

B N M Z E A G A W

D E V E L O P E R S E E D

M A G A Z I N E



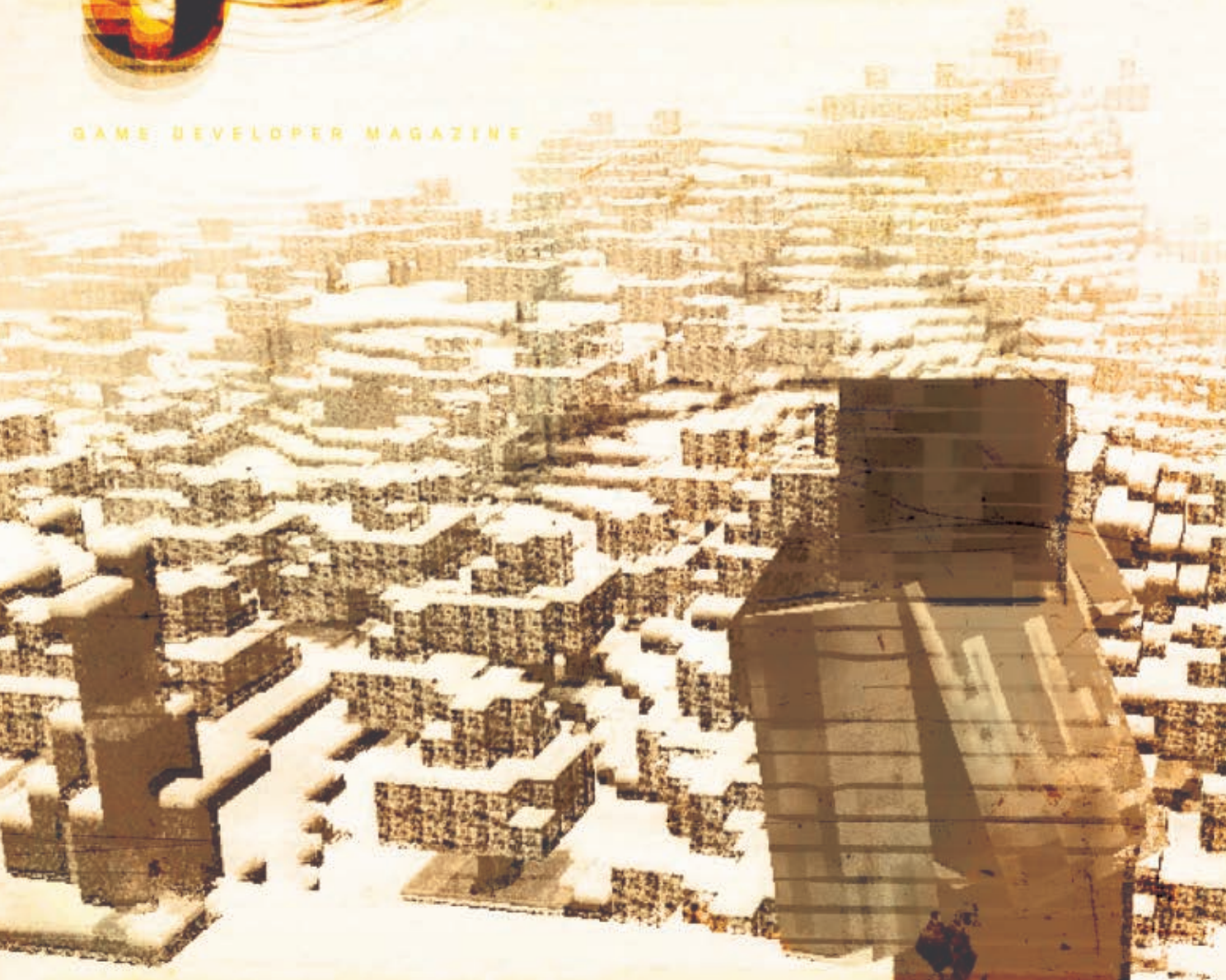
2011



MINECRAFT

THE LEADING GAME INDUSTRY MAGAZINE VOL 10 NO 2
FEBRUARY 2011 INSIDE: SHANK POSTMORTEM &
TOUGH LOVE: ENGAGING PLAYERS WITH CHALLENGING
GAMEPLAY

GAME DEVELOPER MAGAZINE



M I N E C R A F T
THE LEADING GAME INDUSTRY MAGAZINE
FEBRUARY 2011 INSIDE: SHANK POSTMORTEM &
TOUGH LOVE: ENGAGING PLAYERS WITH CHALLENGING
GAMEPLAY

Perforce | The *Fast* Software Configuration Management System



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

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

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

PERFORCE
SOFTWARE

Download a free copy of Perforce, no questions asked, from www.perforce.com. Free technical support is available throughout your evaluation.

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

gd

GAME DEVELOPER MAGAZINE

POSTMORTEMS

15 KLEI ENTERTAINMENT'S SHANK

When Klei Entertainment's nearly-completed online project lost its publisher support the studio was abruptly forced back to the drawing board. It was a lean time for the studio but from the set-back came the freedom of independence. Work on SHANK began almost immediately, and thanks to a mature toolset the studio was able to deliver the game after only 18 months of work.

