMOs use multiple shards to break up their game universes. EVE ONLINE, by contrast, uses a single shard. Here, various members of the team, from programmers to designers, share why they feel one shard is the way to go. *By the EVE Online team.*
- 22 STAYING POWER**
Microsoft researcher Bruce Phillips finds that players often quit single-player campaigns before they're complete. Citing psychology and exclusive statistics, Phillips proposes that "learning goals" can help keep players from getting discouraged. *By Bruce Phillips*

DEPARTMENTS

- 2 GAME PLAN** *By Brandon Sheffield* [EDITORIAL]
The Personality Problem
- 4 HEADS UP DISPLAY** [NEWS]
Thinking After Dark conference report, GP2X Wiz launch, GDC: THE GAME, and more.
- 41 TOOL BOX** *By Greg Snook* [REVIEW]
Blade Games World's Blade3D
- 43 THE INNER PRODUCT** *By Noel Llopis* [PROGRAMMING]
Mock Objects: Friends or Foes?
- 46 PIXEL PUSHER** *By Steve Theodore* [ART]
Metagames
- 49 DESIGN OF THE TIMES** *By Damian Schubert* [DESIGN]
Tactical Transparency
- 51 AURAL FIXATION** *By Jesse Harlin* [SOUND]
Thinking Outside the Booth
- 56 ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]
Ask a Pizza



THE PERSONALITY PROBLEM

THE IPHONE'S APP STORE AS A MICROCOSM OF THE INDUSTRY AT LARGE

A FRIEND OF MINE RECENTLY released a game for iPhone by the name of *TRIXEL*. It's a fine puzzle game, somewhat similar to *LIGHTS OUT*, in which you flip mismatched colored tiles to match an existing tile image. People who play it definitely seem to like it. But visually this game has almost no personality. Certainly, the tiles are large and colorful, there are power-ups and collectables, and the audio was carefully attended to. But if you look at a screenshot and a description, you would likely not be compelled.

Another group of friends, the folks at Capybara Games, put out an iPhone game called *CRITTER CRUNCH*. This is a puzzle game as well, similar to *MAGICAL DROP*, and starring a cute frog-thing that eats cutely animated characters. Now, I can't really speak to which of these two games is more successful, but I can say that if I look at a screenshot of each, one compels me with characters and bright colors, whereas the other looks either a bit kiddie or a bit math-oriented, depending on how you feel about it (and in reality, the game can get a bit hardcore).

Taken as a microcosm of the industry, the iTunes App Store emphasizes some larger industry truths. In the case of something so impulse-buy-oriented as iPhone games, when a number of free titles already exist, one really needs a hook to succeed. But then, hooks are necessarily oriented toward certain audiences. Some folks may really like the cute characters in *CRITTER CRUNCH*, but others may be completely turned off. Both *TRIXEL* and *CRITTER CRUNCH* are good, and both lie within the puzzle genre. So how do you get people interested in *TRIXEL*, when *CRITTER CRUNCH* is sitting next to it in the virtual shelves?

Looking at the bigger picture, console games are only on the store shelves for a limited time,

before they're shuffled away to make room for something new. They have very limited space in which to get the interest of the consumer who just wanders into a GameStop looking for something new to play, which happens more than most of us realize. Someone like you or I will go to the store with a head full of previews, trailers, screenshots, story descriptions, and maybe a few behind-the-scenes stories. But the average consumer is just showing up at a store, looking to be entertained. These games need to grab consumers immediately as well, and have something the idle browser can latch on to.

CASUAL CONSUMERS

» I recently overheard a conversation in a GameStop—a late-teens customer walked in, and found the box art for *FINAL FANTASY XII* appealing. He brought it to the cashier and asked what kind of game it was. “An RPG,” was the response. “Oh. What's that?” “Um, you know, a role-playing game. You have a group of guys, and you go on a quest, and you level up and stuff.” “Oh. Is that fun?”

This anecdote just shows that we can't rely on the store itself to sell our products. Developers complain about releasing games on Apple's App Store amidst a sea of other titles, with no way to distinguish a title other than getting featured by Apple. Well shouldn't we be used to dealing with that by now? The same thing happens in retail. And indeed, isn't it better than a situation in which your game drops out of the store entirely after a couple months, as with retail? And there are no used games there to cannibalize your actual sales (though one could argue that free games might take a chunk away).

So at this point it becomes a marketing issue. I wouldn't say that independent iPhone developers need a marketer,

but they do need to do some marketing themselves. I'm not just talking about sending out free review codes to folks you might know in the media, though that helps a lot. The reason a personality-free game like *SUDOKU* is so popular now is likely because of this kind of marketing—the mom-oriented media got ahold of it, and it took off.

What I'm talking about is “marketing” in the actual planning phase. If you want the game to sell, realize you're not just making it for people who innately get it, like you—you're making the game for people like that GameStop customer. People who don't understand your game, because they haven't played it, and have maybe never played anything in the genre. For these people, you need appealing screenshots that make your game look like something. You need a compelling description, and possibly a demo.

That's the kind of marketing I mean—marketing at the base level. Questions like “Who will this appeal to visually? How can I describe my game in three sentences?” should be at the front of your mind. The kinds of questions publishers would ask, if you had one. Show the game to your mom, or your kid, or your neighbor, and see what they think.

MARKET IN FOCUS

» A lot of companies and developers want to reach larger mainstream audiences, and the iPhone takes all the elements of the wider game industry and puts a greater focus on it. The game has to look pretty, but simple. The concept has to be easy to understand, but difficult to master. It's everything we're doing for AAA titles, but under a microscope. I think there are a lot of lessons to learn here, and the iPhone could potentially be used as a test market for larger concepts.

—Brandon Sheffield

Think Services, 600 Harrison St., 6th Fl.,
San Francisco, CA 94107
t: 415.947.6000 f: 415.947.6090

SUBSCRIPTION SERVICES

FOR INFORMATION, ORDER QUESTIONS, AND ADDRESS CHANGES

t: 800.250.2429 f: 847.763.9606
e: gamedeveloper@halldata.com

EDITORIAL

PUBLISHER

Simon Carless | scarless@gdmag.com

EDITOR-IN-CHIEF

Brandon Sheffield | bsheffield@gdmag.com

PRODUCTION EDITOR

Jeffrey Fleming | jffleming@gdmag.com

ART DIRECTOR

Joseph Mitch | jmitch@gdmag.com

SENIOR CONTRIBUTING EDITOR

Jill Duffy | jduffy@gdmag.com

CONTRIBUTING EDITORS

Jesse Harlin | jharlin@gdmag.com
Steve Theodore | stheodore@gdmag.com
Noel Llopis | nllouis@gdmag.com
Soren Johnson | sjohnson@gdmag.com
Damion Schubert | dschubert@gdmag.com

ADVISORY BOARD

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

ADVERTISING SALES

GLOBAL SALES DIRECTOR

Aaron Murawski | amurawski@think-services.com
t: 415.947.6227

MEDIA ACCOUNT MANAGER

John Malik Watson | jmwatson@think-services.com
t: 415.947.6224

GLOBAL ACCOUNT MANAGER, EDUCATION AND RECRUITMENT

Gina Gross | ggross@think-services.com
t: 415.947.6241

COORDINATOR, EDUCATION AND RECRUITMENT

Rafael Vallin | rvallin@think-services.com
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER

Robert Steigleider | rsteigleider@ubm-us.com

REPRINTS

WRIGHT'S REPRINTS

Ryan Pratt | rpratt@wrightsreprints.com
t: 877.652.5295

THINK SERVICES

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

AUDIENCE DEVELOPMENT

GROUP DIRECTOR Kathy Henry

khenry@techinsights.com
DIRECTOR Kristi Cunningham
kcunningham@techinsights.com
LIST RENTAL Merit Direct LLC | t: 914.368.1000

MARKETING

SERVICES MARKETING COORDINATOR Laura Robison
lrobison@think-services.com

UBM TECHNOLOGY MANAGEMENT

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



Look Who's #1



Top Products for the Top Games



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

The Proven Solution.



PHYSICS



ANIMATION



BEHAVIOR



CLOTH



DESTRUCTION



AI

©2008 THQ International GmbH. THQ, de Blob and their respective logos are trademarks and/or registered trademarks of THQ Inc. All Rights Reserved. All other trademarks, logos and copyrights are the property of their respective owners.



HAUNTED SCHOOLYARD

A REPORT ON THE THINKING AFTER DARK HORROR GAME CONFERENCE

IN LATE APRIL I WAS FORTUNATE enough to participate in Thinking After Dark: Welcome to the World of Horror Video Games, a conference organized by a research group from the University of Montreal called Ludicine and run by Bernard Perron, who has written at length about horror games. Held in what appeared to be an old church (of which Montreal is in no short supply), the small conference attracted a wide range of academic scholars from around the world, a couple of industry folk, and some hard-core fans. The event lasted three days and covered a wide swath of topics related to horror games; titles like *RESIDENT EVIL*, *SILENT HILL*, and *FATAL FRAME* were

research in film studies to show that mechanical genres evolve with iteration (the “DOOM clone” genre was replaced by the “first person shooter” genre around 1998, according to his research), while thematic genres draw from other mediums and are primarily concerned with aesthetics. Both Arsenault and Therrien concluded that the only true definition of genre is popular consensus. Arsenault has made his slides available: www.le-ludophile.com/Files/Arsenault-Thinking_After_Dark_paper.ppt

Alexis Blanchet, from the Université Paris Ouest Nanterre La Défense, shared details of his three-year research project

Richard Rouse III, lead designer of *THE SUFFERING* series and current lead single player designer at Kaos Studios, gave a postmortem of the development process behind *THE SUFFERING*. He discussed the difference between making a “survival horror” game and an “action horror” game (as he put it, the difference between the films *Alien* and *Aliens*), as well as the difficulty in trying to use a commercial game as a vehicle for social commentary without making his team or publisher uncomfortable.

Referring to *The Shining* during development, along with feedback from his publisher, led Rouse to change the focus of his game and produce something that had a much more healthy dose of horror than it would have otherwise achieved. Rouse also pointed out that horror is uniquely positioned to deal with taboo or controversial topics (in the case of *THE SUFFERING*, capital punishment) because it is able to “fly under the radar.”

Angela Tinwell of the University of Bolton presented her research about which aspects of horror game characters are most directly tied to the uncanny valley effect. Her thesis is that the uncanny valley might actually be beneficial for horror games, as it

can make characters subtly disconcerting. Her research found strong correlations between the intonation of speech, the synchronization of speech with mouth animation, and animation of the forehead and feelings of uncanniness. It also showed that fantastical characters were usually found to be more convincing than realistic humans.

William Huber from the University of California San Diego talked about his method of data mining video of *FATAL FRAME II* play sessions to learn about the tempo and rhythm of the game.

His system uses image recognition software to identify each frame of a play session video as one of several game play modes (combat, navigation, cut scene, pause screen, etc). The output can be graphed in terms of types of play over time, and lead him to dub the formula used in *FATAL FRAME II* “catch and release,” in which tension that is built by cut scenes is then incompletely released through navigation and combat play.

Denis Béline and Alexandra Munger from the Université de Sherbrooke presented very early findings based on their study of the physiological effects caused by horror games. They hooked test subjects up to equipment and applied lessons from lie detection research to monitor changes in physical state as the subjects played. Though they are still collecting and analyzing the data, one subject in particular has already made the study worthwhile: during a play session of *FATAL FRAME II*, the subject’s heart actually stopped for three seconds. This kind of data can only be collected by accident because of the ethical problems related to scaring people for the purposes of research, so they are extremely lucky to have recorded such a result (the test subject was unharmed).

Despite its seemingly narrow focus on horror video games, the talks at the Thinking After Dark conference were very diverse and thought provoking. A selection of papers from the conference are to be published in a special issue of *Loading...*, the journal of the Canadian Game Studies Association. A book collecting the work of many of the speakers called *Gaming After Dark* is also scheduled to be published this spring. More details can be found at the conference web site: <http://conference2009.ludicine.ca/en>

—Chris Pruett



recurring topics of discussion. In true Canadian fashion the talks were split almost evenly between French and English, and translated slides were provided on a separate screen. Here are a few highlights from the event.

Dominic Arsenault and Carl Therrien, both part of Ludicine, discussed the difficulty of using a single genre to categorize horror games. Both separated genre into mechanical and thematic components (“survival vs “horror”), and discussed how the division between the two makes categorizing horror games in this way particularly problematic. Arsenault drew from

aimed at compiling a database of all games based on films. One of his many findings was that horror is a fairly infrequently adapted genre (there have only been 51 horror film game adaptations to date, compared with 232 action films, 222 adventure movies, and 169 comedies), and that the majority of adaptations come from films that are rated PG and have original screenplays. Curiously, spikes in the release of film-based games seem to occur just before new console hardware launches. Blanchet’s findings are available online (in French) at: <http://jeuvideal.com/?p=215>

GP2X WIZ

AN OPEN-SOURCE LINUX-BASED HANDHELD

THE KOREA-ORIGINATING GP2X WIZ IS THE LATEST IN GAME PARK

Holdings' line of handhelds, which includes the GP32 and GP2X series of consoles. Like its immediate predecessor, the system is Linux-based, and this time around includes a 533 MHz ARM9 3D accelerator as its CPU, 64 megs of SD RAM, 1 gig of internal flash memory, and a 320x240 2.8 inch AMOLED touch screen.

The most striking element of this new handheld is its dual d-pads. Technically the left pad is a real d-pad, and the right "pad" is actually four independent buttons in the configuration of a d-pad, but for ambitious developers looking to make ambidextrous games or experiment with alternate control schemes, this is an intriguing platform.

The system's SDK is quite similar to that of the GP2X, so hobbyists with experience on that platform should feel right at home. Blogger from Play-Asia-Rulez and amateur coder Ed Mandy described the experience:

"Writing code for the GP2X Wiz has been a breeze. I'm actually fairly impressed that Game Park Holdings was able to use the same binary format on the Wiz that the GP2X used. That allowed me to use the same compiler, toolchain, and libraries that I had for developing on the GP2X [from the Open2X project: www.distant-earth.com/open2x]. The only real change that I made was specifying the use of shared libraries instead of statically-linking libraries as I had done with the GP2X."

The company is trying to turn the console into a viable sales platform as well, selling both proprietary games and flash games for the console through its own iTunes-like download service, which had not yet launched as of press time. The console comes with 12 embedded games and demos, notably a fighting game and beat-em-up from acclaimed Korean indie developer Byulbram. The console has the financial support of the government-run Electronics and Telecommunications Research Institute (ETRI).

With the GP2X Wiz, there will be three open source, emulation-fueled handhelds on the market at the same time, including the Pandora and the Dingoo A-320, both mentioned previously in these very pages. Whether the hobbyist culture is large enough to support all three remains to be seen. Will the factions become compartmentalized? Will regional origins give each system its own flavor? Regardless, it's a good time to be a hobbyist developer, if you want to see your work end up on consoles, with XNA Creators Club and now three handheld vehicles through which to display new creations. The GP2X Wiz is available now, for \$180. —Brandon Sheffield



GDC: THE GAME

WANT TO REPLICATE THE GDC

experience for free, and without the pesky smells and sounds of a convention center? Now you can! Think-Service's game-oriented blog GameSetWatch commissioned

Munroe's own words: "A bunch of randomly generated convention-goers wandering around the Moscone Center, annoying and impressing each other, talking about things they know about

able to make a game, which is a bit meta if you think about it. "I wanted to try something that was more of a 'text game' rather than 'text adventure game,'" said Munroe. "Think of it as a round of cards rather than an immersive and colorful narrative. If you don't like the hand you're dealt, you can always reshuffle with a restart. If you find you're playing "guess-the-verb" (IF's most infamous minigame), restart and read the beginning carefully."

GDC: THE GAME is available for free at www.gamesetwatch.com/gdcgame, which plays in a browser and requires Java, or with Google Parchment, which does not require Java: <http://tinyurl.com/q5both>. Good luck finding a coder! They're as tough to nail down in the game as they are in the real world.

—Brandon Sheffield

```
Raigan looks around for someone, then checks his watch.
Ryan talks endlessly to you about chiptune music, but it's obvious from his vague blaster he knows nothing about it.
>Talk to Raigan
What do you want to talk to Raigan about?
>physics
Your interest in 2D physics proves boring to Raigan.
Now you have dropped in Raigan's esteem.
Someone named Mare seems a little lost.
Katherine talks to you about 2D physics, which she obviously knows nothing about.
Now Katherine has dropped in your esteem.
Shooting you a dirty look, Raigan heads to the code session "AI Behaviors from Improvisational Actors".
Ryan floats away to art session "Lighting With Purpose".
You blink, and suddenly Wei-fu is gone, as quietly as he arrived.
```

interactive fiction author Jim Munroe to make a GDC-specific text adventure, which winds up being more of a simulation than an adventure game.

GDC: THE GAME, contains in

and things they know nothing about, and as the player character you can stand there and watch it happen or jump in."

The desired end result is to assemble a virtual team, and be

CALENDAR

» Paris Game AI Conference

CNAM, Amphi C Abbé Grégoire Paris

June 10–11

Price: Free
<http://aigamedev.com>

» 2009 Game Education Summit

Carnegie Mellon's Entertainment Technology Center Pittsburgh, PA

June 16–17

Price: \$349–\$399
www.gameeducationsummit.com

» Develop Conference

Hilton Brighton Metropole Brighton, U.K.

July 14–16

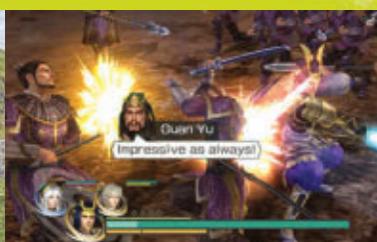
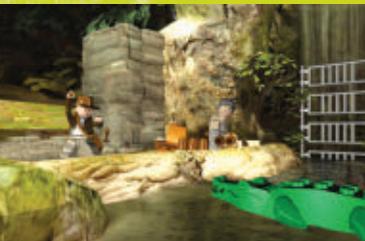
Price: £250–£600
www.develop-conference.com

» Casual Connect Seattle 2009

Benaroya Hall Seattle

July 21–23

Price: \$450–\$750
<http://seattle.casualconnect.org>



50

2009 TOP 50

DEVELOPERS

R E P O R T

THERE ARE A LOT OF GAME DEVELOPERS OUT THERE, DOING A LOT OF GOOD work. Those that weather this worldwide economic storm will reap greater profits when it ends, but for now, it remains a difficult time.

In the latest in our long-established Top 50 Developers countdown, we honor fifty game creation studios, worldwide, that did stellar work in the past year. We specifically honor companies releasing major titles or updates in calendar year 2008. Thus, the effects of development cycles on single-studio developers means this list is going to look very different from year to year.

Developers were ranked according to a composite score influenced by number of 2008 releases, average review score (per Metacritic), the sales data kindly released by Media Create (JP), NPD Group (U.S.), and GfK-ChartTrack (U.K.), and generalized and detailed survey responses.

Our survey, which collected over 500 responses from readers of *Game Developer* and *Gamasutra.com*, asked responders to rate the reputations of any developers of their choice on a 1–10 scale. It also asked responders to rate developers with which they have worked directly in terms of overall impression, pay and perks, professionalism in production, and likeliness to work the developer again.

It is worth noting that the imperfect nature of sales reporting regarding online downloads, microtransactions, and subscriptions means that the Top 50 Developers list is potentially more retail-heavy than we might like it to be. But we're working on ways to improve that going forward, and you'll see a number of notable online-centric firms on the list nonetheless.

Developers newly added to the Top 50 this year include: Success, Rare, Funcom, SCE Studio San Diego, Ironclad Entertainment, Mythic Entertainment, Criterion Games, Monolith, Marvelous Interactive, Yuke's, Intelligent Systems, Visual Concepts, SCE London Studio, SCE Japan Studio, Omega Force, Volition, Atlus, Ready at Dawn, Vicarious Visions, DICE, Lionhead Studios, Media Molecule, Kojima Productions, Treyarch, Koei, Rockstar North, and Treasure.

An enhanced paid version of this report with full feedback, charts, and stats is available at www.gamedevresearch.com.



TOP 50 DEVELOPERS 2009



50) TREASURE

(BANGAI-O SPIRITS)

» Sitting at the end of our ranking is the cult-favorite developer Treasure. It put out two solid DS releases in 2008—including the quirky sequel BANGAI-O SPIRITS—and ended up with an average review score of 82%. Treasure didn't get excessive attention on our reputation survey, but even so, devoted fans granted it a cool 10-point average.

49) RARE

(VIVA PIÑATA: TROUBLE IN PARADISE)

» The Microsoft-owned Rare had a prolific year, with sterling critical reception for two VIVA PIÑATA titles and a new BANJO-KAZOOIE, making for an 81% review average. Commenters gave Rare mediocre reputation scores, but noted that the studio "always tries to innovate" and considered Rare a "great company to work for."

48) SCE STUDIO SAN DIEGO

(MLB 08 THE SHOW, NBA 09 THE INSIDE)

» Sony's sports-focused subsidiary produced two multiplatform titles in 2008: MLB 08 THE SHOW and NBA 09 THE INSIDE, which resulted in multiple SKUs, and this studio's first showing on our survey. These titles were well-received critically, making for a 77% review average, the highest score of

which was an 85% Metacritic score for the PlayStation 3 version of MLB 08.

47) FUNCOM

(AGE OF CONAN: HYBORIAN ADVENTURES)

» The Norwegian developer, known for adventure games like DREAMFALL and THE LONGEST JOURNEY released the online action-RPG AGE OF CONAN: HYBORIAN ADVENTURES in early 2008. The title received an 80% average review score and positive initial sales, giving Funcom our number 47 spot, though it subsequently ran into update- and subscriber retention-related issues. Funcom employees scored the developer well and praised it for "a lot of effort put into helping employees" and for having "almost no overtime."

46) MONOLITH

(CONDEMNED 2: BLOODSHOT)

» The narrative-oriented, Warner Bros.-owned Washington State developer of NO ONE LIVES FOREVER and F.E.A.R. released the multiplatform survival-horror first person title CONDEMNED 2: BLOODSHOT in 2008, which earned Monolith an 81% review average for the year. In our reputation survey, scores for the studio were found somewhat wanting, but ratings given by those who have worked with the studio were strongly positive.

45) OMEGA FORCE

(DYNASTY WARRIORS: GUNDAM)

» Koei's DYNASTY WARRIORS-focused studio did not place on 2007's ranking, but a 15-game release schedule—not all of which were WARRIORS games—secured Omega Force a place on this year's list. Still, the years-long slide in critical opinion of these games continued in 2008, and this developer's games averaged a paltry 54% (compare that with parent Koei's 68%). These games do continue to sell, of course, and WARRIORS OROCHI 2 and DYNASTY WARRIORS: GUNDAM proved potent on Japanese charts.

44) INTELLIGENT SYSTEMS

(ADVANCE WARS: DAYS OF RUIN, FIRE EMBLEM: SHADOW DRAGON)

» The two 2008 games developed by this wholly-owned Nintendo developer were new entries in the perennially acclaimed ADVANCE WARS and FIRE EMBLEM series, both of which were appreciated by critics. The games brought the developer an 84% average review score, and FIRE EMBLEM: SHADOW DRAGON made a strong showing on Japanese sales charts. Survey respondents also showed their appreciation for IntSys's work on "the best turn-based strategy games in the industry."

43) VISUAL CONCEPTS

(NBA 2K9)

» NBA 2K9 for Xbox 360 proved the biggest seller for Take Two's sports-specialized studio, which takes on EA's sports imprint head-to-head. The PlayStation 3 version also charted well, and a plate of eight titles across a variety of platforms in 2008 gave this California developer a 73% review average and its first-ever Top 50-ranked position on our survey.

42) SCE LONDON STUDIO

(SINGSTAR VOLUME 2 and 3.)

» Sony Computer Entertainment's largest internal developer comes in with a significant number of retail releases, all of them SINGSTAR-related. The studio continues to support the PlayStation 2 more heavily than any of Sony's other developers, though SINGSTAR has found a home on PS3 as well. The successful series made a reasonably favorable showing with critics in 2008, and came home with a 72% average review.

41) YUKE'S

(WWE SMACKDOWN VS. RAW 2009, THE DOG ISLAND)

» WWE titles were the driving force that brought Osaka-based Yuke's onto our survey this year. The developer released WWE



SMACKDOWN vs. RAW 2009 on multiple platforms to respectable commercial success, and last year's edition of WWE brought Yuke's some success in the U.K. market. 2008 also saw U.S. release of Yuke's' insidiously cute THE DOG ISLAND for PlayStation 2 and Wii. All of the above combined to give the studio a 75% Metacritic average.

40) MARVELOUS INTERACTIVE

(HARVEST MOON: TREE OF TRANQUILITY)
 » A combination of Western and Japanese releases of HARVEST MOON games gave this Tokyo publisher's internal studios multiple notable releases in 2008. The titles that saw U.S. release earned fair-to-middling favor with critics and gave MMV an unremarkable 69% composite review score. HARVEST MOON seems an evergreen series, and is what brings Marvelous onto our list.

39) IRONCLAD GAMES

(SINS OF A SOLAR EMPIRE)
 » Ironclad, a fully-independent, Burnaby, BC-based developer released its PC strategy title—heavily co-produced with Stardock—SINS OF A SOLAR EMPIRE to critical acclaim and sales of over 600,000 copies, which perhaps unexpectedly made the game one of the top twenty best-selling PC titles of the year in U.S. retail,

according to NPD. Though piracy is always a trouble, fans carried the game through, and the game sports an 88% review average.

38) SCE JAPAN STUDIO

(SIREN: BLOOD CURSE, PATAPON 2)
 » A stable of sometimes quirky, acclaimed titles brought Sony's Japanese home studio onto our lineup this year. Japanese retail releases for PATAPON 2 and ECHOCHROME, plus the horror title SIREN: BLOOD CURSE and several niche-focused releases, garnered an overall Metacritic score of 76%. Readers praised SCE's "fantastic, interesting, artistic, compelling games" and gave the developer a 9.0 reputation score.

37) SEGA

(MARIO & SONIC AT THE OLYMPIC GAMES, VALKYRIA CHRONICLES)
 » Sega's home development studio, based in Tokyo, rose to #37 from last year's #44. This was driven by double the number of titles compared to last year's schedule, better review scores, and sales of the runaway Wii hit MARIO & SONIC AT THE OLYMPIC GAMES. The feudal YAKUZA sequel RYU GA GOTOKU KENZAN and the WWII-ish fantasy SRPG VALKYRIA CHRONICLES also proved themselves in the Japanese marketplace.

36) VOLITION

(SAINTS ROW 2)
 » In 2008, this THQ-owned developer released SAINTS ROW 2, which sold about two million copies across all platforms and brought home an average review score of 81%. Based in Champaign, IL, Volition also received favorable marks on both our reputation and specific surveys, and received praise for its quality of life "for families in particular."

35) FIRAXIS

(CIVILIZATION: REVOLUTION, CIV IV: COLONIZATION)
 » Firaxis released four titles in 2008 after two in 2007. CIVILIZATION: REVOLUTION and CIV IV: COLONIZATION sold and scored well for the Hunt Valley, MD studio, though review scores fell just under 2007's to a still-impressive 83%. Firaxis has evidently earned goodwill from the community at large, as it received quite favorable reputation scores and praise for CIV IV's "unparalleled refinements for the genre."

34) MYTHIC ENTERTAINMENT

(WARHAMMER ONLINE: AGE OF RECKONING)
 » Last fall's WARHAMMER ONLINE: AGE OF RECKONING sold over 1.3 million copies in its first quarter of release and became the fifth-

best-selling U.S. PC retail game in 2008. The experience from DARK AGE OF CAMELOT that this Fairfax, VA EA subsidiary rolled into WAR also secured an 86% review average.

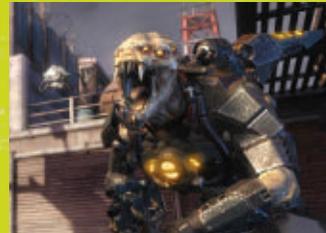
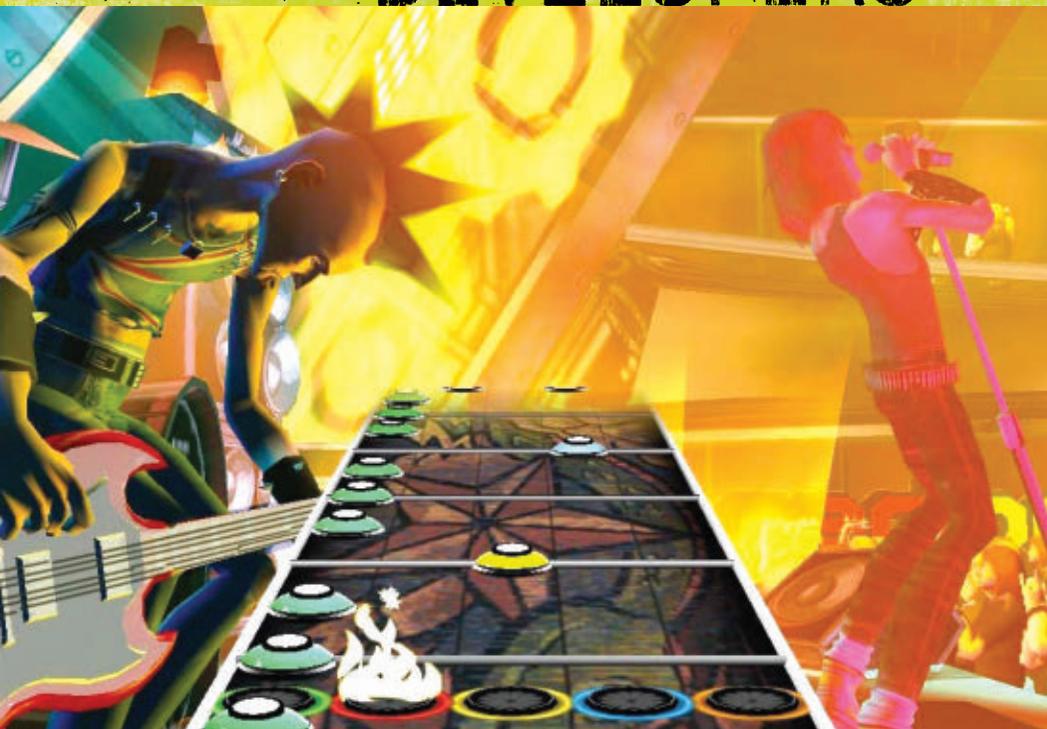
33) TOSE

(CHRONO TRIGGER DS)
 » This developer generally shuns the spotlight, preferring to let its clients take the credit for its games, but TOSE has once again made it onto our ranking on the strength of a refreshed DS version of CHRONO TRIGGER and other stealthy titles. Thanks to higher profile releases, Kyoto-based TOSE rose by one spot this year, despite a review average that fell by ten points and a slightly trimmed release schedule [public releases, at least!].

32) CRITERION GAMES

(BURNOUT PARADISE)
 » Criterion comes in at slot 32 on the strength of the high-scoring and commercially successful BURNOUT PARADISE, which gave this Guildford, U.K. division of Electronic Arts an 87% review average. Criterion was a favorite on our survey and received high reputation marks and special praise for the way it has "surpassed DLC and [has] reached something altogether new and different."

TOP 50 DEVELOPERS 2009



31) ATLUS

[ETRIAN ODYSSEY II, PERSONA 4]

» This Tokyo developer makes our list on the strength of its home-grown RPGs. PERSONA 3: FES, PERSONA 4, and ETRIAN ODYSSEY II each found an appreciative audience with U.S. critics, making for a review average of 82%. The majority of the company's titles also found sales success within Japan and the U.S., and this fan favorite received strongly positive reputation scores and comments. One commenter praised the developer for always giving the fans "something completely unique."

30) READY AT DAWN

[GOD OF WAR: CHAINS OF OLYMPUS]

» This Irvine, CA independent rides in on the strength of the acclaimed GOD OF WAR: CHAINS OF OLYMPUS for PSP. Survey commenters called the "up and coming" Ready at Dawn, which is believed to be readying original IP, "sure to be an amazing studio." CHAINS OF OLYMPUS sold over 300,000 copies in the first month of its release, and this studio was a critical darling, scoring a 91% average review score.

29) SUCCESS

[THE DARK SPIRE]

» Fourteen 2008 titles help this Japanese maker of THE DARK SPIRE skate onto our lineup this

year. Tokyo-based Success has had several of its titles published by Atlus in the U.S. of late, but its publishing and developing efforts in Japan are varied and prolific, with many original and non-licensed IPs. However, the titles that did see U.S. release in 2008 only managed a 68% Metacritic score overall. In Japan, the adventure game AOISHIRO saw a respectable showing on weekly sales charts.

28) DIGITAL ILLUSIONS (DICE) STOCKHOLM

[BATTLEFIELD: BAD COMPANY, MIRROR'S EDGE]

» The combination of solid sales performance brought by BATTLEFIELD: BAD COMPANY and the innovation shown in MIRROR'S EDGE gave DICE a respectable spot this year. MIRROR'S EDGE didn't make the same initial sales splash as BAD COMPANY, though owner EA claims the former has sold over a million copies worldwide as of this writing. Both titles performed well critically, giving the Swedish developer an average review score of 81%.

27) SQUARE ENIX

[CRISIS CORE FINAL FANTASY VII]

» Square Enix worked with outside parties so much on its development in 2008 that relatively few internally-

developed titles were left on its plate, but this also drove up average review scores—PSP titles CRISIS CORE FINAL FANTASY VII and DISSIDIA: FINAL FANTASY both performed well in the Japanese marketplace. Survey response to the developer has shifted somewhat since 2007, causing Square Enix's average reputation score to drop to 7.5 from 8.7.

26) GAME FREAK

[POKEMON PLATINUM]

» The Tokyo developer best known for POKEMON released only one title in 2008: the DIAMOND/PEARL remake POKEMON PLATINUM, which lived up to the series' reputation with an 84% average review score. That title came out only in Japan in 2008, but those sales alone amounted to over two million copies and made the title one of the top-selling games in that territory.

25) NEVERSOFT ENTERTAINMENT

[GUITAR HERO: AEROSMITH]

» Neversoft's second year with the GUITAR HERO series saw continued strong sales year round for this Los Angeles-based developer. Reputation and detailed survey scores slid a bit this year, yet readers called the Activision-owned studio "great to work with" and "stable, reliable, professional."

24) MAXIS

[SPORE]

» SPORE, the second-highest-selling game on PC in 2008, gave Maxis a boost to #22 this year. Spinoffs of the new IP and continued sales of Maxis-created original THE SIMS SKUs bolstered Maxis' commercial performance, and for next year, interesting questions remain as to what form a Will Wright-less Maxis will take.

23) VICARIOUS VISIONS

[GUITAR HERO WORLD TOUR]

» Located in a suburb of Albany, NY, this Activision subsidiary makes a respectable showing on the back of strong sales of GUITAR HERO WORLD TOUR (Wii) and ON TOUR (DS), plus strong marks from survey respondents with direct experience of the developer, which gave VV a spot just inside the top 25. Reputation survey scores were less kind, coming out to an average of 6.5, but commenters called the developer a "very well-run studio" and an "awesome environment for working."

22) LIONHEAD STUDIOS

[FABLE II]

» Lionhead, based in Guildford, U.K., released FABLE II this year, which became the best-selling RPG for the Xbox 360, and the 89% average review score the game earned gave



Microsoft-owned Lionhead the seventh-highest overall score on our survey. Commenters seemed pleased with how *FABLE II* “deliver[ed] on [its] promise,” and one claimed it was “a big jump from the disappointing *FABLE*.” A respectable 8.2 average developer reputation score bore this out.

21) MEDIA MOLECULE

(LITTLEBIGPLANET)

» Also located in Guildford, U.K., this new developer makes a virgin showing on our ranking on the heels of the release of its acclaimed and award-winning *LITTLEBIGPLANET*. *LBP* sold over a million copies worldwide and earned a 95% review average. It also earned Media Molecule quite a bit of goodwill, judging by the average 9.0 score survey respondents gave the developer on our reputation survey.

20) HARMONIX

(ROCK BAND 2)

» Boston-headquartered Harmonix’s mainstay *ROCK BAND* and its expansions remained strong in 2008. The developer received quite a few favorable comments from our survey, and readers praised Harmonix for “thinking of the fans in almost every situation.” Several commenters had praise for quality of life at the dev, and one in

particular called the studio “one of the best companies I have had the chance to work for.”

19) KOJIMA PRODUCTIONS

(METAL GEAR SOLID 4)

» *METAL GEAR SOLID 4*’s resounding success gave this Tokyo-based Konami studio its first appearance in our ranking. *MGS4* sold over a million copies in its first day of release and went on to sell more than three million in 2008. Plus, it was a critical darling and received a whopping 94% average review score. Readers took note of the Hideo Kojima-headed studio’s success, praising it for leading “the way in cutscene animations and engine development.”

18) KOEI JAPAN

(OPPOONA)

» Those non-*WARRIORS* titles that Koei developed in-house in 2008 saw few releases in the U.S., but the publisher/developer’s combined Japanese and Western schedule encompassed 22 titles in various genres, which sold respectably in Japan. Those titles that did see U.S. release brought in a 68.7% overall Metacritic ranking. The Yokohama studio has seen a downward slide in goodwill in recent years, and it received a reputation score of 6.6 from readers.

17) EA REDWOOD SHORES

(DEAD SPACE)

» A string of *SIMS*-related expansions and spin-offs sold very well for EA’s Redwood City campus, allowing the developer to move up from last year’s spot at #21. In particular, survey respondents praised Redwood Shores’ “incredible work” on the multiplatform *DEAD SPACE*, which sold respectably and pleased critics.

16) NAMCO BANDAI GAMES

(SOUL CALIBUR IV, TAIKO NO TATSUJIN)

» This merged publisher releases many games developed out-of-house, but still retains formidable in-house talent. 2008’s *SOUL CALIBUR IV* lived up to expectations critically and commercially, and DS and Wii versions of the *TAIKO NO TATSUJIN* series did particularly well in Japan. Critical reception to Bandai Namco’s games improved to 69% over last year’s 61%, but a compressed release schedule and slightly lower reputation scores caused the company to fall one spot from its 2007 rank.

15) INSOMNIAC GAMES

(RATCHET & CLANK FUTURE: QUEST FOR BOOTY, RESISTANCE 2)

» 2008 sequels to the Burbank, CA developer’s best-known franchises kept Insomniac on the list this

year, and average review scores rested at 81%. Survey commenters gave glowing praise similar to last year’s remarks—employees of the company called it a “fantastic place to work,” while external developers noted “consistently high quality titles every year.”

14) BETHESDA GAME STUDIOS

(FALLOUT 3)

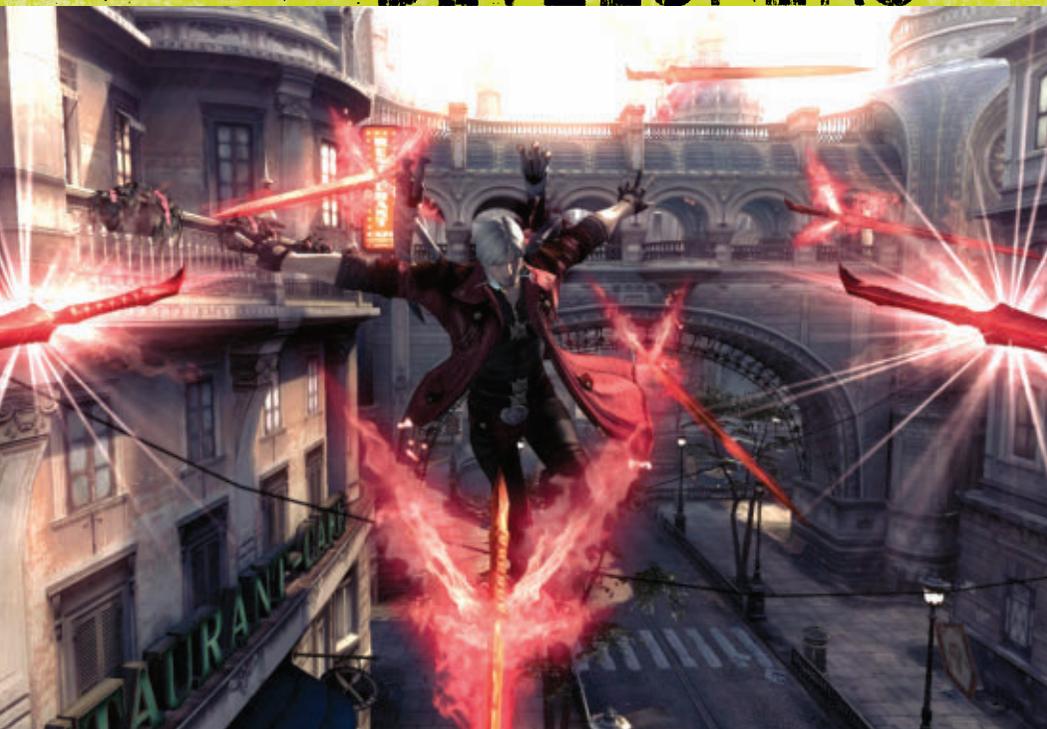
» This Maryland developer’s reinvention of the *FALLOUT* franchise became a massive critical and commercial success and pushed Bethesda up to #14 from last year’s #26 slot. Bethesda scored an 86% overall review average and some none-too-shabby marks from our survey respondents, who praised the developer’s “detailed story work,” “excellent ambition,” and “quality large open world gameplay.”

13) TRAVELLER’S TALES

(LEGO INDIANA JONES, CHRONICLES OF NARNIA: PRINCE CASPIAN)

» Warner Bros.-owned developer Traveller’s Tales knows how to work a good thing. *LEGO INDIANA JONES* and *LEGO BATMAN* have brought the English house further improved sales and review scores (from 66% to 72%) compared to last year, bringing Traveller’s Tales all the way up to #13 from #31. TT also expanded its

TOP 50 DEVELOPERS 2009



lineup from 2007, diversifying with a CHRONICLES OF NARNIA: PRINCE CASPIAN licensed multiplatform release. Survey commenters noted how TT's LEGO titles "continue to be better crafted and more fun to play with the whole family."

12) CAPCOM OSAKA

(MONSTER HUNTER FREEDOM UNITE)

» Capcom ended up with the sixth-best slot in sales this year, powered by massive Japanese sales of MONSTER HUNTER titles (including over 2 million of MONSTER HUNTER FREEDOM UNITE alone) and Western sales of DEVIL MAY CRY 4. Lower review scores kept the developer down a bit compared to last year, and reputation scores were not as high (down to a 7.7 average from 8.7), but this juggernaut keeps moving forward.

11) HAL LABORATORY

(SUPER SMASH BROS. BRAWL, KIRBY SUPER STAR ULTRA)

» Tokyo-based developer and Nintendo second-party HAL developed one of the best-selling Wii games this year. SUPER SMASH BROS. BRAWL sold over six million copies worldwide in 2008 and managed an average review score of 93%. POKEMON RANGER: SHADOWS OF ALMIA and KIRBY SUPER STAR ULTRA also sold well for HAL across all territories.

10) TREYARCH

(CALL OF DUTY: WORLD AT WAR)

» Treyarch rides in on the strength of CALL OF DUTY: WORLD AT WAR, one of the top ten best-selling games of 2008. This Santa Monica subsidiary of Activision received lower-than-average reputation survey marks, but COD's sales across four platforms—generating the fourth-highest sales of any developer on this survey—and a respectable 77% review average makes Treyarch nothing to sneeze at.

9) EA TIBURON

(TIGER WOODS PGA TOUR 09, MADDEN NFL 2009)

» EA's sports-focused Florida campus held even with last year's #9 position. The studio inherited TIGER WOODS PGA TOUR from EA Salt Lake, and Tiburon's 2008 multiplatform iteration of the series further boosted sales, alongside the latest entries of MADDEN and NCAA FOOTBALL. Tiburon ranked #7 overall in sales, yet received some less-than-favorable marks from our survey respondents.

8) EPIC GAMES

(GEARS OF WAR 2, UNREAL TOURNAMENT III)

» GEARS OF WAR 2's "epic" sales and sterling critical reception,

along with UNREAL TOURNAMENT 3's none-too-shabby performance, pushed this Raleigh, NC developer up ten spots this year. Commenters had even more praise than last year: one mentioned how "[the company's] engine empowers a lot of developers" and another called Epic "the best studio to work for ... anywhere!"

7) VALVE

(LEFT 4 DEAD)

» The runaway hit zombie shooter LEFT 4 DEAD assured that Valve had a place on our chart this year. Both versions of L4D scored an 89% review average and gave Valve the seventh-highest overall review score on this survey. The multiplayer FPS also sold over a million copies in 2008 alone, and the PC version was one of the top-selling games of the year for the platform.

6) KONAMI

(PRO EVOLUTION SOCCER, CASTLEVANIA: ORDER OF ECCLESIA)

» Tokyo-based publisher/developer Konami held rock-steady with its 2007 position. Perennial sales of PRO EVOLUTION SOCCER titles and a healthy and diversified plate of 28 releases—including many licensed titles, new DANCE DANCE REVOLUTION and BEATMANIA entries, and a new

CASTLEVANIA—kept Konami buoyant. Review scores and reputation scores from our respondents both fell slightly this year, but the developer received higher specific survey marks from those that had worked with the studio.

5) EA CANADA

(FIFA SOCCER 09)

» EA's Burnaby, BC studio released four fewer titles than 2007's 35, but the developer's release schedule is still impressive. FIFA SOCCER 09 broke U.K. sales records, selling over two million copies in that territory alone, becoming the top-selling console title in the region according to GfK-ChartTrack.

4) ROCKSTAR NORTH

(GRAND THEFT AUTO IV)

» The release of GRAND THEFT AUTO IV across three platforms propelled this Scottish development house high onto our ranking for the first time. An average review score of 95%—the highest on our list—and the fifth-highest sales of any developer in 2008 gave Rockstar North the #4 spot. Commenters gave the developer a slightly lower reputation score of 7.6, but one notably commented that "GTA4 helped bring the games-as-art debate to the general public."



1) NINTENDO

(WII FIT, MARIO KART Wii)



» The current hardware market leader holds onto the top spot for a second year in a row, thanks entirely to absolutely massive retail sales that were matched by none. Nintendo released four fewer in-house created games in 2008—eight compared to 2007's twelve—yet perennial sales of its Wii and DS powerhouses MARIO KART Wii, Wii FIT, and Wii PLAY—along with slightly higher review scores [72% overall]—were more than sufficient to keep the Kyoto company's internal development house on top.



3) UBISOFT MONTREAL

(RAINBOW SIX VEGAS 2, FAR CRY 2)

» Brisk sales and a beefed-up schedule of twenty-five releases in 2008 gave Ubisoft's Montreal campus the number-three slot, up from #12 in 2007. Ubisoft titles also gained a more favorable critical reception in 2008—its average review score rested at 71%, up seven points from the previous year. RAINBOW SIX VEGAS 2, FAR CRY 2, and ASSASSIN'S CREED were all successful critically and at retail, and commenters were mostly favorable to the developer, calling it "not perfect," but "not afraid to try new things."

2) BLIZZARD ENTERTAINMENT

(WORLD OF WARCRAFT: WRATH OF THE LICH KING)

» Even though this Irvine, CA-based fan favorite only released one title in 2008, and an expansion pack at that, sales of WORLD OF WARCRAFT: WRATH OF THE LICH KING and legacy titles gave Blizzard the #2 spot in sales and the #2 overall position, up from #3 in 2007—impressive even without its massive subscription revenues. Respondents gave the developer the highest marks of any other, and glowing comments, noting how the company "consistently lives up to expectations."

	NAME	OVERALL	AVERAGE REVIEW	GAMES DEVELOPED IN 2008	REPUTATION
1	NINTENDO	1	72%	8	8.1
2	BLIZZARD ENTERTAINMENT	2	91%	1	8.8
3	UBISOFT MONTREAL	3	71%	25	7.9
4	ROCKSTAR NORTH	4	95%	3	7.6
5	EA CANADA	5	71%	31	7.0
6	KONAMI JAPAN STUDIO	6	70%	28	7.5
7	VALVE	7	89%	2	9.0
8	EPIC GAMES	8	87%	2	8.1
9	EA TIBURON	9	71%	25	5.6
10	TREYARCH	10	77%	6	7.0
11	HAL LABORATORY	11	77%	4	8.5
12	CAPCOM OSAKA	12	71%	14	7.7
13	TRAVELLER'S TALES	13	73%	19	8.0
14	BETHESDA SOFTWORKS	14	86%	4	8.3
15	INSOMNIAC GAMES	15	81%	2	9.1
16	BANDAI NAMCO GAMES	16	69%	17	8.0
17	EA REDWOOD SHORES	17	73%	15	7.7
18	KOEI JAPAN	18	69%	22	6.6
19	KOJIMA PRODUCTIONS	19	94%	1	8.6
20	HARMONIX	20	77%	8	8.1
21	MEDIA MOLECULE	21	95%	1	9.0
22	LIONHEAD STUDIOS	22	89%	1	8.2
23	VICARIOUS VISIONS	23	73%	6	6.5
24	MAXIS	24	71%	5	8.0
25	NEVERSOFT ENTERTAINMENT	25	75%	5	8.4

TREVOR WILSON is a web developer, freelance journalist, occasional game developer, and amateur photographer based in Salt Lake City, Utah. Email him at twilson@gdmag.com.

AWARD-WINNING GAMES
AWARD-WINNING DEVELOPER

BURBANK

DURHAM

WHAT ARE YOU WAITING FOR?

INSOMNIAC
GAMES

www.insomniacgames.com/careers

facebook

Become a fan:
Insomniac Games



Follow us:
insomniacgames

infinite

DR. KJARTAN EMILSSON AND CCP TEAM

s p a c e

MOST OF THE LARGER MASSIVELY MULTIPLAYER ONLINE GAMES USE SEPARATE INSTANCES, OR SHARDS, of the game's universe in order to manage player populations and server issues. We feel that a single shard should be the natural choice of any MMO developer, and that's what we do with EVE ONLINE. When you ask the question "Why a single-sharded architecture?" it's also informative to look at the deeper question: "Why have shards?" There are two main reasons why a developer chooses a sharded implementation of a game—lack of content and technical challenges. These are actually inter-related.

CONTENT

» Most current MMOs take place in environments essentially limited by strong physical constraints: avatars moving across earth-like landscapes or within enclosures like buildings. Furthermore, within these specific environments, players are confronted with a multitude of scripted activities such as quests and NPC encounters that only take place there. The most limiting physical constraint concerns avatar density. This is both a technical problem and a usability problem. Players do not want to constantly navigate an overcrowded environment. In order to keep avatar density within reasonable limits, you either need a very large playing field or a limitation on the number of players in a given field. Both of these options are restricted by the amount of content you can design, and since content is the biggest cost in modern games, this quickly becomes a financial limitation.

The obvious solution is to have procedurally-generated content, such that you can essentially have a playing field as large as you want. The drawback with that approach is that you will most likely never reach the same artistic level displayed in hand-crafted environments, and scripted activities might become repetitive and lack context.

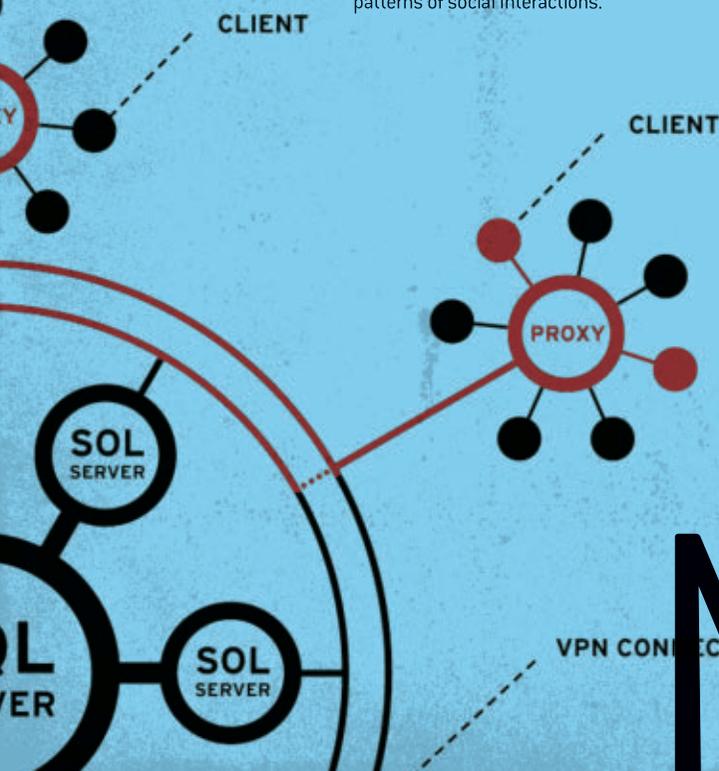
The real solution to this problem is to embrace the notion that in an MMO, just like in any other social network, players are the content. Once that is accepted as a fundamental design guideline, it becomes easier to navigate the challenges involved in creating and maintaining a single shard architecture and actually gives the advantage to that design model.

Looking more closely at this assumption, we can identify two types of content generated by people: material content, which we describe as persistent user-created assets within the world, and social content, here considered as persistent patterns of social interactions. »



an
argument
for
single-
sharded
architecture

in
MMOs



infinite space



The first one is easy to comprehend. However it is implemented, persistent player-created content can populate large playing fields and make the world more “meaningful” for large groups of other players. This is the case in RTS games, where the backdrop may be relatively bland and automatically generated.

In EVE, for example, a lot of the high-end gameplay revolves around conquest and control of territory in unregulated areas of the map. By choosing where to place primary space stations, players shape the topography of the strategic battlefield. In selecting the position of those stations’ supporting starbases and the configuration of their offensive and defensive systems, they shape the tactical context in which critical battles occur.

The second type of content, social content, is the most potent, but also requires careful design. The field of social interaction encompasses a very wide range of activities and concepts:

- **Pure socialization, such as chat and messaging**
- **Combat between players or cooperative combat against the environment.**
This scales all the way from 1v1 combat to conflicts between factions numbering thousands of players
- **Economic activities**

With socialization, the main “content” is the social tapestry that materializes in buddy lists, membership of player associations, or guilds and forums. For all of these, a single shard adds to the richness of the content because players don’t need to be split between servers; they can discuss issues and share experiences that arise in the shared world that are relevant to the whole player base rather than a specific server—essentially giving them a shared history as a whole

society rather than a disjointed one based on smaller server populations. Furthermore, gaining fame becomes much more rewarding due to the size of the audience, thus strengthening the impetus to do so. The technical challenges to creating a single shard communication infrastructure should not be underestimated and we address them later in this article.

Early last year, as part of CCP’s efforts to nurture the development of the functioning society formed by EVE’s player base, a democratically-elected player council was formed to act as representatives of player interests in the development process. The single-sharded nature of the game enables the formation of a single coherent society and makes it much more likely that the elected players will form a representative cross-section of the interests of the electorate. Because everyone is sharing a single server, and thus a single social context, the community has a common baseline for discussion and debate, and famous figures are more likely to be known to the entire player base rather than just fragments thereof.

We have also seen the benefits of single-sharding with combat, in the form of increased complexity of conflicts in terms of both space and time. This heightened

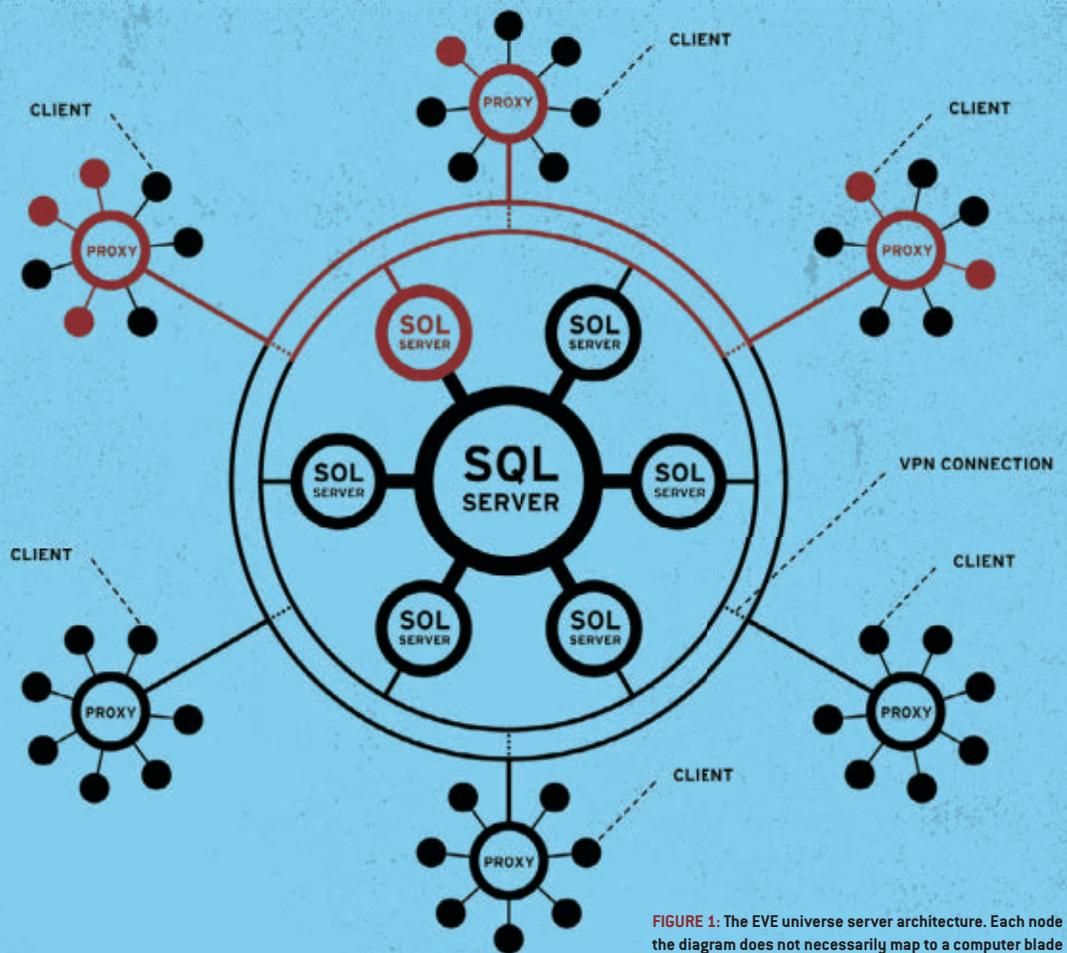


FIGURE 1: The EVE universe server architecture. Each node in the diagram does not necessarily map to a computer blade in the server cluster. Blades are typically multiprocessor and multicore and many of them are configured to run multiple proxy or solar system (SOL) nodes.

complexity results in a variety of roles within a conflict and less routine in waging it. In EVE we have had wars involving tens of thousands of players, pitched against each other for several years. Whether a player contributes as a grunt on the front, a middle-man in logistics or as a long term strategic planner is up to them. The size and longevity of such conflicts clearly sets them apart as true content. Instead of being ephemeral and soon-to-be-forgotten skirmishes, these have become epic stories that fascinate players and build up reputation and true in-game power. Again, providing for the sheer scale of these encounters involves technical challenges, both on the server and client side.

For instance, the current pivotal war in EVE (referred to without apparent irony as "The Great War") is generally agreed to have started in late 2005 when an old, established alliance of player corporations decided to eliminate a new but growing power for political and ideological reasons. The conflict snowballed, expanding its theatre of war, ebbing and flowing as some groups surrendered or collapsed and fresh ones joined the fray. Today—just over three years later—it appears to be entering its final stages, with many ancient grudges and vendettas being resolved. The organizations directly involved in the fighting at this stage contain, between them, just over 30,000 player characters. And, just as many groups were drawn into the fighting by the opportunity to settle old debts from previous wars, events from these three years of continuous warfare will likely fuel future conflicts.

Finally, there is the economy. Even though many people don't realize it, the economy is truly the pinnacle of social interaction. This is of course assuming that it is player-driven rather than being dictated by the designers. It is only in a truly player-driven economic environment that price fluctuations of items and commodities realistically start to reflect the sum total of the socio-economic landscape of the world. The market becomes a mirror of the activities of all

participants in the game and it acts to change players' actions by its reflection.

Such a player-driven system doesn't strictly require a single shard to function, but it is catalyzed by the extended size inherent to single-sharding. A small economy will be manipulated by a few strong players and exposed to large fluctuations and instabilities. The larger the economy gets, the more resilient it becomes. Once beyond those instabilities, it truly starts to reflect the macro-economic landscape of the game-world, becoming an all-pervading, autonomous, and ever-changing mass of social content that no designer could ever think of hand-crafting.

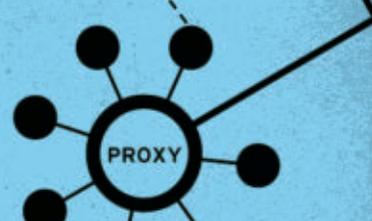
EVE ARCHITECTURE OVERVIEW

» In line with our design goal to have a single shard in EVE, we quickly decided to go with procedurally generated content for the physical landscape of the game's universe. Fortunately, space lends itself rather well to that. The natural distance scales in a typical galaxy make aggregation points emerge naturally, so that the whole logic of a solar system can easily be run within one process space. For clients this goes down to an even finer granularity, as they only need to physically simulate their closest surroundings.

But apart from that, the whole back-end logic abstracts the notion of servers, such that requests within specific game logic (either on client or back-end) are transparently mapped to different nodes depending on context. We have a distributed logic running on top of a cluster of nodes. All data manipulated by these nodes is read and written to a single database that binds the whole world together (see Figure 1).

Some might argue that a single solar system thus acts as a kind of shard, but that is not correct. Any player from the global player base can enter any solar-system, and all economic activities will have immediate repercussions to the whole economy. Furthermore, all the social structures mentioned above are truly global and transparently cross system boundaries. The solar

infinite space



system does introduce "crowding" problems, but these are problems that cannot really be solved except by game design.

For all practical purposes, the whole cluster acts and feels like a single process space. But this is not to say that this approach hasn't had its challenges. Here are a few.

GAME DESIGN CHALLENGES

» As we noted earlier, player density can be a real challenge from a technical perspective as well as a game usability perspective. In EVE, we run into two general kinds of harmful player clustering which cause design headaches. The first is what used to be called the "Yulai problem."

A few smart players determined that the Yulai solar system was particularly well connected to the various areas of the star cluster, and started selling their wares there in bulk. Those few canny marketeers on their own weren't a problem, but it didn't stay that way for long—as more buyers visited the system more sellers set up shop there. As more sellers flocked in it became more desirable to buy there. Pretty soon it seemed like the entire playerbase was shopping in Yulai, and that system's population made up a noticeable percentage of the total online playerbase at any given time.

In an effort to curb the growth of this trade hub before it started causing serious server issues, we made some changes to the map by shifting jump routes around to lessen the appeal of Yulai. Within a few months the "Yulai problem" became the "Jita problem," as players figured out the new best location and moved all their business there instead. The formation of such hubs seems to be an inherent human phenomenon, and while we still have regular design discussions about effective ways to distribute market activity more evenly without distorting the market itself, it's eventually something that we solved with hardware and software solutions.

The second clustering issue is almost the polar opposite to the problem of market hubs—that of huge battles for strategic objectives. Whereas hubs are permanent and predictable fixtures, fleet-sized combat tends to be transitory and unpredictable. In our case at least, the emergent gameplay that delivers such value



to implement this in practice in such a way that it's actually beneficial for military commanders to split their forces under the majority of common circumstances.

PROGRAMMING CHALLENGES

» A large-scale single-sharded environment is not without its unique challenges on the code side either. Here are several of the problems we faced in our implementation.

Running out of memory. Over the years, maximum memory usage on the nodes that run EVE's solar systems (Sol nodes) have been steadily increasing with the increased population and expansions to the game. In particular, solar systems such as Jita have been pushing the process memory usage up. When EVE launched, it ran on 32-bit Windows Server 2003, giving us a virtual memory limit of 2GB for each process. Later we upgraded to a 64-bit OS and this increased the limit to 3GB. But sometimes even this would not be sufficient, so we decided it was time to make the server processes 64-bit.

Initially we were slightly worried, as the physics simulation has to stay in sync on the client and the server. We were uncertain whether it would drift apart due to different code being generated for the complex mathematics involved. Eventually careful testing revealed that this would not be a problem and that the algorithms were numerically stable in such a mixed environment.

The release of the 64-bit binaries was without incident. In fact, performance increased quite a bit, due to the better optimization possible with a larger number of registers in 64-bit mode. Baseline memory consumption did rise, of course, because all pointers were now twice the size. However, we were finally free from the virtual memory constraint of 3GB. Now each process can allocate as much as it wants and never run out of address space.

As for physical memory, it turns out that the 4GB of physical memory on each machine—which typically runs two Sol node processes—is quite sufficient (as seen in Figure 1). Most of the allocated virtual memory lies dormant and there is little paging. We still haven't seen a node die because of page thrashing.

Programming asynchronous systems and distributing execution. Running logic on top of a cluster of distributed nodes means that a typical function call might cross process boundaries and even go across the public

internet before returning. This calls for a lot of asynchronous programming, which is notoriously cumbersome to implement. This is where Stackless Python comes to the rescue. At CCP we firmly believe that we need to make programming as simple and intuitive for the game programmer as possible. We decided early on that in order to create a complex game such as EVE we would need some kind of multi-threaded programming. We also recognized that using scripting for game-level code was necessary simply to get things done. We were very fortunate to come across Stackless Python at a very early stage in development and ended up using it for all game logic.

Stackless Python is a variation of the Python programming language that introduces the concept of "tasklets" and "channels." A tasklet is a thread of execution that is independent of OS threads, and tasklets communicate and synchronize with the help of channels. They consume no more memory than their execution stack and don't require kernel mode switching,

We are running a real-time game on a huge scale which means that speed is top priority, and we can use tricks which would not be allowed in other situations—similar to the operations of a bank (although recent events in Iceland might suggest otherwise).

regularly compels huge political power blocs numbering thousands of players to make spirited attempts to beat the life out of their rivals. Without warning a particular system's population will shoot up from maybe half a dozen players to over 1200, generating a huge spike in server load before rapidly dropping back down to its original level. The abstract design solution to this is to spread the combat out across multiple systems simultaneously, but this is directly opposed to the ageless military principle of "hit them with everything we've got." It is an ongoing challenge to figure out how



Unreal Technology News

by Mark Rein, Epic Games, Inc.

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

Epic's Unreal Engine 3 won Game Developer Magazine's Best Engine Front Line Award for three consecutive years, and it was inducted into the Hall of Fame this year.

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

Upcoming Epic Attended Events:

GameHorizon Conference
Newcastle, England
June 23-24, 2009

Develop Conference
Brighton, England
July 14-16, 2009

SIGGRAPH
New Orleans, LA
August 4-6, 2009

GDC Europe
Cologne, Germany
August 17-19, 2009

Please email:
mrein@epicgames.com
for appointments.



WHO NEEDS RETRACTABLE CLAWS WHEN YOU HAVE UNREAL ENGINE 3?

Activision's Raven Software recently shipped *X-Men Origins: Wolverine* for PC, Xbox 360 and PlayStation 3 in tandem with the 20th Century Fox film starring Hugh Jackman. It's the first Unreal Engine 3-powered game from the studio, although *Singularity* is coming out, also from Activision, later this year.

"We had struggled to make *Marvel Ultimate Alliance* next-gen, and then we saw *Singularity* and we were like, 'Holy crap, that's the type of tech we want to use,'" said Dan Vondrak, project lead on *X-Men Origins: Wolverine* at Raven Software.

The new action game puts players in control of one of Marvel's most popular characters, Wolverine, and offers a full array of abilities and attacks ripped straight from the comic books.

Vondrak said that during production, UE3 allowed the artists to jump ahead of the rest of the team. They were able to create huge jungles with sun rays coming through and leaves blowing and water puddles.

"Working with Unreal allowed us to add depth to the game. That's why we were able to create a Wolverine model with three layers of regeneration. We have the skeleton, the meat, and the muscle and skin, plus the clothing on top of that. That's all made possible using Unreal materials and shaders. It's really powerful when we coupled it with our smart tech guys who put everything together to make it work."

Vondrak said the designers utilized Unreal Matinee to create the bigger moments from the game, some of which were original and others were expanded from the movie.

Matinee allowed the team to create action sequences featuring moving trucks and flying helicopters. While the final animations were done by animators, Unreal aided them in getting everything just right – like Wolverine's perfect landing atop a whirring helicopter in mid-air.

"The Kismet tech is really powerful," added Vondrak. "When you look at what Epic has been able to do with this technology with the *Gears of War* games and then look at *Wolverine*, you can see the type of meaty combat that translates across genres.

"Kismet allowed us to throw all of these huge sequences into our game, which gives players a very cinematic experience. All of these set pieces like when Wolverine is in the air skydiving from helicopter to helicopter were created by our designers using Kismet."

One example of Matinee, Kismet and AI all working in tandem can be seen in the epic battle between Wolverine and the 100-foot-tall Sentinel robot.

Players will pit the tiny, but powerful, Wolverine against this monster in a three-pronged battle that will start on the ground and then take to the air.

Vondrak said that all of the sequences, including what traditionally would have been cut scenes, were made playable thanks to Unreal.



X-Men Origins: Wolverine by Raven Software

"Unreal Engine 3 was just fantastic to work with," said Doug Smith, senior technical artist on *Wolverine* at Raven.

"One of the challenges with *Wolverine* is that we wanted to make a game that's true to Wolverine without spending a ton of time building up our tech. The Unreal Engine was a great stepping stone to make that happen quickly," Smith remarked.

"It was a great way to actually give something to artists and designers that was mature and fully flushed out. We knew we could make a good-looking game if we worked it right, and I loved working with Unreal."

Thanks to Raven Software for speaking with freelance reporter John Gaudiosi for this story, which will be posted in full at www.unrealtechnology.com.

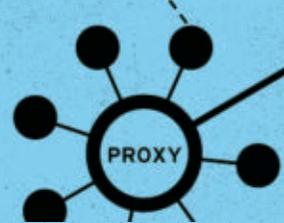


For UE3 licensing inquiries email:
licensing@epicgames.com

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

WWW.EPICGAMES.COM

infinite space



which makes them very fast (see "Asynchronous Programming" by Javier Blazquez, May 2009). With impunity, a programmer can create a new tasklet to fork off any processing he or she needs to. Tasklets in EVE use cooperative scheduling, meaning that we only switch to another tasklet at known switching points. This virtually eliminates the need for complex locking and synchronization, as is often the case with multithreaded programming.

A particular case where this programming model shines is with I/O. Efficient I/O makes use of a non-blocking interface to the OS. On Windows, this often takes the form of sockets and I/O completion ports: A thread starts an I/O operation and then polls an I/O completion port to see when it is done. But from the programmer's perspective this is extremely complicated and error prone. A programmer just wants to `send()` and `recv()` and not worry about event loops and such things.

To facilitate this, we present the Python programmer with a blocking I/O interface that is in fact only tasklet-

blocking. When a tasklet executes a `socket.recv()` function we issue an asynchronous I/O request to WinSock. We then suspend the tasklet, allowing another tasklet to run. Later, when we notice that an I/O request has completed, we prepare the result for the blocked tasklet and make it runnable again.

This way we can program discrete pieces of game logic with their own straightforward execution paths that just behave as you would expect, while behind the scenes we suspend and reschedule their execution, making good use of the operating system's resources.

Designing and maintaining a real-time database. We chose a relational database to be at the center of the server architecture. The key focus with our database design is to keep things simple, as we strongly believe that simple is incredibly powerful, and easier to maintain. All important DB usage goes through stored procedures to make things as efficient as possible. There is not a single trigger or a cascading foreign key in the database, simply because we don't want complex magic happening behind the scenes. We want things to be visible from the source code, whether it is a child record delete or an update of related records.

We also have strict coding guidelines for DB code that we have generated over the years. We have in-house experts reviewing DB code checks and forcing developers to fix code not done correctly. Monitoring the database usage is extremely important, and we store all kinds of statistics for that purpose in the database as well. We track how often each stored procedure is called per day, how often an index is scanned, and so on. Looking at such statistics usually tells us right away if a developer has committed code

that has sent the database into a frenzy. We are running a real-time game on a huge scale, which means speed is top priority, and we can use tricks which would not be allowed in other situations—similar to the operations of a bank (although recent events in Iceland might suggest otherwise). One example of this would be when we use "read uncommitted" when loading a solar system configuration, knowing there won't be any inserts or updates to the data we are reading so we can allow ourselves to read with no locking. Another trick would be that we most often only need to allow the user to filter data within a day. For example, we do not need to allow a user to select all records from the player journal between 14:00 and 15:35; it is sufficient to allow filtering by only a date. In that case we simply need to keep track of the clustered key at 00:00 every day and use that in our queries. This means we don't need to index on date/time columns in every nook and cranny, making things faster and slimmer.

While having only one database creates performance challenges, it also makes some things easier. Having a distributed database system, where one database stores all characters and other databases store each shard, brings all kinds of complexities that we don't have to deal with. There is no need to replicate data between databases, nor to call multiple databases nor move characters and belongings between sharded databases and so on.

OPERATIONAL CHALLENGES

» With the performance demands above, maintaining excellent performance of the database hardware can be quite a task. It is the central point of our virtual world, so any latency or slowness at this level reflects across the entire universe of EVE. Ensuring that the database servers have headroom in all of the key performance areas is critical—the main bottleneck that we have had to overcome is I/O performance of database storage.

Over time, we have successively moved away from traditional fiber channel disk array storage to much faster solid state storage devices. Initially this was done for our hottest tables only, but we have recently moved the entire database to solid state drives. This approach has helped us to maintain an environment of virtually no database lag, and we still have a huge ability to scale up.

An area that has required constant attention and work from our operations team is how much we can break down a heavily laden area of the game world into chunks, and spread this load over multiple nodes. Currently at this level we can allocate a maximum of one server node to power an entire solar system, and one server node runs mostly on a single CPU core, splitting off networking and other asynchronous operations to another core.

The design headaches that occur around the "Jita problem" mentioned above are specifically where we start to run into this limitation. When thousands of players go into the Jita system, or engage in fleet battles, we have often had trouble finding enough CPU power to handle the immense amount of processing required to keep the game simulation running lag-free. In the server room we keep Jita running on its own dedicated machine—the biggest, meanest blade server that we can get our hands on.

Software-based improvements like StacklessIO and 64-bit server code (see "running out of memory" above) really made a huge difference to our capacity in this area. Last year saw a three-pronged assault in our "War on Lag," where

we rolled out StacklessIO, EVE64, and some top of the line server hardware almost simultaneously. The result was that our capacity in Jita went from around 600 players to 1,200 players—a 100% improvement in capacity in under 6 months. Work continues in this fashion in order to again double this number, because we know that we will need to.

CONCLUSION

» As we have seen, running an MMO in a single shard introduces strains on system architecture, low-level runtime, databases, and operations, and it even affects the game design level. Moreover, as the number of players grow, the strains will show up in different, sometimes unexpected places. As such, the development of a single shard game is a never-ending task, constantly needing innovations and clever solutions to keep it growing. But pushing the limits is also a source of innovation, and leads to discoveries that are both enjoyable from a professional point of view, and also add new dimensions to the player's experience. So the answer to question "Why a single shard?" could simply be "Because it's challenging and rewarding for everybody"—and that is what gaming is all about, isn't it?

EVE Database Statistics



Database size: 1.4 TB
Number of tables: 775
Number of stored procedures: 2,715
Transactions per second: 2,500
Number of DB calls to the database per day: 175 million
Number of inserts per day into the hottest log table: 12 million
Number of items created per day in the items table: 10 million

DR. KJARTAN EMILSSON is managing director, CCP Asia. HALLDOR FANNAR is CTO. KRISTJÁN JÓNSSON is senior software architect. JÖRUNDUR MATTHÍASSON is a programmer/DB captain. MATTHEW WOODWARD is a game designer. JAMES WYLD is a virtual worlds administrator. Email them at ccp@gdmag.com.

**NEW ORLEANS:
ULTRA-ANALOG**



**SIGGRAPH 2009:
MEGA-DIGITAL**

Join some of the world's finest players at SIGGRAPH 2009 in New Orleans.

Exchange insight and inspiration with the animators, artists, researchers, developers, and producers who are creating this year's most amazing experiences. In the city that has been inspiring musical innovation, culinary excellence, visual splendor, and architectural wonder for 300 years. You'll return from SIGGRAPH 2009 with re-energized imagination, renewed skills, and insider information to spark your creativity and surpass your goals for the coming year.

 **SIGGRAPH 2009**
NEW ORLEANS

Conference 3 – 7 August 2009 Exhibition 4 – 6 August 2009
Ernest N. Morial Convention Center, New Orleans, Louisiana

www.siggraph.org/s2009



Sponsored by ACMSIGGRAPH

BRUCE PHILLIPS

staying

MOST GAMES ARE CHALLENGING BY DESIGN. WINNING EVERY TIME ISN'T FUN, BUT NEITHER IS always losing. The typical user experience is somewhere between these two, with most players experiencing some degree of failure. Players will lose races, blow up, fall to their deaths, get lost, or fail in thousands of different ways. Some will persevere and continue to play, while others will get discouraged and give up.

As we game developers seek to expand our audiences, some traditional methods of keeping players engaged are becoming less effective (according to Microsoft's databanks, which I use as the main source throughout this article). Fortunately, we can do some relatively simple things to motivate players.



Despite these attempts to balance difficulty for a wide range of people, the players will still experience failure. More importantly, many of these folks will stop playing because of these failures. It's rare for people to leave a restaurant because they don't like the food, and it's not too common for people to walk out of a movie because it's bad—but game players do put down the controller and leave the game all the time. What's worse, when game players have a negative experience, they are likely to tell their friends, family, and community.

When someone quits a game prematurely, we haven't just lost a player; we've created a detractor.

QUITTER TALK

» How serious an issue is quitting? It's worse than you might have guessed.

Table 1 shows the average Gamerscore completion for each of the top 13 Xbox Live games for 2008. The data was drawn from about 14,000 players. As you can see, even the games with the highest achievement completion rates (FABLE II and CALL OF DUTY 4) had players who, on average,

attained less than half the possible Gamerscore.

This particular sample tends to be more hardcore than the average player, and I would expect the actual completion rates for the entire population to be lower than the numbers recorded here.

Of course, the Gamerscore tells only part of the story. Players could finish a game and do little else, resulting in a low Gamerscore but high completion rates. However, most games award achievement points for completing the single-player campaign.

Table 2 (pg 25) shows how many players finished a sample of the games listed in Table 1 as determined by whether they earned a campaign completion achievement (on any difficulty). For even the most popular games on Xbox Live last year, about 30 percent of players didn't play to the end.

Players don't finish games for many reasons, but no matter what explanations arise, it's also likely that a significant number of players stopped out of frustration and that is what we will discuss here.

What leads some people to persevere after experiencing failure and others to give up? Why do some people anticipate eventual success where others only see continued failure?

There are probably many answers to that question, some of which are out of the game designer's control. However, there are at least two things we can and should do. The first has to do with how we word feedback to players, and the second is related to the goals we provide.

THE PSYCHOLOGY OF FEEDBACK

» Research on motivation, primarily in education, suggests that an important factor for explaining how people respond to failure is their perception of why the failure occurred. Those who believe that their failure is

rethinking feedback^{to keep} players in the game g power

the result of stable factors, such as native ability or intelligence, which they cannot easily change, are most likely to give up or not even try. However, people who believe that a failure is the result of unstable factors that they can change through effort or strategy are more likely to believe they can overcome initial setbacks. The determining factor is the person's mindset about his or her ability.

The same holds true for video games. Players who believe they can learn and master the game persevere, while those who think they lack a particular game-playing ability, or that some other stable factor lies between them and success, are likely to quit.

Fortunately, this is susceptible to change. There are things we game developers can do to encourage a mindset that anticipates success rather than failure. But before we get to that, consider these two studies, both of which illustrate the simple and subtle means through which we can shape players' perception of ability.

M. L. Kamin and Carol S. Dweck (see References) conducted a study in which they had students take a difficult test. After the test, they praised half the students for being smart (the "ability" group) and the other half for their effort or strategy (the "learning" group). The participants were then given the choice of two new tasks to complete: a simple one at which they were likely to succeed but learn little, or a difficult task that would be more interesting but would likely result in mistakes. Most of the ability group chose the simple task, while the learning group tended toward the more difficult task.

As far as psychology experiments go, this was a very simple manipulation. The researchers merely changed a few words in their feedback, which produced significant changes in the students' attitudes.

In another experiment, two Stanford University researchers manipulated the attitudes of participants before a task. Craig Anderson and Dennis Jennings (see References) told half their subjects, prior to having them take a test, that their success on the test was likely dependent on innate ability—either they had the ability to perform well, or they didn't. The other half of the subjects were told that doing well was a matter of determining the right strategy—anyone can do it, but it takes effort.

However, Anderson and Jennings designed the test so that everyone would initially fail (does this sound like a game you've played?). After taking

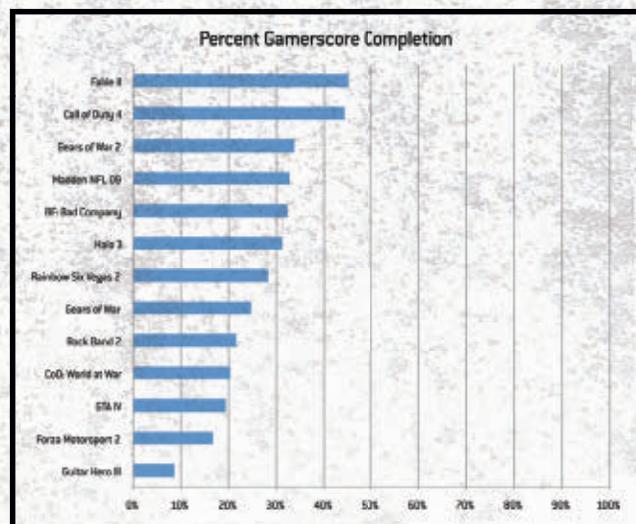


TABLE 1 The percent completion is found by dividing each player's Gamerscore by the total possible Gamerscore for the title; those numbers are then averaged.

the test once, the subjects were asked how they thought they would do on another, similar test. Those who were led to believe that success depended on strategy and effort were more likely to expect future success. Those who believed that success was a matter of ability did not.

Interestingly, this manipulation had an even more dramatic effect. The subjects with the strategic mindset were more likely to have monitored their own methods for completing the tasks so that they were able to modify them in subsequent attempts. That is, they were able to learn from their experiences. The participants with the ability mindset did not monitor their strategies and therefore did not learn as much as the other group from their experiences. In short, the manipulation affected both the participants' anticipation of success (or lack thereof) as well as what they learned when taking the test.

staying power

PERCEPTION MANAGEMENT

» Intuitively, these results make sense. If people believe that success is dependent on ability, then no matter how much effort they expend on the task, they believe they are going to fail again. When people say things like, “I can’t cook,” or “I can’t draw,” or “I’m no good at first-person shooters,” they don’t typically then sign up for a cooking class or begin carrying a sketchbook everywhere or practice playing HALO. “Why bother trying to improve if you don’t have the innate talent?”

What is less intuitive, and what we need to leverage as game developers, is our ability to manage expectations and mindsets.

To do this, we have to change our mindsets as well. I strongly suspect that most designers spend very little, if any, time considering what the player experience should be in the 10 seconds between a failure event, such as dying or losing a race, and the moment when play resumes—or worse, the moment the player quits. (There are some notable exceptions, including TEAM FORTRESS 2 and CALL OF DUTY, which I discuss in the following section.)

Given the opportunities to keep players involved and motivated, it’s unfortunate that game developers rarely take advantage of these moments. In fact, they might be the most important 10 seconds of your game. While we spend weeks creating a few seconds of a cut scene and hours perfecting a texture, we spend very little time considering and implementing appropriate feedback at those very moments when a player decides whether to continue playing (See also research summarized in “GDC: Top 10 Video Game Research Findings,” by Jill Duffy, www.gamasutra.com/view/feature/2645/gdc_top_10_video_game_research.php).

Let’s take a quick look at some of the player experiences surrounding

defeat. Historically, these moments have been brutal. The “game over” screens for most arcade games were terse, bordering on insulting: “You Lose,” “Game Over,” “You’re Dead.” How far have we come since then? Not very. Most games fade to black, switch cameras to provide a view of the corpse, or simply pop the player back to a save point.

But we still see ghosts from the arcades. Some of these are intentionally reminiscent of the past, though most are just thoughtless designs. Language such as “You Suck” (yes, I’ve seen it), “Failed,” and “Game Over” encourages players to put down the controller and do something else.

I’m not suggesting we swap in touchy-feely or overly encouraging language. In fact, there’s a fine line between providing appropriate feedback and being patronizing. What we should be doing is focusing on the player’s actions and emphasizing improvement.

EXAMPLES OF FEEDBACK

» There are several good examples of feedback in games that teach strategy. The most common are the in-game hints players get at appropriate moments, such as after a death. The CALL OF DUTY series has been doing this for a while, leaving a message for players when they die from a grenade in MODERN COMBAT, for example, or are stabbed in WORLD AT WAR. These messages ask players to focus on developing strategies to avoid dying in similar situations in the future. (Although in practice, these indicators can be frustrating as well, if the player sees a grenade death but feels it was unfair—the message then becomes insult to injury.)

Less common are feedback systems that inform players about their improvement at the game. One great example happens in TEAM FORTRESS 2. When players die, a message informs them about how they did relative

GDC⁰⁹ Europe
www.GDCEurope.com

August 17–19, 2009
Game Developers Conference® Europe
Cologne Congress Center East
Cologne, Germany

The Game Developers Conference® Europe will serve as the premier pan-European developer event in 2009 focusing on Business Management, Game Design (with Art and Audio), Production, Programming, and Visual Arts. In addition, it will also cover the development of Mobile Games.

Visit www.GDCEurope.com for more information.

Supported by
European Games Developer Federation
gamescom
BIU

Cologne

THINK SERVICES

Register with code: **GDCE09** and receive 10% off Main Conference and All Access Passes*

*Discounts cannot be combined with other GDC Europe discounts or other promotions including group registrations and alumni discounts. GDC Europe reserves the right to review and end this promotion before the end of professional discount deadlines or before the end of online registration.

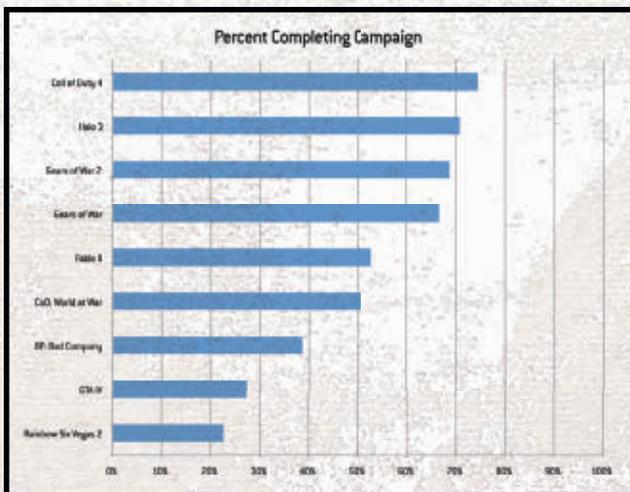


TABLE 2 The bar graphs show how many players earned a campaign completion achievement—in other words, finished the game—for the titles listed.

to previous attempts, for example: “On the bright side... You came close to your record for time alive as a Scout in that round.” The message goes on to indicate how long the player lived that round and what his previous personal best was. This is excellent feedback.

We also may want to shift our thinking about tutorials. Tutorials should not be considered the 10 minutes of instruction players get when they first start playing, or the part of the game we develop at the last minute after we finish making the “real” game.

Tutorials (perhaps we should stop using that word, too, as instruction needn’t simply be text-based information) are the game and should occur over the course of the entire experience. Players are constantly learning to play, right until the end, and we need to provide relevant and informative feedback to them.

When we think about the game this way, we force ourselves to think about what players need to know at each point in the game, when to deliver that information, and how to track the data we need to provide this feedback. Sure, the bulk of what the player needs to know to get up and running happens early, but most good games require players to learn and adapt throughout. We should be doing our part to feed players the information and encouragement they need to keep up with these changes (though the “how” of this could take up an entire article on its own).

CREATING GOALS

» The second thing we need to get better at is creating goals for players. Research has shown that the goals created for people—by teachers, bosses, parents, and game designers—go a long way toward shaping their mindsets about prospects for success and how they respond to setbacks. Most relevant to game developers are two types of goals: performance goals and learning goals.

Performance goals. Performance goals (or outcome goals) represent the most prevalent goal type in video games. There are three defining characteristics of performance goals.

CONTINUED ON PAGE 26



GAME DEVELOPMENT PARTNERS
COST EFFECTIVE | HIGH QUALITY | TIMELY DELIVERY

SPECIALIST IPHONE DEVELOPERS

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

Approved Developers for:

● iPhone / iPod Touch ● Android G1 ● PSP

DEVELOPMENT / PORTING SOLUTIONS FOR MOBILE & ONLINE GAMES

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

Special Attractive & Economical Rates Partner with Us, Today!
Developers of Award-winning games.

E-mail: info@spiel-s.com Website: www.spiel-s.com

CALL FOR PAPERS!

Announcing: **GAME ENGINE GEMS**
Series Editor: **ERIC LENGYEL**

New series covers rendering techniques, shaders, scene organization, visibility determination, collision detection, audio, user interface, input devices, memory management, artificial intelligence, resource organization, and cross-platform considerations.

Submit papers at www.gameenginegems.com

Jones & Bartlett is also pleased to announce the acquisition of **WORDWARE PUBLISHING** COMPUTER BOOKS including best sellers like:





See them all at www.jbpub.com


 An IGDA Partner Program member

staying power

CONTINUED FROM PAGE 25

First, either people achieve the goal or they don't. There is no middle ground. Examples of performance goals include finishing a level in a platformer, getting the "Overkill" achievement in HALO 3, or finishing a race in FORZA MOTORSPORT in under two minutes. Examples of performance goals outside games are grades in school, or medals at the Olympics. There is no reward for progress toward the goal; you don't get half a driver's license for denting only one side of the car.

Second, the criteria for success are typically not defined by the goal-seekers.

Third, performance goals are usually complex activities that encompass a variety of smaller component skills. Passing a driver's license test requires many different skills, such as understanding the rules of the road, parallel parking, and braking safely.

What's wrong with performance goals? While performance goals are pervasive in school, work, and games, research on learning and motivation has shown that they often produce perceptions of lack of ability as well as decreased motivation. This is particularly true in cases where rewards or praise are contingent on successful completion. Further, negative feelings resulting from failed performance

goals are more likely when a person's perception of their ability is already low, as may be the case with novice game players.

Consider this example. A child gets an A on his math test (a performance goal) and his parents tell him how smart he is. Maybe he even gets a reward. These are good parents. It is a popular belief that rewarding and praising abilities in situations like this is good parenting. However, this kind of feedback can also have negative results. If the child's parents have consistently rewarded him for his ability, and because his parents made their praise contingent on a performance outcome (success on tests), it may backfire in situations where performance is poor. He will view his failures, like his successes, as a measure or indicator of ability, and failure equals lack of ability.

Now consider a player who believes he has low game-playing ability. A performance goal, such as completing a level in a shooter, will lower his motivation to continue trying, if he fails repeatedly. When the goal focuses on ability, and the individual believes he does not have that ability, motivation and performance suffer. To extrapolate from the research further, he is less likely to focus on strategies for improvement if he views success as being contingent on a skill or ability he doesn't have.

RESEARCH FINDINGS ABOUT PERFORMANCE VS. LEARNING GOALS

- » Performance goals elicit a failure-avoidance pattern.
- » Performance goals elicit a negative emotional reaction to failure.
- » Single episode failures of learning goals tend not to affect perceptions of future success, while failing at performance goals leads to self-attributions of lacking ability.
- » People who adopt learning goals show a higher rate of success on a task than those who pursue performance goals.
- » Learning goals make people spend more time on tasks.
- » People pursuing learning goals show more persistence in the face of difficulty.
- » Learning goals create a preference for challenge and risk in future tasks.

GDC 09 Austin

September 15-18 2009
Austin Convention Center | Austin, Texas

The Game Developers Conference® Austin focuses on connected games including online games, virtual worlds, and social networking. It will also feature two-day summits including Game Audio, Game Writers, Independent Games, and the newly introduced iPhone Summit.

REGISTER WITH
CODE **A09WM09**
AND RECEIVE
10% OFF
MAIN CONFERENCE
AND ALL ACCESS
PASSES*

Learn more at
WWW.GDCAUSTIN.COM

*Discounts cannot be combined with other GDC discounts or other promotions including group registrations and alumni discounts. GDC reserves the right to review and end this promotion before the end of online registration.

resources

Anderson, C. A. & Jennings, D. L. (1980). "When experiences of failure promote expectations of success: The impact of attribution failure to ineffective strategies," *Journal of Personality*, 48, 393–407.

Ames, C. & Archer, J. (1981). "Competitive versus individualistic goal structures: The salience of past performance information for causal attributions and affect," *Journal of Educational Psychology*, 73, 411–418.

Butler, R. (1987). "Task-involving and ego-involving properties of evaluation: Effects of different feedback conditions on motivational perceptions, interest, and performance," *Journal of Educational Psychology*, 79, 474–482.

Clifford, M. M. (1986a). "The comparative effects of strategy and effort attributions," *British Journal of Educational Psychology*, 56, 75–83.

Clifford, M. M. (1986b). "The effects of ability, strategy, and effort attributions for educational, business, and athletic failure," *British Journal of Educational Psychology*, 56, 169–179.

Elliott, E. S., & Dweck, C. S. (1988). "Goals: An approach to motivation and achievement," *Journal of Personality and Social Psychology*, 54, 5–12.

Kamins, M. L., & Dweck, C. S. (1999). "Person versus process praise and criticism: Implications for contingent self-worth and coping," *Developmental Psychology*, 35, 835–847.

Seijts, G. H., & Latham, G. P. (2006). "Learning goals or performance goals: Is it the journey or the destination?" *Ivey Business Journal*, 70, 1–6.

informed about their progress in mastering basic skills.

Of course, people have their own motivations and mindsets that they bring to games. Some people have a learning mindset and are likely to focus on getting better at a game. Others prefer goal-based achievements and do in fact feel motivated by them. In both cases, players are likely to have some preexisting beliefs about their game-playing abilities. However, the type of goals presented and the feedback they receive during both success and failure can have a significant effect on how they respond to those setbacks.

Through better feedback and goal-setting, we can encourage a mindset of competence, reduce frustration, and encourage players to play longer, try harder, and feel more confident about future gameplay challenges. 🎮

BRUCE PHILLIPS has been a member of the User Research team at Microsoft Game Studios since 2001. He received his BA in Psychology from Carleton University and his Ph.D. from the University of Victoria. Email him at bphillips@gdmag.com.

I don't think we should remove performance goals from games. A lot of players enjoy these types of challenges, and most games are structured around activities such as levels, rounds, races, and so on. However, we should consider incorporating other types of goals into games, too, specifically those focused on learning.

Learning goals. In many ways, learning goals are the opposite of performance goals. While performance goals focus on ability, learning goals focus on effort. It's not so much about doing as it is about trying. Improvement and progress toward the goal is as important as success.

It's important to explain that learning goals are not simply smaller or more frequent performance goals. Rather, they involve a philosophical shift in thinking about how we reward player progress.

To illustrate the differences between these goals, consider the performance goal of finishing a level in a FPS. At a low level, a player typically has to cross some boundary that triggers the level completion event, or maybe has to reduce a boss' hit points to zero. These events either happen or they don't. Additionally, the player is usually rewarded—the story advances, the player gets a new weapon, and so on.

However, despite whether they complete the level, most players will improve their abilities over the course of playing. Some will finish faster and experience fewer frustrations, some will take longer, and some will eventually give up, but most will show signs of improvement.

This is good stuff to call out. It stands to reason that adding learning goals—which focus on the skills and abilities that, when improved, make it possible for players to achieve performance goals—would enhance players' appreciation of their own abilities. All this takes is a little more time focusing on the journey, versus the destination.

Learning goals make people try harder, take more risks, spend more time on a task, become less discouraged when facing setbacks, and, in the end, succeed more frequently (also see the sidebar). Doesn't that sound like the kind of player we should be cultivating?

MEASURE FOR MEASURE

» One likely reason we don't often incorporate learning goals is that implementing them into a game is more difficult and requires more thought than traditional performance goals. It requires breaking from molds and doing something new. It's much easier to pop up a "level completed" message, a story cinematic, or an "achievement unlocked" notification after the player hits a predefined milestone in the game than it is to integrate learning goals that reflect the improvements players make. Only recently have games been tracking player data in a way that could support learning goals, which could also be a contributing factor. But most likely, we have simply been stuck following conventional wisdom about how we reward players and provide feedback.



Feedback can help teach players more effective strategies.

complex activities. Of course, these also need to be skills or strategies that your game can track. For example, if the player needs to understand how to play with stealth, it might be impossible to track <understands the stealth system>, but you could track <was hit by enemy> or <used crouch>.

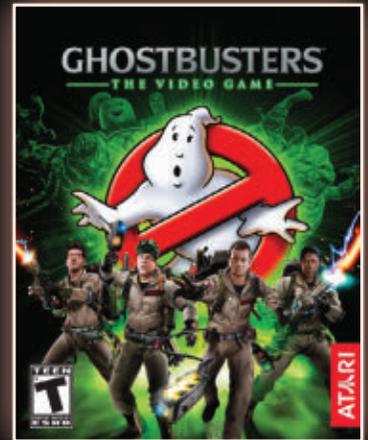
Then display progress on these component skills to the player. Rather than listing how many times x or y event happened, communicate metrics that relate to improvement, much like the example cited previously from TEAM FORTRESS. The obvious places to display progress information to players are 1) at the end of a level, 2) when they pause or quit the game, and 3) when they die.

Better yet, display a progress chart that players can access whenever they want. One example from GEARS OF WAR 2 are the messages that appear as a player nears a new achievement.

Another example from GEARS OF WAR 2 is the "war journal" which keeps track of the player's current campaign status. There's no reason we couldn't put similar messages in other games to keep players

Streams Into Threads: The Making of *Ghostbusters*: The Video Game*

By Lee Purcell



The jokes and warnings about crossing streams are inevitable, the sense of expectation building steadily, the growing interest throughout the gaming world predictable as the June 2009 release date nears. The *Ghostbusters** phenomenon has spanned more than two decades since the original movie release in 1984, but the iconic symbol of the ghost-fighting team, the “No Ghosts Logo,” is one of the most recognized

“The Intel® Graphics Performance Analyzers have been priceless in helping us improve our PC graphics performance. [Intel®] VTune [Performance Analyzer] was also used to help clear away the multi-threading bottlenecks we were having on the Vista platform, but not on the XP platform.”* —MARK RANDEL, CEO, TERMINAL REALITY

emblems in the world. With the mantle of responsibility high, the decision makers at Atari turned to the talents of an experienced, savvy, game development company—Terminal Reality—headed by Mark Randel to bring the fondly remembered spirit-fighting troop to the screens of gamers around the world.

In the process of developing the game, the technology pros at Terminal Reality recognized the potential of releasing their proprietary game engine, dubbed the Infernal Engine*, as a product. Atari’s distribution plan for *Ghostbusters*: The Video Game* includes a bevy of platforms, a challenge that matched well with the cross-platform capabilities of the Infernal Engine. The game also extends the storyline presented in the original *Ghostbusters* movie and then continued in the sequel, *Ghostbusters II*, essentially representing the third installment in the series and incorporating script direction from two of the original cast members from the film, Dan Akyroyd and Harold Ramis.

The Lure of Games and High Technology

Mark Randel’s interest in computer video games surfaced early and eventually led to an educational path that combined electrical engineering and computer engineering. “I had always wanted to write video games ever since I got my first Atari* 2600 VCS when

I was in middle school," Mark recalled. "I saw a chance to go to work with Bruce Artwick, one of the computer game pioneers while I was at the University of Illinois, so I took the time to learn all I could about making computer games."



Despite the rigors of university coursework in two demanding fields, during the span of his formal education Mark also concurrently developed the highly respected Microsoft Flight Simulator* engine, a remarkable achievement by itself.

"Although my graduate degree is in electrical engineering," Mark said, "I did my thesis work in the Quantum Electronics and Ultrahigh Speed Digital Computer Research Laboratory at University of Illinois, which was then doing pioneering work with gigahertz frequency hardware. I gained a lot of experience—and a pretty good peek at what's coming and how computer chips, memory, and busses are going to interconnect with each other in the future."

His first commercial game release in 1995, *Terminal Velocity**, became a runaway hit, selling more than a million units. Developing cutting-edge game engines remained a strong interest and a focal point of the development path at the company Mark co-founded, Terminal Reality. Mark's educational and commercial accomplishments have been effective preparation for understanding and exploiting the performance capabilities of new platforms. These insights are



reflected in Terminal Reality's cross-platform game engine work—ranging from the parallelization of game physics routines for the Sony PlayStation* 3 console to multi-threading work to accommodate the trend-setting, performance achievements of computers powered by Intel® Core™ i7 processors.

PERFORMANCE TUNING WITH INTEL GPA

The release of Intel® Graphics Performance Analyzers (Intel® GPA) in March 2009 opens up opportunities for game developers to precisely evaluate and optimize performance for notebook computers and mainstream desktop equipment featuring Intel® Graphics chipsets. Residing unobtrusively on a network-based architecture, the two key tools of this solution—System Analyzer and Frame Analyzer—support DirectX* 9 implementations, with DirectX 10 support available before the end of the year.

The System Analyzer presents a broad overview of system performance, while the Frame Analyzer allows developers to

inspect API-level transactions, offering details down to the draw-call level for each individual frame. Aaron Davies, senior marketing manager in the Intel Visual Computing Software Development group, described it in these terms: "The tool provides a performance delta between your original frame and your experimental frame, without recompiling code."

Intel GPA is available for free to members of the Intel® Visual Adrenaline Developer Program. For more details about the capabilities of this solution, visit: www.intel.com/software/gpa.

“I really think in-depth knowledge of how computer chips work—from microprocessors down to the memory and I/O—has given us a distinct advantage with the coding of the Infernal Engine,” Mark said.

The Power of Slime

Give a group of innovative game developers enough processing power and a few ideas from a chart-topping movie from twenty years back, and you may be surprised by what develops. In the case of *Ghostbusters: The Video Game*, the programming team at Terminal Reality started playing around with slime, the ectoplasmic tie that binds, and ended up with an addition to the gameplay sure to amuse and entertain.

Playable demos of the game at Comic-Con 2008 and the New York Comic-Con 2009 offered a taste of the gameplay built into the upcoming title, which takes place three years after the last *Ghostbusters* episode, an interval that has allowed technology updates to the equipment wielded by the ghostbusting team, including the familiar paragoggles, Pk meters, and proton packs. The Dark Matter generator includes a freeze beam and a shock blast. The Slime Blower includes both the Slime Stream and the Slime Tethers.

The highly destructible environment (made possible by Terminal Reality’s VELOCITY* physics) also depicts creatures that are spontaneously assembled out of fragments of nearby objects, presenting a fresh challenge to gamers. In demo presentations, Mark has commented on this feature, saying, “Anything can become a monster.”

While on the demo circuit, Mark also highlighted a game addition called Slime Tethers, which allows players to fire off long, sticky strands of slime and hook objects in a scene—whether a bookcase, annoying pedestrian, parking meter, or street sign.

Heavy use of artificial intelligence is necessary to handle the crowd scenes, which can include as many as a 1,000 people interacting in the field of view and another 500 or so whose activities are tracked offscreen. The realism of many of the catastrophic street scenes owes a debt of gratitude to the skillful artificial intelligence directing milling crowds of individuals.

THE PINNACLE OF HIGH-END GAMING: INTEL® CORE™ i7 PROCESSOR MAGIC

The growing complexity and increasing photo-realism of modern video games call for advanced platform technologies that scale to the processing demands. The Terminal Reality development team was so impressed by the capabilities of the Intel Core i7 processor, they created a number of pre-release demos of the game to showcase special effects and new features. Currently creating a buzz across the gaming world, the Intel® Core™ i7 processor Extreme Edition, the highest performing desktop processor on the planet¹, features intelligent multi-core technology that accelerates performance in response to increasing workloads.

New features enhance the overall gaming experience, such as Intel® Turbo Boost Technology (to maximize speed for demanding applications), Intel® Hyper-Threading Technology (for advanced multi-tasking and support for up to eight threads), and Intel® Smart Cache (to provide a higher performance, more efficient cache subsystem). To experience *Ghostbusters: The Video Game* in its best light, take advantage of the processor that has become prized and sought after in the gaming world, the Intel Core i7 processor Extreme Edition.

¹Performance based on select industry benchmarks, game titles, and multimedia creation applications. Actual performance may vary. See www.intel.com/performance/desktop/extreme/ for additional information.

Imaginative, innovative special effects, advanced physics, extensive use of artificial intelligence to control the actions of people within the game: all of these characteristics promise a lively and entertaining game environment that will introduce new twists to the classic film storyline.

Adding More Threads to the Mix

Terminal Reality has worked closely with Intel over the past year to get *Ghostbusters: The Video Game* running optimally on a wide range of PC platforms. Recently the development team had the opportunity to test game behavior on the Intel Core i7 processor and put together a series of demos showing just what can be accomplished when a higher plateau of performance is available.

“We’ve put together some demos,” Mark said, “that show what you can do on an Intel Core i7 processor that you can’t do on any other system right now. We have specific demos with over 2,500 objects,

THE INFERNAL ENGINE GOES PUBLIC

One of the side benefits of the work on *Ghostbusters*[®]: The Video Game* has been the realization by the development team at Terminal Reality that they have a winner on their hands with the in-house-developed Infernal Engine*, which takes advantage of multi-threading on a variety of very different platforms—from the Sony PlayStation* 3 with its special-purpose units (SPUs) to the gamer's nirvana—a PC powered by the Intel® Core™ i7 processor Extreme Edition. The Infernal Engine, created to exploit parallelism and extend cross-platform compatibility to the widest extent, can be licensed from Terminal Reality in the second quarter of 2009.

The Infernal Engine combines rendering capabilities suitable to constructing photo-realistic environments; a flexible, leading-edge physics solution; and a powerful and adaptable particle system. The emphasis is on cross-platform interoperability and streamlining the production pipeline to boost productivity.

Why create a new game engine when there are so many strong competitors already in the market? "First," Mark responded, "I want to say that there are some very good engines out there for very specific purposes. The Infernal Engine is the first engine that brings rendering, physics, artificial intelligence, sound, scripting, and particles together that runs on most platforms, including PS3, 360, PC, Wii, and PSP. Infernal was designed from the ground up to run major systems in parallel with the PS3 in mind. Once we had the PS3 working well with our multi-threading model, the 360 and PC came together naturally."

"We also retained our original single-threaded solution to be compatible with the Wii and PSP," Mark continued. Specifically, the Infernal Engine does three main tasks very well in parallel, which frees up the main game thread for C++ game coders. Physics and animation run completely in the background, even on the PS3 system, where they run on SPUs. The rendering for one full frame is also queued up, so the GPU is never starved, and no thread is ever waiting on the GPU FIFO."

Joe Kreiner, vice president of sales and marketing with Terminal Reality, commented in an article for IGN.com, "Terminal Reality's Infernal Engine is a breakthrough in efficiency for game development middleware. Our licensees can leverage their work across more platforms, in less time, than any other engine—giving them a competitive edge critical for success. Our licensees get stunning visuals, fast time to market, and the support of Terminal Reality—one of the most experienced independent game developers in the industry."

simultaneously colliding and making great use of eight threads. Basically, I can't wait until everybody has at least eight hardware threads on their desktop or in front of their TV because the immersion—what's possible and what's going to happen in a game—is going to dramatically increase. The number of objects, number of characters, everything on the screen . . . It's like a glimpse into the future."

The collaborative engineering work with Intel included use of both long-standing tools, such as the Intel® VTune Performance Analyzer, and more recent additions to the software development products, such

"With Intel's help, we both took on the challenge of getting Ghostbusters to run well on Intel® Graphics. This enables Ghostbusters to run on the widest selection of PC platforms on the market. The graphics only mildly scale back for this platform, but then can scale up way past what the consoles can do for those with SLI-enabled graphics cards."—MARK RANDEL, CEO, TERMINAL REALITY*

as Intel® Graphics Performance Analyzers (Intel® GPA). Mark noted that during development, the programming team was encountering multi-threading issues under Microsoft Windows Vista*. The game was scaling poorly under Vista, even though it was running well under Windows* XP. Using the Intel

VTune Performance Analyzer, the developers quickly identified and eliminated the performance bottlenecks. The problem, it turned out, was because of an obscure threading bug in the *Ghostbusters* code that affected Vista but not XP. The Intel VTune Performance Analyzer made it easier to pinpoint the bug and fix it.

As is the case with many game developers these days, Terminal Reality is designing the game to run on the widest range of platforms, including notebooks equipped with Intel® Graphics chipsets. The challenge, of course, is to expand the customer base to the universe of notebook users without dramatically compromising the quality of the game graphics when running on mainstream hardware, rather than tricked-

out, high-end gaming machines. “Right now we’re using the Intel GPA to dramatically improve the performance on integrated graphics chipsets. It’s still a work in progress but we’re coming along, and we’re able to get a really good idea now of what shaders are taking a lot of time and where the bottlenecks are in the game.”

“The goal, of course,” Mark said, “is to have the game looking as sharp graphically as possible. I want to turn off the fewest number of details and preserve all the graphics features of *Ghostbusters* on the integrated graphics hardware. We’re really close to being able to do that now.”

During the tuning process, Terminal Reality provided Intel application engineers with the executables of the game. The engineers then used Intel GPA to gain a dynamic view of the execution of the game code—down to the level of pixels, shaders, and routines. Areas in which the code is taking an unreasonable amount of execution time are identified graphically.

“Not only are we able to see where the bottlenecks are,” Mark said, “but we can even see on a primitive basis—a per rendering primitive—with spikes indicating what’s taking the most time.”

Targeting Slowdowns—Intel® GPA Goodness

Investigating the reasons behind performance slowdowns is an area where Intel GPA excels, as the Terminal Reality development group and a team of Intel application engineers discovered. A recent engagement initiated to tackle performance issues in running *Ghostbusters: The Video Game* on computers equipped with Intel Graphics found problems with a particular scene in a library containing 200,000 books—each an individual object. Smooth gameplay

became impossible during this scene as the frame rate dropped to the point that video playback stuttered.

Disabling the Z-test allowed the development team to identify objects that should have been occluded (for performance reasons). Frame analysis of the library scene—made possible with Intel GPA—showed 12,565 *Draw()* calls (other scenes typically have about 3,000 *Draw()* calls). Digging deeper to find a technique to suppress rendering of occluded books, single-frame

analysis using Intel GPA confirmed the frame-rate hit in rendering the books; Intel and Terminal Reality began experimenting with a software switch to dynamically control rendering of the books. Further analysis with Intel GPA showed that the frame rate increased by 3.3 times with book rendering turned off.

To minimize the unnecessary rendering of books (without

designing a full-scale occlusion culling system), Terminal Reality created a pixel height test. Objects that contribute less than a full pixel to the frame (based on test code run by the processor) are not sent to the graphics subsystem for rendering. Intel GPA offered a clear picture of frame-rate increases—showing a 2X improvement in rates. The final code implementation— informed by the frame data acquired by Intel GPA— demonstrated scene-rendering improvements between 2–2.5X fps in the level containing the library scene.

Better Graphics, Enhanced Physics

The complexities and expense of producing Triple-A game titles continues to grow, forcing game developers to choose platforms and tools wisely, adapt on the fly to changes in market sectors, and leverage innovations that help streamline production and exploit performance opportunities. Mark said, “*Ghostbusters*

“The VELOCITY physics engine runs almost completely in parallel. Every collision detection can be done on five threads. Every Jacobian calculation can be done on five threads, and the solver can be run on five threads, even with huge physics islands. Running a large island in parallel was a very difficult problem to solve and being able to subdivide it evenly across multiple processors makes it automatically load balanced.”*

—MARK RANDEL, CEO, TERMINAL REALITY

deep into development of the next-generation version of the Infernal Engine. Unfortunately, we cannot yet talk specifically about the upcoming advances . . .”

In the meantime, there is plenty of action and entertainment to enjoy in *Ghostbusters: The Video Game*, an experience that is a bit like taking a step inside a film

and becoming a character along with the other actors. With snappy dialogue, smart gameplay guided by the latest artificial intelligence, groundbreaking physics, and special effects, *Ghostbusters: The Video Game* is bound to win over a whole new generation of fans and expand the expectations of the gaming community.

Intel and Terminal Reality are also cooperatively evaluating the possibilities of future game enhancements that will be made possible by Intel's next-generation visual computing architecture, code-named Larrabee. “Once we have access to this new graphics platform,” Mark said, “we are hoping to take our code base over to it and get some really cool results.” ■

will have been in development over three years by the time it is released. Gamers expect more and more out of their title. When we started *Ghostbusters*, multiplayer wasn't very important, but we realized later on that we needed a strong multiplayer component in the game to be successful. The graphics bar for 2009 is a very high one, and we feel we have hit the bar for graphics and raised it considerably for physics-based gameplay.”

Despite some good-natured prodding, Mark declined to provide specific details of upcoming projects at Terminal Reality, speaking only in generalities. “We have lots of cool stuff coming up in the BlackOps room of Terminal Reality,” he said, “as we are already

For More Information

To learn about licensing the Infernal Engine*, visit: www.infernalengine.com

For more details about the breakthrough performance of the Intel® Core™ i7 processor Extreme Edition, visit: <http://www.intel.com/products/processor/corei7ee/>

To see the Gamespot* perspective on the *Ghostbusters* game, go to <http://www.gamespot.com/pc/action/ghostbusters08/index.html?tag=result;title:8>

For more information about *Ghostbusters*: The Video Game*, visit <http://www.ghostbustersgame.com/>

For benchmark data on the Intel Core i7 processor Extreme Edition, go to: <http://www.intel.com/performance/desktop/extreme>

For an entertaining retrospective on the evolution of computers, including notable games, visit the Computer Time Line, part of the Atari Archives, at: www.atariarchives.org/deli/Time_Line.php

To get more great articles like this one, subscribe today to Intel® Software Dispatch for Visual Computing at: www.intelsoftwaregraphics.com



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

*Other names and brands may be claimed as the property of others. | Copyright © 2009. Intel Corporation. All rights reserved. 05/09/SM/CS

TOMB RAIDER UNDER WORLD

ERIC LINDSTROM

THE CONCEPT PHASE OF TOMB RAIDER: UNDERWORLD BEGAN WHILE ITS PREDECESSOR, TOMB RAIDER: LEGEND, WAS IN FINAL QA AND nearing submission. Previews for TOMB RAIDER: LEGEND were very encouraging, and we felt that there was still plenty of unrealized potential to tap in the existing feature set. Enough so, the reasoning went, that we could focus on content and leveraging existing functionality to develop a bigger and better Lara Croft adventure in less time. In many ways this is what the team accomplished, but as is always the case in game development, reality was more complex than we anticipated.

It is particularly interesting to note that much of what went wrong in development involved pitfalls that we anticipated but still fell into despite our efforts to avoid them. It's important to note that most postmortems talk about "What Went Wrong" and not just "What We Did Wrong" because sometimes you make mistakes, but other times you suffer from acts of god and do your best to cope. A *Game Developer* article last year ("What Went Wrong?" December 2008) specifically questioned why game development seems to make the same mistakes over and over. In light of that, some of the "wrongs" will be discussed in terms of how our methods to avoid known issues fell short.

what went right

1) LONG ALPHA. We scheduled an unusually long Alpha to deal with unresolved pre-production issues and to set ourselves up for a shorter Beta to make room for more polish time at the end. This paid off handsomely with respect to art production, and it's one of the reasons the game looks so good. We also recovered from a fair amount of design deficit that had carried over from pre-production, and on the code side we managed to get most of our core functionality up to scratch.

Another thing we did right during this long Alpha was to have multiple scope reductions. This was our first "next-gen" title, and we repeatedly underestimated issues of complexity. We would assess and determine that the game was too big, and then cut enough content to bring it under with margin to spare. Then two months later we would see that we were again coming in too big, necessitating further scope reductions. The game was designed to be able to handle this degree of reduction, as seen by the fact that almost all the features and areas originally planned for the game made it into the final version, only smaller, and connected to each other in fewer ways. We managed to reduce the scope by trimming branches everywhere without having to uproot any of the trees entirely.



postmortem

2) FOCUS. When making a sequel, producing a game just like the last one with slightly different content and a few more features is an easy mistake to make. Despite the fact that we had some significant new goals, like making the traversal less linear and bringing back more free exploration, we almost fell into this trap. But many on the team saw that we needed an additional focal point to both rally the team around and to use as a unique selling proposition for the game. The result of this was the concept of epic exploration puzzles.

Making large-scale in-game devices and areas with multiple layers of connected puzzles gave the game an exceptional expression even compared to previous *TOMB RAIDER* games, and it also gave us a litmus test for spending production effort across the game. This led to the creation of a sub team devoted to these puzzles, which proved to be complicated constructs. While we would have been better off had we foreseen this need and planned for it properly from the start, the fact that we saw the growing concern, created this special sub team, devoted more staff and resources to it, and assigned a dedicated producer was an example of how well we solved unforeseen problems in development.

3) PRODUCTION FLEXIBILITY. Even though it was hard to significantly redirect the juggernaut that was *TOMB RAIDER: UNDERWORLD*, we made a lot of successful changes when confronted with breakdowns of production processes. We started out thinking that we knew the best way to design the ruined jungle gyms that form the bulk of a *TOMB RAIDER* playground, based on lessons we had learned from the previous game, but reality confounded us again, and we made changes accordingly.

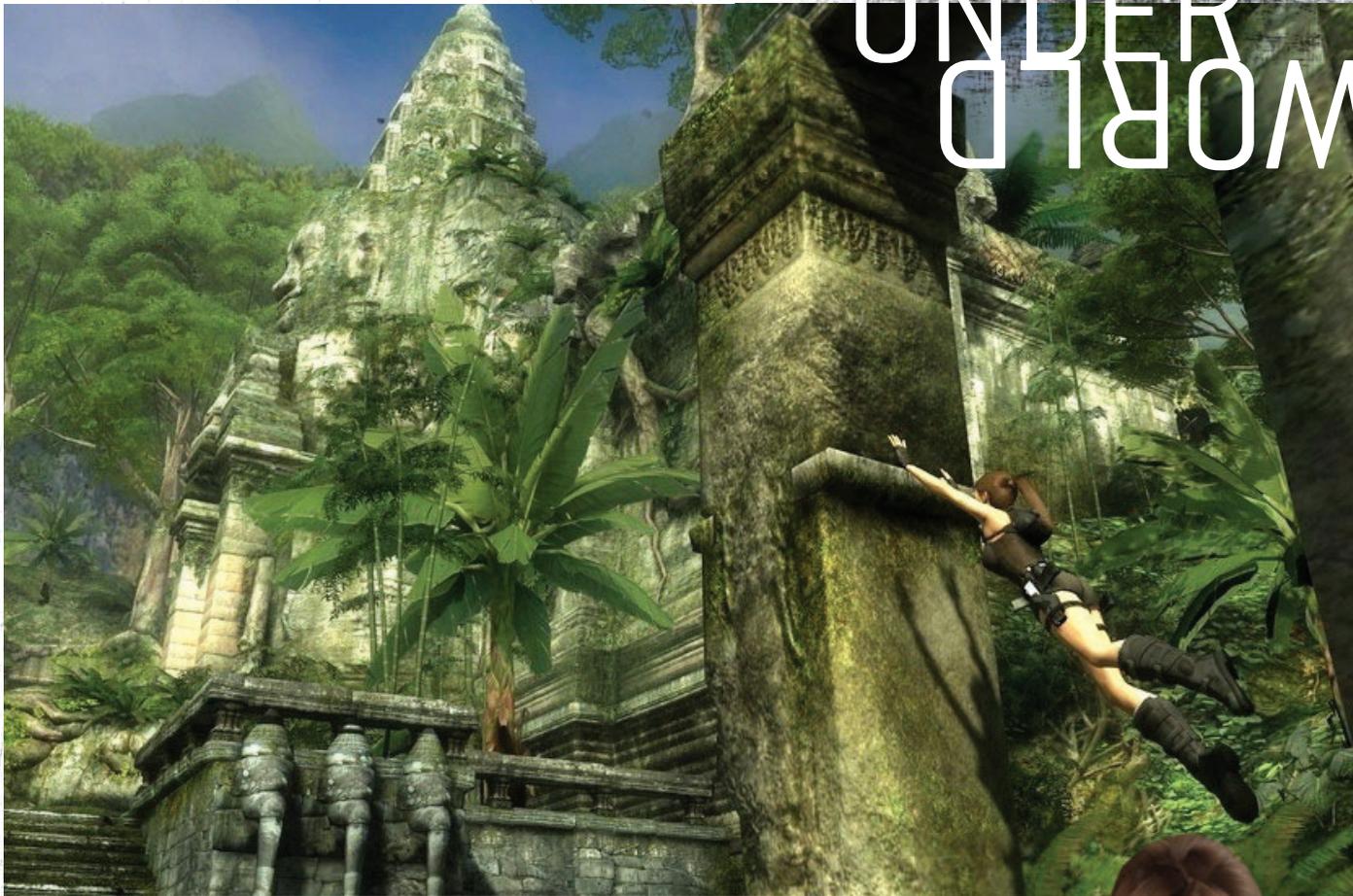
One clear example is related to level design and art. From the beginning we all agreed that it was critical to have both disciplines work in tandem from day one to make environments that were fun, beautiful, and credible. Prior outings had either started with design-generated block mesh, leading to geometry that was extremely difficult to turn into plausible ruins, or with beautifully architected tombs that did not provide fun climbing opportunities or properly authored gameplay. Our solution was to pair level designers and environment artists together by location in the world, such as Mexico or Thailand, and have them work together, iterating on environment architecture to make it both fun and aesthetically appropriate.

While this proved to be the right direction, many of our earlier processes didn't work out. But our willingness to accept the reality that these initial attempts were flawed allowed us take the big step of changing production workflows in progress, often multiple times. It sometimes felt chaotic and frustrating to change process so frequently, but had we stuck with broken paradigms the situation would have been far worse. In the end, our production methods weren't perfect, but they were superior to what we thought would see us through from the outset.

4) METRICS. In a game like *TOMB RAIDER: UNDERWORLD*, where the player can interact with so many different elements of the environment in so many



RAIDER UNDER WORLD



ways, metrics are hugely important. They form the basis of getting Lara Croft off the grid and eliminating the tractor controls that kept her from evolving into the fluidly moving character we have today. *TOMB RAIDER: LEGEND* suffered from changes in jump distances, ledge parameters, and other metrics until late in development, resulting in countless hours of rework for both designers and artists (it was the most painful for the latter), and we were determined to avoid this in the sequel.

Our lead level and systems designers collaborated to establish metrics for every aspect of player interaction until a full set was defined early in development. Some changes were made later, and holes were discovered and needed to be filled, but on the whole the degree to which we maintained and enforced metrics without major changes was a huge improvement over past efforts. If not for this success, the amount of game real estate we included in the game would have been significantly reduced.

5) SANITY. Most people working on the game had never been on a project this large, in terms of team size, degree of coordinating efforts, and the amount of game assets we had to fit into the timeline. Add to that how passionate and invested such talented people are in the quality of the final game, and the mixture could have been explosive. Yes, there were differences of opinion, often strong ones, and certainly there was plenty of stress and the occasional meltdown, but the professionalism rate was extremely high, and this had a large positive impact on the production in ways that went far beyond people getting along.

When scope reductions had to be made, when work had to be redone or discarded, or process changes were needed, maturity was high and ego was low. We had no tantrums over trying to cut someone's favorite elements of the game. Everyone remained rational and evenhanded throughout the length of the project, right into the manic bug-fixing days before submission. Lesser projects with weaker stresses have broken people and teams, and that is why this deserves to be on this list of what went right—because on a project like this, navigating the many and lengthy trials and tribulations intact contributed as much to its ultimate success as any other element.

what went wrong

1) SHARED TECHNOLOGY. At the start of *TOMB RAIDER: UNDERWORLD*, Crystal Dynamics as a studio decided to pursue the Holy Grail of internal development: a robust and powerful shared code base to use as a jumping off point for all future games. This proprietary engine would be augmented and maintained by dedicated engineers who could provide the common functionality our games would need, while team programmers could focus on features specific to their games. The studio believed that because *TOMB RAIDER: UNDERWORLD* and the shared code base would both be based on *TOMB RAIDER: LEGEND* code, the efforts could be combined even given our tight production timeline.

A lot of talented and hardworking programmers were put on the shared code team, including many of those who engineered *TOMB RAIDER: LEGEND*, so skill was not an issue. The biggest problem here turned out not to be over-ambition or complicated dependencies (though these were certainly issues). The problems were much more related to ownership and priorities. Within a team, when schedules begin to slip and need to be put back





on track, the entire team can get together, redefine its mission, and make whatever collective changes are needed to bring production back into alignment with the calendar. But the shared technology group was not on the team. While its charter was to serve the teams, it had to serve multiple teams with conflicting needs. From the point of view of each team, the shared technology group was a cadre of programmers that didn't report to our lead engineer. It was an enormous dependency that we could only influence via petition and persuasion, and frequently could not even schedule around as we had limited visibility into their progress.

We knew from the beginning that basing *TOMB RAIDER: UNDERWORLD* on a nascent and evolving code base was an enormous risk, a big potential pitfall. So why did we walk into this trap with our eyes wide open? At the time, it seemed that the potential payoff was worth the risk, and that we had the right people working on the problem, so we marched along with the shared technology group and did the best we could knowing that some of the challenges we were facing could ultimately yield more efficiencies down the road. Ultimately, however, some of our fears were realized as we had indeed overestimated our ability to overcome all the known risks.

2) REFACTORING. One of the earliest engineering tasks involved refactoring the player code to make it more robust and more able to accept new functionality. This effort was important to be able to efficiently expand our feature set, but it wasn't understood at the time that this refactoring meant taking apart the engine and putting it back together in such a way that it became unplayable for the duration.

The design team had been anticipating a comfortable period of working on layouts, experimenting with interactions, creating more evolved setups, and otherwise engaging in exploratory and iterative design that wasn't possible on *TOMB RAIDER: LEGEND*. But our pre-production hopes disappeared as much of the player functionality went offline. The usual delays and schedule slips prolonged this dark period, during which the engine parts were scattered across the lawn. The problem was compounded when we had to move into production and deeper into Alpha without many core player functions working. This had a big chilling effect on early design and layout efforts, and the effects of this lasted all the way to the final product.

RAIDER UNDER

Fundamentally this was an issue of miscommunication, which is a traditional gremlin lurking on every project. We fought this gremlin from the start, and given the size of our production did a fair job of it, but this particular mishap was unique in how irreversible it was. Improving communication is about better information trafficking up front, but just as importantly, quickly addressing miscommunication when it arises, so we don't consider this one of those traditional mistakes that we repeated. You can't catch everything, and the effects of refactoring slipped through the net, and once started, there was no going back. It was a particularly bad piece of miscommunication to have, and we just had to ride it out. In the end, a lot of valuable hands-on design time was lost at the start, and we would have had a significantly better layout in the shipped game had we caught the issue earlier.

3) PREPRODUCTION. Our preproduction got rocked by the previous two issues—the shared technology effort and how refactoring rendered previously playable mechanics unplayable—but we didn't have the option at the time of moving our ship date. This meant getting much less out of preproduction than we hoped for in two major areas.

First, it put us in a position similar to our time with *TOMB RAIDER: LEGEND*, with designers creating layouts based on paper metrics, where the mechanics would not be playable until later. Though this is a clear violation of good design practice, we felt that the damage could be contained because the design team had just finished making a *TOMB RAIDER* game with many of these same mechanics, so we could get by for longer without them. But the situation was far from optimal.

Second, aside from porting *TOMB RAIDER: LEGEND* to the Xbox 360, this was our first project made for what were then next-generation consoles. It turned out to be far more complex and content-intensive than we anticipated, and because our capacity to create layouts was so compromised in preproduction, we couldn't adequately test, measure, or scope the art side of the production equation. Because we needed to enter production by a certain date to have a reasonable chance at shipping on time, we ended preproduction before we had an adequate understanding of our art production needs and processes, and just as critically, before we had developed the pipeline and support structure we would need later for outsourcing art.

We were also understaffed in certain areas due to slots on our team being held for team members who were still finishing up *TOMB RAIDER: ANNIVERSARY*. In the end it was as though we planned to have a concept phase, a prototyping phase, and a preproduction phase, but in practice only did some concept, some prototyping, and then jumped straight into production before finishing preproduction. Much of this pressure came from the collision between ambition and capacity, made worse by the cognitive dissonance between seeing a game on paper that you want to have, and knowing that the production data indicates you can't have it by the prescribed date, but being unwilling to change either one.

At the time, and indeed throughout the project, we knew it was risky to move on to the next phase of production before the previous one had completely finished, yet we did so anyway. This is one of those repeated mistakes, so why did we do it? Cognitive dissonance is part of the answer. Wishful thinking is another part. But mostly, in an imperfect world, every risk you take is a gamble that may fail, and at the end of the endeavor, you end up needing to explain only the risks you took that didn't pan out, like this one.

4) ACTS OF GOD. This is the category that some postmortems title "Too Many Demos" or some other problem that comes at a team from the outside. We certainly had the problem of too many demos, but those were only one entry in a class of problems that dropped on us unprepared.

Demos were particularly aggravating because we specifically set out to avoid this issue from the start. We demanded long term demo schedules, and we received them. We refused to bow to some requests for demos that weren't on the schedule, and this was honored by the publisher. But there was also a slippery slope, where some unscheduled demos were accommodated because of various circumstances. We sometimes faced a dilemma where marketing gave us a choice in which a particular demo, at the expense of two weeks of production time, seemed like the better long term choice for the success of the product. There are too many stories of great games disappearing into the noisy marketplace to ignore the importance of demos and good PR.

Yes, these demos often result in a reduction of product quality, and yes they distract from team focus, but in the end, as painful as it feels, it's sometimes best to do the demo. This is a clear example of walking into a trap with eyes wide open—making a mistake that you know about in advance—but it happens repeatedly because the answer isn't to refuse demos, or to plan for them better. The answer is to make a game that doesn't come together only at the end.

We also had an unusual number of weddings, honeymoons, production babies, and untimely departures on this project throughout the team, but most painfully among the discipline leads. We lost our art director midway through production; not to another company,

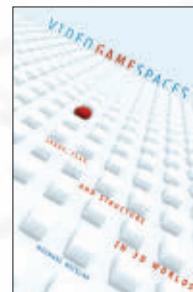
The MIT Press

Video Game Spaces

IMAGE, PLAY, AND STRUCTURE
IN 3D WORLDS

Michael Nitsche

"Video Game Spaces takes readers on a



fantastic journey that resonates with what we love best about games—their double identity as places to both ponder and play. Richly researched and well-written, the book delves deeply into the interdisciplinary

nature of Game Studies, offering key insight along the way." — Katie Salen, co-author of *Rules of Play* and editor of *The Ecology of Games*

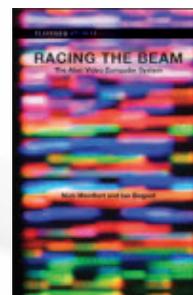
312 pp., 27 illus., \$35 cloth

Racing the Beam

THE ATARI VIDEO COMPUTER SYSTEM

Nick Montfort and Ian Bogost

"Racing the Beam presents not just the techni-



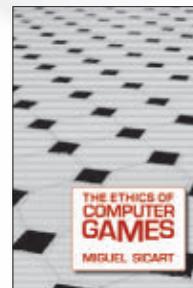
cal challenges but the financial, bureaucratic, and scheduling considerations that harried the Atari 2600 VCS programmers. Modern game designers should read this book." — Chris Crawford, former head of Atari's Games

Research Group, and co-founder of Storytron Platform Studies series • 184 pp., 22 illus., \$22.95 cloth

The Ethics of Computer Games

Miguel Sicart

Why computer games can be ethical, how



players use their ethical values in gameplay, and the implications for game design.

280 pp., 20 illus., \$35 cloth

To order call 800-405-1619 • <http://mitpress.mit.edu>

but to another industry, so there wasn't much we could have done about that. We lost our lead designer toward the end of production when she had a baby; also something we couldn't have done much about (nor would we have wanted to, as the baby is beautiful!). And most tragically, our lead level designer died suddenly during the first half of production. These key figures in our effort were extremely valuable not only because they were smart and capable, but also because they were a part of the success of *TOMB RAIDER: LEGEND*, and therefore knew intimately how to make this kind of game. People like that are rare, because *TOMB RAIDER* games are hard to make for many reasons, and in the timeline we were under, they were irreplaceable. We compensated for these losses with people on the team assuming new responsibilities to fill in the gaps, and some of these team members did amazing work and really saved us, but there's no denying that we would have had a much smoother production and a better end product without these losses.

5) SCOPE. Yes, we tried to do too much. It's human nature to reach as far as you can, and to think your arm is longer than it really is, but some games are more sensitive to scale than others. If you make a sudoku game, you can keep making grids until time runs out and you're done, but with games that have interlocking feature sets, plus a beginning, a middle, and an end, it's much harder to scale properly, and harder still to change scale during production.

There were success stories with respect to our scope, and with reductions we made all the way to Beta, but in the final analysis we tried to do too much, got locked into needing to do it, and scrambled to get it all done to the quality standard we hold ourselves to. The reason why scale kept outdistancing our deadlines is mostly because

we underestimated the complexity of next-gen development and did not spend enough time in preproduction smoking out all the issues.

The biggest casualty of making too much game wasn't the crunch time—we had less crunch than on past games—and it wasn't that we left parts of the game undone, since there are no holes in the experience. The most damage was to an element that was very important to us: polish. We padded our schedules an unprecedented amount, and then some, in order to

have sufficient time to polish. Unexpected complexities in next-generation development, along with communication throughout a team larger than the studio had ever seen, consumed our polish time and more. This makes the mistake of underestimating our polish time more understandable, even though this is also one of those repeated "what went wrong" scenarios, because we were dealing with a whole new class of unknowns. This won't be the case going forward, however, so the team should be equipped next time to avoid the usual mistake of not scheduling enough polish time.

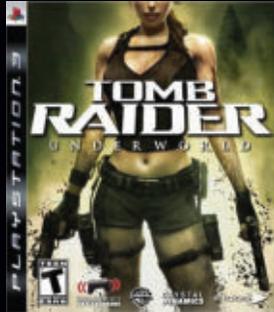
risks and rewards

» Our game was intended to be a strong sequel that drafted heavily off the successes of *TOMB RAIDER: LEGEND*, but turned into a major product in its own right; bigger and more lush than anyone could have reasonably expected at the outset, because of the hard work, talent, flexibility, commitment, and level heads of the many team members who created it. It's very gratifying that some reviews have hailed *TOMB RAIDER: UNDERWORLD* as the best *TOMB RAIDER* game since the first one, and the fan response has been phenomenal.

So why did so many things that we guarded against go wrong anyway? The categories described here are only five of many more. (Broken builds and long build times get honorable mentions, and the added distinction of raising blood pressures more than probably all other issues combined.) Doesn't the foreknowledge of pitfalls make possible the absolute avoidance of them? No, it doesn't. Creative endeavor by its very nature is chaotic, and where creative chaos, ambition, and dedication collide with production concerns and the hard walls of a publisher's deadlines and goals, you're going to fall into many of the same holes, or worse, get steered into them. The answer isn't to do everything possible to avoid all known traps, because doing so would kill innovation and creativity; the trick is to be aware of potential pitfalls, and to be prepared to quickly and nimbly mitigate the negative impact of stepping into one. 🎮

ERIC LINDSTROM is the creative director of *TOMB RAIDER: UNDERWORLD*. Previously he was the story designer of *TOMB RAIDER: LEGEND*, and he's been a game designer for nearly twenty years. Email him at elindstrom@gdmag.com.

GAME DATA



PUBLISHER
Eidos

DEVELOPER
Crystal Dynamics

NUMBER OF DEVELOPERS
84 internal staff, 15 contractors, not including shared technology staff

LENGTH OF DEVELOPMENT
2.5 years

RELEASE DATES
November 18th USA, November 21st EU

HARDWARE
Quadcore PCs with 64-bit OS, 4 GB RAM, 500 GB RAID 1

SOFTWARE
Turtle rendering package, Bableflux, Perforce, MotionBuilder, Maya 8.0, Zbrush, Photoshop, Bink, FMOD, Test Track Pro, and proprietary graphics and physics engines

PLATFORM
Xbox 360, PS3, PC (also Wii, PS2, and Nintendo DS)



BLADE GAMES WORLD

BLADE 3D

REVIEWED BY GREG SNOOK

BUILT UPON THE XNA FRAMEWORK, Blade Games World's Blade3D eases the process of game development for independent and professional developers alike. While the XNA Game Studio products have been a boon to those who want to rapidly prototype ideas and craft full Xbox Live Community Games, these products are primarily a programmer's development environment. For the average developer, generating the tools and infrastructure needed to develop a complete game or prototype can still be a daunting task requiring significant programming ability. Blade3D seeks to alleviate this burden by housing a rich set of development, editing, and debugging tools in a well designed WYSIWYG scene editor.

THE BLADE3D EDITOR

» Blade Games World's goal for Blade3D is to allow anyone to create a game with a reasonable investment of time and energy. As such, the interface of the 3D development environment is designed to contain all the features and amenities of the Blade3D engine without becoming an intimidating mess of dialogs and deeply-nested menus. The layout is clean, easily readable, and surprisingly intuitive to navigate. While there are an ample number of tutorial videos available to walk users through the most common tools, the design lends itself to exploration. Common elements are easy to locate, making the tool easy to use without needing to keep a crib sheet of keyboard commands on hand.

The layout is limited to a single viewport window, which can be distressing to content creators accustomed to using

multiple viewing angles at once. In practice, I didn't find this to be too much of an issue, but a second window would be useful when debugging AI behavior over a large environment or when trying to align objects along multiple axes. The single viewport does allow for the live frame rate monitor to give a fairly accurate depiction of your scene's performance, and swapping between active cameras is simple enough. Thankfully, all menus are user-positionable, and can be docked or left free-floating. This can help multi-monitor users capitalize on the single viewport by pushing all extraneous menus to another display.

Prominent in the layout is the unified object browser and property editor. All raw content items and game entities are shown here in a tree view hierarchy. This combines the properties of a scene graph with a complete content browser. Simple drag and drop controls allow users to place items from this browser into the scene or the visual scripting system. Materials can be dragged directly onto models, models can be dragged into visual scripts, and so on.

Also notable is that all entities in the scene browser can be edited at runtime using a uniform property editor. Any game entity or content asset can expose a set of tunable parameters to the property editor for live editing. This simple concept is in some ways the greatest part of the Blade3D package. While your game is running, you can inspect and tweak the properties of all game and content items. For iteration and tuning, this is a huge time saver.

To round out the tool, Blade Games World has also added

support for multiple levels of undo and redo, the building and deployment of binaries, integrating content from the Blade Games World community marketplace, taking screenshots, and navigating tutorials. There is even a menu button to submit bugs directly to the Blade3D development team. Clearly, Blade Games World has gone through a lot of effort to make this a comprehensive tool for game developers.

ENGINE FEATURES

» The editing environment is clean and intuitive, but developers

need functionality above all else. Luckily, Blade3D covers all the basics admirably. Skeletal animation, kinematics, physics, particle systems, terrain and audio subsystems are all provided as well as more esoteric items like lens effects and character inventory management. On the graphics front, the engine provides a rendering system which supports HLSL shader authoring (with integrated editor) and a data-driven material system.

All subsystems are functional, but some are still under development. For example, while

size matters

www.rtpatch.com

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



the engine supports twenty different model formats, all skeletal animations for a given model must be laid out in a single FBX file and imported in bulk. Upon import, the user can specify the frame ranges of individual animations to carve them back up into individual assets. While the remainder of the runtime animation system is well-crafted, this odd quirk in the content pipeline can take some getting used to.

Similarly, the embedded physics system has been hand-crafted by Blade Games World and is "still a little rough" (to quote its web page) when compared to

traditional physics middleware packages like Havok or PhysX. For basic physical interactions between collision volumes, however, I found it to be more than adequate. Rag-doll support is still in the planning stages, but the current offering supports rigid body collisions between arbitrary meshes as well as spheres, boxes, capsules and height maps. A custom physics solution handling four-wheeled vehicles is also provided to ease the creation of driving games.

VISUAL SCRIPTING AND EXTENSIBILITY

Blade3D provides a visual scripting system in the form of what it calls Visual Logic Diagrams. Using a flow chart-like representation, this system allows users to define logic by dragging objects and operators onto the design surface and hooking them together to achieve the desired result. These data flow diagrams allow authors to create animation blend trees, specify gameplay logic and define character behaviors. Like most other subsystems in Blade3D, these Visual Logic Diagrams can be edited in real time for easy debugging and tuning.

In addition to the Visual Logic Diagrams, Blade3D also provides an embedded script editor complete with syntax highlighting. Here users can craft C# game components to provide custom extensions to existing game objects. The entire engine and scene editor is also extendable using .NET languages and the provided Blade3D SDK.

MARKETPLACE AND COMMUNITY

Blade Games World has created a development community of its own around Blade3D. The current user forum and wiki are sparsely populated, but growing at a healthy pace. Thankfully, this means new users have great access to the Blade Games World staff, who reply to most forum questions within 1–2 business days. The wiki contains many



empty entries, which Blade Games World will hopefully remedy in the near future. In the mean time, the Blade3D documentation, including both written and video tutorials, handles most common questions and gets users started quickly.

Blade Games World has also launched its own marketplace to allow users to buy, sell and distribute their games and game content. Marketplace browsing and downloads are handled directly through the scene editor, allowing content to be directly embedded into your game project. Like the forum and wiki, the marketplace is just getting off the ground. As a result, the content available is on the slim side.

PRICING

Blade Games World has opted for a novel subscription-based model for Blade3D. Rather than purchasing or licensing the engine with a one time fee or royalty system, users pay a monthly fee to keep the editor operational. Several subscription tiers are available ranging from \$14.95 to \$99.95 a month (\$149.95 to \$999.50 if paid yearly).

The tiers all provide the same feature set and editor functionality, but differ in the level of support provided. Lower-priced tiers also require Blade3D watermarks or splash screens, while the higher tiers require no identifying marks and enjoy discounts to all Marketplace content.

The subscription model is innovative in that it allows

independent developers to get started for very little investment and step up their subscription level as needed over the course of development. The subscription is also tied to the ability to use the editor, not the game product created. This means that once your game is complete, you can freely distribute your product without maintaining a Blade3D subscription. Likewise, the monthly subscription model allows developers the flexibility to cancel and reinstate their subscriptions when the project needs to go on the back burner.

CUT TO THE CHASE

Blade3D is a rich toolset for the indie game creator or the seasoned developer looking for a quick way to prototype ideas. The Blade Games World team has created a simple yet solid world editor, and packed into it a host of features to aid game developers of all skill levels. While some systems lack depth, all facets from animating user interface screens to managing game character inventories are provided in one form or another. When coupled with a responsive community and innovated pricing structure, the result is a welcome addition to any developer's toolbox.

GREG SNOOK is a development lead at Microsoft Game Studios, currently working on an unannounced project. He is also the author of Real-Time 3D Terrain Engines using C++ and DirectX 9 as well as a contributor to the Game Programming Gems book series. Email him at gsnook@gdmag.com.

BLADE GAMES WORLD BLADE3D

★★★★

STATS

Blade Games World
PO Box 1329
Issaquah WA 98027-0053
www.bladegamesworld.com

PRICE

Tiered subscription model starting at \$14.95 per month

SYSTEM REQUIREMENTS

Minimum:
Windows XP with Service Pack 2, 2.0GHz CPU, 1GB RAM, SM 2.0b graphics card, 128MB, display resolution: 1280x1024 or higher, internet connectivity for activation and monthly verification

Recommended:

Windows XP with Service Pack 3 or Windows Vista, 3.2GHz CPU, 2GB RAM, SM 3.0b Graphics Card, 256MB, display resolution: 1600x1200 or higher, 32bit color depth, full-time internet connection for community access

PROS

- 1 Well designed, comprehensive scene editor
- 2 User extendable through visual script, custom components, and .NET development.
- 3 Innovative subscription model

CONS

- 1 Relatively new community and marketplace needs time to mature
- 2 Some content pipeline quirks to overcome
- 3 Single-viewport interface limits scene visibility



MOCK OBJECTS: FRIENDS OR FOES?

UNIT TESTING WITH MOCKING FRAMEWORKS

LAST MONTH WE COVERED ALL THE DETAILS

necessary to start using unit testing on a real-world project. That was enough knowledge to get started and tackle just about any codebase. Eventually you might have found yourself doing a lot of typing, writing redundant tests, or having a frustrating time interfacing with some libraries and still trying to write unit tests. Mock objects are the final piece in your toolkit that allow you to test like a pro in just about any codebase.

TESTING CODE

» The purpose of writing unit tests is to verify the code does what it's supposed to do. How exactly do we go about checking that? It depends on what the code under test does. There are three main things we can test for when writing unit tests:

- *Return values.* This is the easiest thing to test. We call a function and verify that the return value is what we expect. It can be a simple boolean, or maybe it's a number resulting from a complex calculation. Either way, it's simple and easy to test. It doesn't get any better than this.
- *Modified data.* Some functions will modify data as a result of being called (for example, filling out a vertex buffer with particle data). Testing this data can be straightforward as long as the outputs are clearly defined. If the function changes data in some global location, then it can be more complicated to test it or even find all the possible places that can be changed. Whenever possible, pass the address of the data to be modified as an input parameter to the functions. That will make them easier to understand and test.
- *Object interaction.* This is the hardest effect to test. Sometimes calling a function doesn't return anything or modify any external data directly, and it instead interacts with other objects. We want to test that the interaction happened in the order we expected and with

the parameters we expected. Testing the first two cases is relatively simple, and there's nothing you need to do beyond what a basic unit testing-framework provides. Call the function and verify values with a CHECK statement. Done. However, testing that an object "talks" with other objects in the correct way is much trickier. That's what we'll concentrate on for the rest of the article.

As a side note, when we talk about object interaction, it simply refers to parts of the code calling functions or sending messages to other parts of the code. It doesn't necessarily imply real objects. Everything we cover here applies as well to plain functions calling other functions.

Before we go any further, let's look at a simple example of object interaction. We have a game entity factory and we want to test that

the function `CreateGameEntity()` finds the entity template in the dictionary and calls `CreateMesh()` once per each mesh.

We can write a test like the one in Listing 1, but after we call the function `CreateGameEntity()`, how do we test the right functions were called in response? We can try testing for their results. For example, we could check that the returned entity has the correct number of meshes, but that relies on the mesh factory working correctly, which we've probably tested elsewhere, so we're testing things multiple times. It also means that it needs to physically create some meshes, which can be time consuming or just need more resources than we want for a unit test. Remember that these are unit tests, so we really want to minimize the amount of code that is under test at any one time. Here we only want to



ILLUSTRATION BY JON KIM

listing 1 trying to test object interactions

```
TEST(CreateGameEntityCallsCreateMeshForEachMesh)
{
    EntityDictionary dict;
    MeshFactory meshFactory;
    GameEntityFactory gameFactory(dict, meshFactory);

    Entity* entity = gameFactory.CreateGameEntity(gameEntityUid);
    // How do we test whether it called the correct functions?
}
```

listing 2 using mock objects to test object interactions

```
TEST(CreateGameEntityCallsCreateMeshForEachMesh)
{
    MockEntityDictionary dict;
    MockMeshFactory meshFactory;
    GameEntityFactory gameFactory(dict, meshFactory);

    dict.meshCount = 3;

    Entity* entity = gameFactory.CreateGameEntity(gameEntityUid);

    CHECK_EQUAL(1, dict.getEntityInfoCallCount);
    CHECK_EQUAL(gameEntityUid, dict.lastEntityUidPassed);
    CHECK_EQUAL(3, meshFactory.createMeshCallCount);
}
```



test that the entity factory does the right thing, not that the dictionary or the mesh factory work.

INTRODUCING MOCKS

» To test interactions between objects, we need something that sits between those objects and intercepts all the function calls we care about. At the same time, we want to make sure that the code under test doesn't need to be changed just to be able to write tests, so this new object needs to look just like the objects the code expects to communicate with.

A mock object is an object that presents the same interface as some other object in the system, but whose only goal is to attach to the code under test and record function calls. This mock object can then be inspected by the test code to verify all the communication happened correctly.

Listing 2 shows how a mock object helps us test our game entity factory. Notice how there are no real `MeshFactory` or `EntityDictionary` objects. Those have been removed from the test completely and replaced with mock versions. Because those mock objects implement the

same interface as the objects they're standing for, the `GameEntityFactory` doesn't know that it's being tested and goes about business as usual.

The mock objects themselves are shown in Listing 3. Notice they do no real work; they're just there for bookkeeping purposes. They count how many times functions are called, record some parameters, and return whatever values you fed them ahead of time. The fact that we're setting the `meshCount` in the dictionary to 3 is how we can then test that the mesh factory is called the correct number of times.

When developers talk about mock objects, they'll often differentiate between mocks and fakes. Mocks are objects that stand in for a real object, and they are used to verify the interaction between objects. Fakes also stand in for real objects, but they're there to remove dependencies or speed up tests. For example, you could have a fake object that stands in for the file system and provides data directly from memory, allowing tests to run very quickly and not depend on a particular file layout. All the

techniques presented in this article apply both to mocks and fakes, it's just how you use them that sets them apart from each other.

MOCKING FRAMEWORKS

» The basics of mocking objects are as simple as what we've seen. Armed with that knowledge, you can go ahead and test all the object interactions in your code. However, I bet you're going to get tired quickly from all that typing every time you create a new mock. The bigger and more complex the object is, the more tedious the operation becomes. That's where a mocking framework comes in.

A mocking framework lets you create mock objects in a more automated way, with less typing. Different frameworks use different syntax, but at the core they all have two parts to them:

- A semi-automatic way of creating a mock object from an existing class or interface.
- A way to set up the mock expectations. Expectations are the resulting interactions a successful test will produce: functions called in that object, the order of those calls, or the parameters passed to them.

Once the mock object has been created and its expectations set, you perform the rest of the unit test as usual. If the mock object didn't receive the correct calls the way you specified in the expectations, the unit test is marked as failed. Otherwise the test passes and everything is good.

GOOGLE MOCK

» This is the free C++ mocking framework provided by Google. It takes a very straightforward implementation approach and offers a set of macros to easily create mocks for your classes, and set up expectations. Because you need to create mocks by hand, there's still a fair amount of typing involved to create each mock, although they provide a Python script that can generate mocks automatically from C++ classes. It still relies on your classes inheriting from a virtual interface to hook up the mock object to your code.

Listing 4 shows the game entity factory test written with Google Mock. Keep in mind that in addition to the test code, you still need to create the mock object through the macros provided in the framework.

MOCKITNOW

» This open-source C++ mocking framework written by Rory Driscoll takes a totally different approach than Google Mock. Instead of requiring that all your mockable classes inherit

listing 3 mock objects for the previous test

```
struct MockEntityDictionary : public IEntityDictionary
{
    MockEntityDictionary()
        : meshCount(0)
        , lastEntityUidPassed(0)
        , getEntityInfoCallCount(0)
    {}

    void GetEntityInfo(EntityInfo& info, int uid)
    {
        lastEntityUidPassed = uid;
        info.meshCount = meshCount;
        ++getEntityInfoCallCount;
    }

    int meshCount;
    int lastEntityUidPassed;
    int getEntityInfoCallCount;
};

struct MockMeshFactory : public IMeshFactory
{
    MockMeshFactory() : createMeshCallCount(0)
    {}

    Mesh* CreateMesh()
    {
        ++createMeshCallCount;
        return NULL;
    }
};
```

from a virtual interface, it uses compiler support to insert some code before each call. This code can then call the mock and return to the test directly, without ever calling the real object.

From a technical point of view, it's a very slick method of hooking up the mocks, but the main advantage of this approach is that it doesn't force a virtual interface on classes that don't need it. It also minimizes typing compared to Google Mock. The only downside is its very platform-specific implementation, and the version available only supports Intel x86 processors, although it can be re-implemented for PowerPC architectures.

PROBLEMS WITH MOCKS

» There is no doubt that mocks are a very useful tool. They allow us to test object interactions in our unit tests without involving lots of different classes. In particular, mocking frameworks make using mocks even simpler, saving typing and reducing the time we have to spend writing tests. What's not to like about them?

The first problem with mocks is that they can add extra unnecessary complexity to the code, just for the sake of testing. In particular, I'm referring to the need to introduce a virtual interface for each object that we want to mock. This is a requirement if you're writing mocks by hand or using Google Mock (not so much with MockitoNow), and the result is more complicated code: You need to instantiate the correct type, but then you pass around references to the interface type in your code. It's just ugly, and I really resent that using mocks is the only reason those interfaces are there. Obviously, if you need the interface and you're adding a mock to it afterward, then there's no extra complexity added.

If the complexity and ugliness argument doesn't sway you, try this one: Every

unnecessary interface is going to result in an extra indirection through a `vtable` with the corresponding performance hit. Do you really want to fill up your game code with interfaces just for the sake of testing? Probably not.

But in my mind, there's another, bigger disadvantage to using mocking frameworks. One of the main benefits of unit tests is that they encourage a modular design, with small, independent objects that can easily be used individually. In other words, unit tests tend to push design away from object interactions and more toward returning values directly or modifying external data.

A mocking framework can make creating mocks really easy, to the point that it doesn't discourage programmers from creating a mock object any time they think of one. And when you have a good mocking framework, every object looks like a good mock candidate. At that point, your code design is going to start looking more like a tangled web of objects communicating in complex ways, rather than simple functions without any dependencies. You might have saved some typing time, but at what price?

WHEN TO USE MOCKING FRAMEWORKS

» That doesn't mean that you shouldn't use a mocking framework though. A good mocking framework can be a lifesaving tool. Just be very, very careful how you use it.

The case when using a mocking framework is most justified when dealing with existing code that was not written with unit testing in mind—code that is tangled together, and impossible to use in isolation. Sometimes that's third-party libraries, and sometimes it's even (yes, we can admit it) code that we wrote in the past, maybe under a tight deadline, or maybe before we cared much about unit tests. In any

resources

Google Mock

<http://code.google.com/p/googlemock>

MockitNow

<http://www.rorydriscoll.com/mockitnow>

case, trying to write unit tests that interface with code not intended to be tested can be extremely challenging. So much so, that a lot of people give up on unit tests completely because they don't see a way of writing unit tests without a lot of extra effort. A mocking framework can really help in that situation to isolate the new code you're writing, from the legacy code that was not intended for testing.

Another situation when using a mocking framework is a big win comes if you use it as training wheels to get started with unit tests in your codebase. There's no need to wait until you start a brand new project with a brand new codebase (how often does that happen anyway?). Instead, you can start testing today, and using a good mocking framework will help isolate your new code from the existing one. Once you get the ball rolling and write new, testable code, you'll probably find you don't need it as much.

Apart from that, my recommendation is to keep your favorite mocking framework ready in your toolbox, but only take it out when you absolutely need it. Otherwise, it's a bit like using a jackhammer to put a picture nail on the wall. Just because you can do it, it doesn't mean it's a good idea.

Keep in mind that these recommendations are aimed at using mock objects in C and C++. If you're using other languages, especially more dynamic or modern ones, using mock objects is even simpler and without many of the drawbacks. In a lot of other languages, such as Lua, C#, or Python, your code doesn't have to be modified in any way to insert a mock object. In that case you're not introducing any extra complexity or performance penalties by using mocks, and none of the earlier objections apply. The only drawback left in that case is the tendency to create complex designs that are heavily interconnected, instead of simple, standalone pieces of code. Use caution and your best judgment and you'll make the best use of mocks. ☺

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

listing 4 test using Google Mock

```
TEST(CreateGameEntityCallsCreateMeshForEachMesh)
{
    MockEntityDictionary dict;
    MockMeshFactory meshFactory;
    GameEntityFactory gameFactory(dict, meshFactory);

    EXPECT_CALL(dict, GetEntityInfo())
        .Times(1)
        .WillOnce(Return(EntityInfo(3)));

    EXPECT_CALL(meshFactory, CreateMesh())
        .Times(3);

    Entity* entity = gameFactory.CreateGameEntity(gameEntityUid);
}
```



METAGAMES

THE HIDDEN LIFE OF FILES

SOME YEARS AGO YOUR HUMBLE correspondent was racing to meet a milestone, frantically hunting for files that needed updating before a big demo. After an hour of angry wrestling with the windows file dialog (you know, the one with helpful the ability to accidentally create new folders while opening files) I hurled my mouse across the room, spewed a stream of profanities, and loudly proclaimed my intention to hunt down and eviscerate the creator of the offending dialog who, I quickly discovered, was now working two desks away as our lead designer. Although it didn't end up in fisticuffs, it was still disheartening to hear that Windows team hated the damn thing too. From that day to this, every time I type a file backlash, something inside me dies.

The main moral of the story is, of course, think ahead before you start cussing in an open plan office, particularly if you work two miles from the Microsoft main campus. However, it also illuminates the seamier underside of game artist life. We've spent many column

check-in list? Nobody comes away from a group critique thinking "damn, I really need to brush up on my file naming skills or I'll never make lead!" File handling rarely tops your artistic agenda.

METACRITICAL ISSUES

» So naturally, it sounds like just the sort of thing you'd love to read about in *Game Developer*. No, seriously! Before your eyes glaze over completely, here's a very quick pitch about why you should spare a little thought for this humble aspect of the artistic life:

First, you spend a lot of time dealing with files. Each interaction is pretty negligible in itself—but there are lots of them. How many times a day do you have to start digging around for something buried five or ten layers deep in your game directory, or worse in a clunky, slow, source control browser app? Those seconds can add up to a sizeable chunk of your day. Worse, those little clerical delays also break your concentration instead of keeping you on task. You shouldn't have

Poor organization also means that changes of direction can be prohibitively expensive. There is an embarrassingly large number of companies which can't actually recreate their game content because nobody knows what files really produced the in-game assets.

inches in Pixel Pusher pondering the subtle inflections of the word "art." It's a safe bet, though, that the none of the competing definitions of art revolve around efficient file management or solid naming conventions. What art school kid wakes up in the morning and dreams of a cleanly organized

to care about the files, you should care about the artwork.

Second, poorly organized files make debugging difficult, frustrating, and costly. Debugging your work in the game is tricky enough to begin with, with all the thousands of things that could go wrong in your art package, or in



ILLUSTRATION BY HARVEY JAMES

the game itself. If you aren't 100 percent certain that the art file you're looking at is actually the source of what you see in the game, it quickly becomes impossible. Poor organization also means that changes of direction can be prohibitively expensive. There is an embarrassingly large number of companies which can't actually recreate their game content because nobody knows what files really produced the in-game assets. If your company is one of those with a big storage closet of artists' hard drives kept "just in case," you know what we mean.

your organization, but in the end your real job is to tell stories and fire up players' imaginations, not to shuffle binary data around in tidy parcels. Managing your files well is dreadfully important, because it can really get in your way if done poorly; but a good solution is one that occupies as little of your time and attention as possible.

To sum up: good organization is critically important, but no fun. The best thing to do about it, therefore, is to automate and regularize the management of your files in every way you can. You need the safety of a reliable audit trail and the convenience of quickly getting to your stuff; but you don't want to waste precious artistic synapses deciding what to call things and where to put them.

METASTASIS

» To tackle the problem seriously, it's worth thinking a little about the basic assumption we bring to the problem. The idea of putting "files" into "folders" as a way of organizing digital information goes all the way back to the 1950s. It's held up a little better than paper punch cards as a storage medium—but only just. It has serious limitations—we're just so used to them that we don't always

be inspired



14-16 JULY 2009 DEVELOP in Brighton

The Develop Conference is an inspiring place – over 80 great sessions given by a host of international development experts and around 1200 developers getting together to share ideas, learn from each other and socialise.



Plus this year there's innovative new content with the launch of **EVOLVE** – a new one-day event which will open the Develop Conference on Tuesday 14 July and a new track within the conference on Wednesday 15 July.

Here's a taste of this year's programme:



Conference Keynote
David Jones, Founder, Realtime Worlds



Keynote Out of the Box(ed Product): Thinking for an Online Age
Jeff Hickman, Executive Producer, Mythic Entertainment



Keynote Resetting the Game
David Perry, Industry Consultant



Keynote The Art of LittleBigPlanet – A Big Medley
Kareem Ettouney and Mark Healey, Co-founders, Media Molecule



Open Software for Closed Hardware
Steve Goodwin, SGX Engine



Keynote Bridging The Gap Experiences Learned with Agile Project Management across Multisite, Multicultural and Multilingual Project
Lisa Charman, Ubisoft and Patric Palm, Hansoft



Keynote The Runtime Studio in Your Console: The Inevitable Directionality of Game Audio
Guy Whitmore, Director of Audio, Microsoft Game Studios



Keynote Building LEGO Worlds – online, offline, and everything in between
Jonathan Smith, Development Director, Travellers Tales



Making Videogames History: Starting the National Videogames Archive
Iain Simons, National Videogame Archive

Other speakers confirmed include:

Autodesk • Bizarre Creations • Blitz Games • Chillingo • Climax • Creative Assembly • comScore • Crytek • Denki • Disney Black Rock Studios • Eutechnyx • Fishlabs • FluffyLogic • Glu Mobile • Google • Guerrilla Games • Gusto Games • ICO Partners • Kerb • Lightning Fish Games • Lionhead • Matmi • MySpace • Nokia • Mediatonic • Microsoft • ngmoco • Playfish • Rare • Team 17 • The Mustard Corporation • Silicon Knights • Sidelines • Tag Games • Zoe Mode



Make sure you stay ahead of the game – come to Develop in Brighton!

www.developconference.com



Mobile Sponsor

International Media Sponsor

Media Sponsor

Media Sponsor

Media Sponsor

Media Sponsor

Media Sponsor



Media Sponsor

Media Sponsor

Media Sponsor

Media Sponsor

Member Discounts

Organised by





perceive them as problems we can fix.

The fundamental problem with folder hierarchies is that they are better at hiding information than finding it. In a real production, every game asset falls into lots of categories. We try to use folders to reflect this, hence familiar paths like "objects\characters\human\allied\grenadier.ma" This is descriptive, sure—it's an object, it's a character, and so forth. But different people who care about that asset care about different aspects. The level designer placing enemies cares about the faction the asset belongs to. The character artist wants to see all the characters and not worry about level files. The animators want to distinguish between game animations and cinematics. The engineers may want to shuffle files around for efficient disk loading. Not surprisingly, each of these groups will have different ideas about the right way to organize the file tree.

Fighting over the "right" way to organize a project's folder hierarchy is a grand old game business tradition. Every team starting out fresh on a virgin project is certain they can design a file tree that really works (not like that shambolic mess they had last time!). It's a quaintly noble ambition; but it's not going to happen. Folder hierarchies can't reconcile the competing agendas of different groups within the team. Moreover, the folks setting up the hierarchy at the beginning of the project have only a vague idea about what the real contents of the shipping game will be like. Character names and roles will change, working files with joke names will turn into critical assets, and big areas of the game will be cut. The end result is like a trackless jungle—without an experienced guide, you'll get lost inside.

This messiness has real costs. Programmers who work with tree structures for a living say that the ideal tree is "balanced"—the number of choices at each sub-level is more or less the same. The goal of course is that the path to anything in the tree should be as short as possible. If you're sick of typing things like "C:\documents and settings\myself\project\content\art\models\objects\

vehicles\damaged\alien" into the file box, you can see the point. However the "efficient" arrangement may not be the same as the understandable one, or the one that groups things logically for any individual task. Typing time isn't the only cost: your tools also have to contend with this haphazard structure. If your scripts include page after page of hard-coded path strings and arcane "up two folders and over one" rules, bugs are the inevitable result. Worst of all, organizational awkwardness increases the likelihood of lost files, duplicated work, and broken dependencies.

METAPHYSICAL INSIGHTS

» So, we know this is a tough problem, and we know it's important. Does this mean we must all become file Nazis, or start hiring file-management interns? No. Even if we wanted to tackle it manually, the demands of managing a modern game project are too harsh. It's not realistic to think it can be handled with email memos and threats from management. It's a database problem, and it should be handled by computers—not artists with more important, human tasks to do.

Actually, everyone already knows the technological answer to this problem. It doesn't depend on 1950's-era office practices. It doesn't demand a degree in computer science, either. Most of us maintain music libraries with thousands, or even tens of thousands of songs. But only the hardest of the hard core users actually design a folder tree to sort those files into a tidy hierarchy—we all let iTunes or Windows Media Player or WinAmp handle the physical location of the files, while we browse through the library using many different cues for sorting and organizing. We can find things by genre, by date, by artist and so on. We can even create playlists containing any random collection of stuff we like. This seems mundane, but in fact it is a sophisticated database system. It's sophisticated in the best sense—so good we take it for granted.

Obviously we can't manage our games in WinAmp. But the

underlying tech that makes music libraries manageable is something we can and should introduce into our studios. Music players use the mp3 tag information to record the relevant information about each file or album. The nerd term for this is "metadata."

Using metadata, files aren't defined solely by their physical location; they can be found by many different types of clues. Unlike folders, metadata "tags" aren't mutually exclusive: Bits of my Louis Armstrong collections might be available under "Jazz", "Swing", "Vocals" or "Blues," or by date, or by rating. Your Alif Tree remixes might appear under "Electronica" or "French Hip Hop," or you might just look for the name of the song or album. Organizing by metadata turns the rigid, hidebound business of filing into the much more efficient process of finding what you need by whatever route suits your memory at the moment. It doesn't force you to actually shuffle things around on disk if your needs change.

GOOD METABOLISM

» It's easy to see how good metadata can help teams manage enormous masses of art content. Instead of wrangling folder arrangements, you can concentrate on finding and working with the data you need. You can design a set of metadata tags that match the needs of your production. Your animators care a lot about which files are animations and which are models, but they don't care much if a model file is a level or a vehicle; your producers will want to sort by production status; your art leads may want to browse work by author at review time. Happily, you can add new metadata as new needs arise, without having to physically shuffle files around on disk.

Metadata also lets you tackle one of the nastiest problems in game art management: tracking the relationships between files.

Most companies recognize that accounting for files which get exported into the game is a life-or-death issue. However, plenty of critically important files never go into the game directly. Many teams, for example, do a terrible job of dealing with the high-res source art that is used to generate normal maps—since the zillion-poly Mudbox models never go into the game directly, they often end up completely outside of source control, and their vital role in the game resides entirely in the memory of individual artists. I've personally had the experience of asking where the Zbrush source for a character was kept and hearing that it lived in a folder called "junk drawer" in somebody's email drop box. It's far too common for artists to check in complex assets without remembering to check all of their incoming references, leaving files that can only be opened on the author's machine. Metadata helps to keep tabs on the myriad bits of information that really make up the final game assets.

METADORS

» Good file tracking is like clean underwear—most of the time nobody knows if you've got it, but should you get run over by a bus, people will be grateful that you were prepared. Hopefully, by this point you're convinced of two basic points. First, that file management (dowdy and dull as it may be) is a critical part of game art production. Second, that there's a better way to handle it than with old-fashioned folders and ham-handed naming conventions, by using the power of metadata. Next month we'll look at some practical steps you can take to actually build a metadata based pipeline and realize some of these benefits in your daily work. Until then, remember that the LEFT hand yellow folder icon goes up one folder, and the RIGHT hand one makes a new folder. Sheesh. ☹

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



TACTICAL TRANSPARENCY

EXPOSING THE BOARD STATE TO PLAYERS

A LOT OF EXPLANATIONS HAVE BEEN

given for the explosion of poker in the early part of this decade. Factors cited have included in the rise of online poker, the surprise victory of amateur Chris Moneymaker in the 2003 World Series of Poker, the success of the movie *Rounders*, and even the NHL lockout which left ESPN scrambling for cheap content to show in winter months. I'd like to propose one additional reason: the rise of a superior form of poker.

For decades, when you saw a game of poker being played in a movie, what you saw being played was probably Five-Card Draw—all players are dealt a hand, may replace some cards in a single draw, and then reveal, with opportunities to bid along the way. The dirty secret of Five-Card Draw is that it's not a very good strategy game. Players have little information to base their strategy on—their own hand, how many cards their opponents draw, and how nervous their opponents seem. Draw poker is entirely about bluffing and luck. This makes for classic cinema, but from a gameplay perspective, it's hardcore and fairly unsatisfying to play.

But that's not what they play on ESPN2 at 2 AM. As poker exploded, Texas Hold'em was the game of choice. In Texas Hold'em, all players must make a five-card hand out of two cards they have privately (their 'hole' cards) and five others that everyone shares. Suddenly, you have very good information about what your opponents can do—three-fifths of their final hand is on the table, after all. The odds of victory and defeat becomes as much a math problem as one about bluffing and luck.

Bluffing is still important, but it isn't the dominant path to victory. Strategy is—and this strategy is created by the amount of information given to the player. As game designers, we must understand that the level of information we give players affects how strategic or tactical our games are.

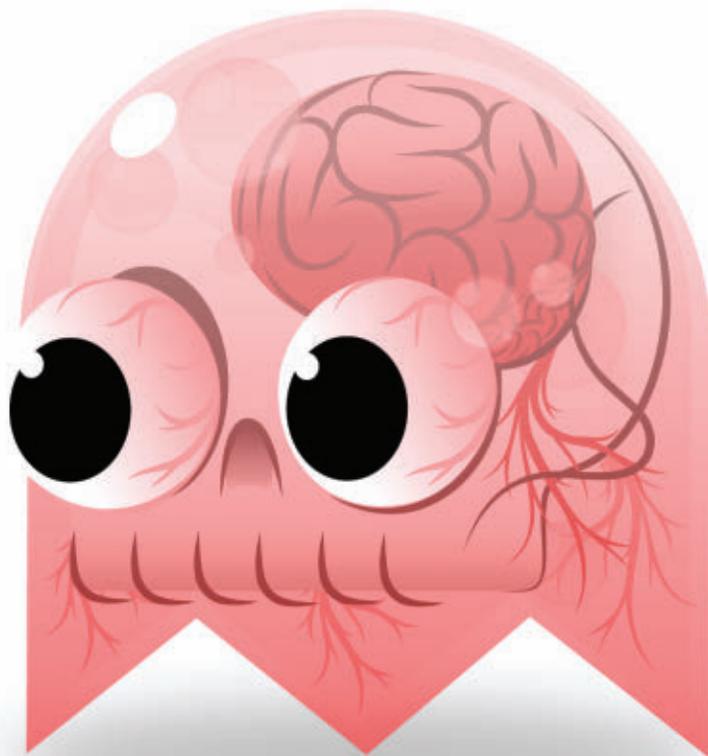


ILLUSTRATION BY DEREK YU

INTRODUCING TACTICAL TRANSPARENCY

» When discussing games, the term "board state" refers to the current state of variables in the game. In chess, the board state is the location of all of the pieces. In poker or *Magic: The Gathering*, the board state is what's on the table as well as in the player's hand. In a first-person shooter, it would be the health, weapons, and ammo available to both you and your opponent.

Tactical transparency is a description of how much of the board state is exposed to each player. In classic chess or checkers, the board state is entirely exposed to both players. Every piece of information a checkers player could use is in front of him. Surprises are failures on the part of the player to foresee all possible outcomes of the current board state. The information available to the player is perfect.

IMPERFECT INFORMATION

» Of course, most games do not expose their entire board state. Many successful games shroud some of the board state, forcing players to adapt tactically as the game unfolds. In many cases, however, you offer them hints. These hints are imperfect information that allow the players to make informed decisions without necessarily ensuring they will make the right one—or even that a winning move exists.

Blackjack is not perfectly tactically transparent—the dealer's second card is hidden—and the game would be quite broken if it were transparent. That said, the player knows two key pieces of information about the dealer's board state: the dealer's top card, and that the dealer is bound by the



rules as far as when he can hit or stay. These two bits of information are enough that a smart player playing optimally can very nearly erase the house edge in the game.

In *Magic: The Gathering*, you normally have no idea what cards your opponent has in his hand. However, you do know what resources are available to him—primarily the mana base he has on the table. If he has nothing but red mana, you know that he could cast direct damage spells like Lightning Bolt and Disintegrate. Even more importantly, you know what he can't do—he can't, for example, cast a Counterspell, as those require blue mana. This information allows the player to build a battle plan.

Game resources are often a good way for a designer to create imperfect information. If a missile goes whistling over your head in *QUAKE 2*, you know your opponent has a rocket launcher, which is useful information. You don't know how much ammo he has, but you do know that a player can't carry more than 50 rockets. Sometimes, imperfect information can be made more perfect if the player is willing to track it. The resources that a player has drawn in *Settlers of Catan* are kept hidden, but if you're willing to track them as they are distributed and spent, you can gain a pretty good idea of an opponent's capabilities.

TACTICAL TRANSPARENCY IN NON-STRATEGY GAMES

» The concept of tactical transparency is most strongly adhered to in turn-based strategy games, of course, as fans of the genre value information highly. However, even games well outside of the strategy genre can benefit from the philosophy. One example is the "light meter" in games like *THIEF* or *SPLINTER CELL*.

The light meter is a UI element that shows if the player is successfully hiding in the shadows, and how likely the AI is to detect the player. I'm sure that some designer on the *THIEF* team tried to argue that having an extra GUI element displaying this information was unrealistic and immersion-breaking. But the opposite is true—by giving the player perfect information about how visible the game thinks the player is, the player learns much quicker what is actually good stealth and what is not, and ceases to worry about external forces that could affect their perceptions: camera angle, monitor glare, or glitches in the game's lighting algorithm. The player's focus may be on an artificial UI element, but he is much deeper into the stealth experience as a result.

In platformers or first-person shooters almost every bad boss fight I can think of is ultimately a failure of tactical transparency. Boss creatures tend to be about megahits from the

creature, or moments of unique vulnerability. In bad boss fights, these aren't communicated well to the user. When the player defeats such a creature, he feels that it is more due to luck than skill. And while winning due to luck is fun, it doesn't nearly match the sense of mastery that a victory based on skill or tactics provides.

Tactical transparency can affect even basic decisions a player has to make. In most games with a height field, there is an angle at which the terrain is simply too steep to climb—let's say 60 degrees. The problem is that slopes at 59 and 61 degrees look identical. Trying to climb both will make the world feel wildly inconsistent.

To ensure there is no ambiguity, world builders should painstakingly ensure that steep terrain is clearly visually different, such as being painted with a different texture. Slopes near the non-climbable threshold should be forbidden (i.e. no slopes between 50 and 70 degrees). Designers may argue that it limits their freedom create immersive environments. However, few world building decisions break immersion more than hitting an unclimbable slope the player wasn't expecting while he's desperately running for his life.

CASE STUDY: CLASSES IN MMOs

» One of the classic design debates is whether MMOs should have rigid classes, or whether players should be able to combine skills with freedom. While there are good arguments on both sides, one interesting way to look at the debate is to see how tactical transparency affects PvP.

In *WARHAMMER: AGE OF RECKONING*, if you see a Witch Elf, you know her capabilities—she can use stealth, debuff via poisons and deadly kisses, and do a lot of burst damage. She also is a lot less deadly if kept at range. This information is still imperfect: you don't know her mastery path, her gearset, or any of the other ways she's fine-tuned her character spec. Still, if you run across a Witch Elf, the strategy and tactics you employ are going to be very different than if you stumble across a Sorceress.

If you couldn't tell a Witch Elf from a Sorceress, strategy would become impossible until combat engaged and you saw what abilities your opponent was using—and then the decisions that you made would be purely twitch-speed tactical. In a pure classless environment, in which any character can have any combination of abilities, strategy disappears. You know nothing about the combat capabilities of your opponent, and cannot proactively make adjustments to deal with them. Without good information on which to base tactical decisions, most players will

devolve their tactics to a single abilities rotation that works the majority of the time.

This is not to say that classes are the right answer. But advocates of classless systems need to understand that if they don't increase tactical transparency in the absence of what classes have to offer, PvP tactics may be throttled in the crib.

INFORMATION WANTS TO BE FREE

» Too many designers try to keep too much hidden. Many are in love with the idea of surprising the players. The problem is that such surprises often feel unfair and capricious, and force the player into a pattern of trial and error to learn their limits. Perhaps if we called this pattern "die-and-quickload," designers will finally understand how unfun it is.

Another culprit is the urge to hide numbers and information. This can be a good urge—too many numbers, and you risk making the game overly complex and, by extension, more hardcore. Still, hiding the numbers is a case of treating a symptom, not the problem. The problem, all too often, is that there are too many variables, and that too much information is necessary to understand the board state.

Hiding the number is the wrong approach. Making the players choose between a 'strong upgrade' and a 'great upgrade' is much more frustrating than making them choose between +6 or +8 strength. Rather than hide the information, designers should expose it, while still ensuring that the mechanics are as simple as possible, so players don't have to be a Mensa member to understand the trade-offs.

Finding the right balance of tactical transparency for your game is a balancing act. If a game requires no tactics or strategy and success is guaranteed, then it risks becoming a forgettable exercise in button mashing. If a game awards victory only in limited conditions, but does not give players enough information to strive toward that condition, the game becomes an exercise in luck, as is the case in *Five Card Draw*. How much should you expose? This will vary from game to game, but in my experience, the designer should err on the side of more. Sometimes it may seem like you're exposing too much, but let's not forget, chess is often considered the greatest game of all time—and it shows everything.

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



THINKING OUTSIDE THE BOOTH

TAKE YOUR VOICE ACTORS OUT OF ISOLATION

AS MUCH AS THE PRACTICES AND tools for Audio continue to evolve with every new game the industry produces, one area of our discipline that remains fairly static is voice. For the most part, voice work in games is patterned after the model established by the animation industry. Voice directors cast roles with an eye toward actors familiar with the idiosyncrasies of voice-over work. The talent is recorded one at a time, in isolation, while standing in front of a microphone with the voice director in the control room at the other end of the talkback mic.

That's the traditional approach—but there are other approaches to game voice recording that can help to bring much needed realism, sincerity, and believability to your actors' performances.

A DIFFERENT DIRECTION

» With the performance of cinematic game dialogue moving from one of audio-only into the realms of facial and full-body motion capture, the goal across the board becomes increased attention to realism. Additionally, these changes in technology are driving a secondary change away from solely using voice-over actors, and instead bringing more on-camera actors into the world of game voice production.

To help achieve a more true-to-life performance out of the actors, one step the voice director can take is to move away from the standard practice of recording actors in isolation. Actors are used to not only acting but also reacting to the performances of other cast members. This is particularly true of on-camera actors. Directors can choose to record actors performing together in groups, or to have the other actors present at the session, off-camera. This process is called a cast record. Another option is for the

talent director to sit in the same room with the performing actor and read the lines of the other characters. This approach to direction can yield a much more realistic performance because the actors are no longer working in a vacuum, but rather reacting to the performance and delivery of another human being. The vocal performances are often more dramatic, more dynamic, and more convincing.

However, if you're working with seasoned voice-over actors, this approach to direction will most likely frustrate them more than it will help the process along. Know your actor's skill set and what will get the best performance out of them before forcing experimentation upon them.

A SEAT AT THE TABLE

» Another approach that can be helpful for improving the quality of the voice assets in your game is simply the use of rehearsal. Most voice actors read their lines cold, meaning they're handed the script when they walk into the session, and the first time they're performing it is as it goes into Pro Tools. However, in some instances, a great opportunity toward having your cast understand the scope of their characters and how they fit within the larger game is to conduct a table read. A table read is a rehearsal where the core character actors literally get together around a table and read through the script alongside of the voice director, and usually a few key team members such as the creative lead of the project, the scriptwriter, and a producer or two.

A table read is a great tool to use when the game has a group of no more than seven core characters and a fairly involved cinematics script. There's no point in sitting around and rehearsing AI grunts and barks such as "He's over there!"—but if

your game is centered around a rich story, a table read will help to create a cohesive performance that allows the actors to know the full context of their performances. Again, as with a cast record, a table read works best for on-camera actors, as voice-over actors are much more used to cold reading their scripts as final performances.

BLOCK IT OUT

» Lastly, with the ever-increasing practice of motion capture being used not only for in-game animations but also cutscene performances, voice directors may find themselves tasked with directing both actors and stuntmen on the motion capture stage. As such, voice directors should consider holding rehearsals the day before a major cutscene motion capture session in order to establish and rehearse the physical blocking of the scenes. Once your actors become mobile, pre-determined blocking becomes a crucial tool in constructing the scene and avoiding awkward or unsuccessful improvisations from your moving characters.

Even if your game won't include new technologies such as digital facial scanning and motion capture, consider this less-isolated approach to voice recording. Perhaps more than any other element, the current generation of games is pushing the boundaries of realistic virtual worlds. At the heart of any compelling virtual character performance is a convincing vocal. Any steps the voice director can take toward maximizing believability will help to sell not only the character, but also the entire world in which they live. 🎧

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.





THE MAKERS OF EVE ONLINE ARE ACTIVELY RECRUITING LEADING TALENT

QA MANAGER • TECHNICAL DIRECTOR • SENIOR PROGRAMMERS
ARTISTS • GAME MASTERS • QA ENGINEERS

» ATLANTA • REYKJAVIK • SHANGHAI «

VISIT US ONLINE AT WWW.CCPGAMES.COM/JOBS TO LEARN MORE AND APPLY TODAY

© 2005 CCP Inc. All rights reserved. Reproduction without the written permission of the publisher is expressly prohibited. EVE, EVE Online, CCP logo are registered trademarks of CCP Inc.



THE MAD DOCTOR WANTS YOU TO APPLY...

**...OR THE
GIRL
GETS IT!!!**



TALENTED GAMES FOLK APPLY!
WWW.NEVERSOFT.COM

MAKE MORE ENEMIES

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

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

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



VFS student work by Thaddeus Maharaj

VFS

vfs.com/enemies

Connect with the Largest Educational System in the World
110 Colleges, 2.6 million students, affordable customized training

MEI Game & Simulation SPECIAL INTEREST GROUP

meigamesim.intronetworks.com

Train Your Employees
Apply for State and Federal Grant Programs
Access Business Development Resources

California Community Colleges

Multimedia & Entertainment Initiative

ECONOMIC & WORKFORCE DEVELOPMENT

Advancing the Business of Creativity

CENTER FOR DIGITAL IMAGING ARTS AT BOSTON UNIVERSITY



capture
imagination

3D ANIMATION
+ INTERACTIVE MEDIA

GAME ART + CHARACTER ANIMATION

..... certificate programs

TWO CAMPUS LOCATIONS

WALTHAM, MA & WASHINGTON, DC

Financial assistance and career services available.

Now accepting applications. **Apply today!**



CDIABU.COM

GET EDUCATED

ADVERTISER INDEX

COMPANY NAME	PAGE
CCP North America	52
Center for Digital Imaging	55
Epic Games	19
Havok	3
Image Metrics	C3
Insomniac Games	14
Intel	C2, 28-33
Jones and Bartlett Publishers	25
Neversoft	53
Pocket Soft	41
Rad Game Tools	C4
Santa Barbara City College	54
Spiel Studios	25
The MIT Press	39
Vancouver Film School	54



GAMASUTRA

The Art & Business of Making Games



The Leading
Professional
Game Source
for Job Seekers
and Employers

Access the Web's Largest Game Industry
Resume Database and Job Board

Take Control of Your Future Today!
Log onto www.gamasutra.com/jobs



Game Developer (ISSN 1073-922X) is published monthly by United Business Media LLC, 600 Harrison St., 6th Fl., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as United Business Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is \$49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: \$69.95; all other countries: \$99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 4601 W. 6th St. Suite B, Lawrence, KS 66049. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence. All content, copyright *Game Developer* magazine/United Business Media LLC, unless otherwise indicated. Don't steal any of it.



ASK A PIZZA

STRAIGHT TALK WITH GAME DEVELOPMENT'S MOST CONSUMED FOOD

IF YOU'VE BEEN IN GAME DEVELOPMENT, IT'S A GIVEN THAT YOU'VE EATEN more than your fair share of pizza. This month, *Game Developer* magazine was fortunate enough to land an interview with this widely eaten foodstuff — and one of our industry's unsung heroes.

What's it like being the crunch food of choice for game developers?

It's pretty cool. I like knowing that my calories are going to a good cause instead of, say, a local frat party. It's really exciting that there's a chance my carbs and fats will turn into the fuel for the next great advance in video games. Of course I know that most of the time I'll be the fuel to just barely ship an inferior sequel to some insipid licensed title. But I have to hold out hope just like everyone else in this business, right? Plus, in comparison to the frat party, there's a reduced chance I'll be vomited back up later.

Do people on the team ever get upset that the producers just order pizza over and over?

Oh, sure, there are complaints. I've endured my share of the sarcastic, "Oh, awesome, freaking pizza again!" comments. I try not to let that affect me. Obviously, anyone would get tired of the same thing over and over, even a food as delicious as myself. To deal with it, I rationalize. I know that part of the reason I appear so often is that I'm easy to order and I can be ready at a moment's notice, which is important since the producers usually forget until the evening, even though it's totally obvious the team will be there late. Oh, and I'm cost effective, too! So that's a plus as well.

Seems like you've really been around the block! What are the different ways you've been consumed?

Well, I'd say the main way is what I call the "shovel," which kind of looks like what it sounds like! [laughs] Most people tend to eat pizza the same way — put as much as you can in your mouth and just freaking chew, know what I mean? It's not like people think about it much. They've got so much else on their minds anyway. I guess there's a little variation, now that I think about it. Some people leave the crusts, and other people do weird stuff like put mayonnaise on me. I think that's a European thing. What else? Oh yeah, smokers can't taste anything so they load up on the crushed red pepper and Tabasco. That's about all I can think of right now.

Do you have any particular advice for people ordering you in the future?

Hmm. I'd say, try to vary it up a little. Sure, your favorite might be the Extreme Meat Supreme, but other folks might appreciate the Mega Meat Combo, and still others might want the Jumbo Cornucopia of Meats with the Meat-Stuffed Crust. All of those are legitimate choices, so a selection between them will go a long way to pleasing as many people as possible in a single order.

Oh yeah, and don't forget those hippie vegetarians. I can't tell you how many times those whiny people have come into the kitchen and recoiled in horror at the sight of me. So you might want to do something for them just



ILLUSTRATION BY CHRIS FORD

so they freaking pipe down, if you feel me. And don't even get me started on lactose intolerance.

Recently, there's been concern about "Quality of Life" in certain sectors of—

You mean the people who are mad about crunch and don't want to do it? Yeah, I've heard of those folks. I've got something to say to them, so I hope they're listening. You can talk all you want about working nine to five and how the long hours are burning people out, but as long as I'm around, I think it's gonna be pretty difficult to change anything. What game developer could possibly turn down the prospect of a fresh, mouth-watering pizza? I mean, seriously.

So you're saying that you make crunch tolerable to—

Tolerable? Look, eating pizza is a communal thing. People do it together. There's a certain camaraderie that comes into play when it's late at night and game developers are huddled around the table shoving me down their throats together. As long as I'm there to "grease" the wheels, so to speak, I don't see why it would change. I'm as much a part of the system as anything else.

Oops, looks like we're out of time. Well, thanks for chatting with us!

My pleasure. See you all during your next crunch! 🍕

MATTHEW WASTELAND is a pseudonymous game developer who has a fairly common first name. Email him at mwasteland@gdmag.com.

RACERS HAVE FACES TOO



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

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

Join our cause today by calling **310.656.6565**
or by visiting image-metrics.com.

image metrics

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



Proud Sponsor of:

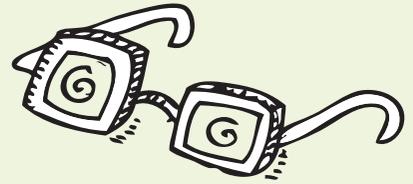
FRAHG
FRIENDS AGAINST
HELMETS in GAMES



WHAT'S IT LIKE USING

RAD

GAME TOOLS ?



.....

It feels like you have



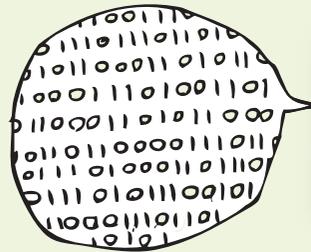
when you use Bink - our amazing, super *FAST* video and audio codec - all in a simple, clean API.



You'll fall in LOVE with Granny, our run-time dynamic

3D ANIMATION SYSTEM

with AMAZING content exporters.



And with Miles, our multichannel



sound system, you get the world's *FASTEST*

MP3 and Ogg DECODERS

+ way cool *new* high-level audio tools.

.....

IN A WORD, USING OUR TOOLS IS rad!



www.radgametools.com
(425) 893-4300