By Jamie Cheng

24 MOJANG'S MINECRAFT

MINECRAFT is an indie dream: Self-funded, largely the result of one person's vision, and an immediate, runaway success. However, behind the scenes, MINECRAFT's development was struggle to stay ahead of ever increasing user demands, a complete code rewrite, and the studio's explosive growth from a one-man hobby to an ongoing business.

By Markus Persson

FEATURES

7 DISSECTING THE POSTMORTEM

Each month, game creators put their work on the dissecting table so that *Game Developer* readers can learn what went right and what went wrong during the project. Here, Ara Shirinian looks at the big picture and collects data from past postmortems to identify the common issues affecting the game development process.

By Ara Shirinian

33 PRESSED BY THE DARK

The Two-Factor Theory of Emotion describes how emotional states can be modulated by physiological changes. Stressful situations increase engagement and can give rise to often contradictory emotions. It's an idea that has wide implications for game design and goes against the conventional wisdom regarding easy difficulty in games.

By Chris Pruett with illustrations by unomoraalez

DEPARTMENTS

- 2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]
Declaration Of Independence
- 4 **HEADS UP DISPLAY** [NEWS]
Best Xbox Live Indie Games of 2010 and CANABALT goes open source.
- 41 **TOOL BOX** *By Spellbound Entertainment* [REVIEW]
Trinigy's Vision Engine 8 and product news
- 47 **THE INNER PRODUCT** *By Geoff Evans* [PROGRAMMING]
Behind The Mirror
- 55 **PIXEL PUSHER** *By Steve Theodore* [ART]
Signs of Life
- 58 **DESIGN OF THE TIMES** *By Damion Schubert* [DESIGN]
Narrative and Player Agency
- 61 **EYE ON GDC** [GDC]
2011 GDC Full-Day Tutorials
- 63 **THE BUSINESS** *By David Ederly* [BUSINESS]
Embracing Risk
- 64 **GDC CAREER PAVILION** *By Mathew Kumar* [GDC]
Level Up
- 67 **GOOD JOB!** *By Brandon Sheffield* [CAREER]
Mariel Cartwright Q&A, Who Went Where, and New Studios.
- 69 **EDUCATED PLAY** *By Jeffrey Fleming* [EDUCATION]
Sash MacKinnon's CHAOS INVADERS
- 71 **AURAL FIXATION** *By Scott Lawlor* [SOUND]
Only Three Little Things
- 80 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]
Welcome To Our Operation



DECLARATION OF INDEPENDENCE

WHAT IS AN INDIE TODAY? DOES THE TERM EVEN MATTER ANYMORE?

IT FEELS AS THOUGH THE GAME INDUSTRY IS IN

constant flux these days, which is part of what makes this an exciting place to work. For a while, that change had a lot to do with growing pains. Some folks used to (or still do) lament bloating budgets, huge teams, and giant marketing budgets. We longed for the time of the bedroom programmer.

Then it started to happen—small teams like The Behemoth were making console games on their own. Tiny dev shop Introversion Software was releasing its own titles on PC. Back then, there was a lot of discussion about who was indie, who wasn't, and what indie meant. Could The Behemoth be indie while releasing games on console, using external distribution? Many said no, at the time. Indie games had to be smaller, more independent.

Later, along came Steam, the App stores, Xbox Live Arcade, PlayStation Network, and Xbox Live Indie Games. Suddenly anyone could self-publish (or essentially be "published" by the service itself). Likewise, self-publishing on browsers or Flash portals has gained widespread acceptance. Newer business models such as free to play, pay for items, or the "pay what you want" model also allow greater flexibility than ever.

INNOVATION ENVY

» This framework supports a new breed of indies, much less reliant on outside income or big publishers for survival. Small companies are also more nimble, which makes it easier for them to adapt to (or innovate) new trends. By and large, innovation is the purview of independence these days.

There isn't a massive talent bleed from the triple-A game houses to the indie market, but it has gotten to the point where the idea of the bedroom programmer isn't such a rarity. It's still exciting to see indie successes, but it's no longer a surprise. While big companies are slashing profits or trying desperately to catch up to the social space, the one-man project MINECRAFT has sold over a million copies, most at 9.95 euros each.

MINECRAFT developer Markus Persson didn't even integrate any social features. Could he have? Yes, and it might have made his game a bigger success. But he didn't need to spend ages figuring out his social strategy before he could deploy his game. Had he delayed it in the interest of assessing market conditions and building a social brand first, it's unlikely that he'd be where he is today.

WHO'S INDIE NOW?

» Those old debates about who is indie and who isn't now seem a bit odd. What's the difference between The Behemoth releasing CASTLE CRASHERS on XBLA, and Mommy's Best Games releasing SHOOT 1UP on XBLIG? CASTLE CRASHERS received support and

some QA from Microsoft, but when it comes to code issues and updates, the dev team is responsible. SHOOT 1UP was released on XBLIG with no help from anyone—but in the grander scheme, both games are essentially published by Microsoft's platform.

Who then, is indie? People often refer to John Blow and his seminal game BRAID as indie—is Blow more indie than The Behemoth because his team is smaller? Both BRAID and CASTLE CRASHERS live on the same service. Does "indie" have a staff size limit? If so, what is it? The MINECRAFT team began as one man, and is now seven, with aims to get larger. Did that team cease to be indie, as its only game now reaches beta? How about Team Meat—that's pretty much just Edmund McMillan and Tommy Refenes with a bit of help. But their game has sold like gangbusters on XBLA proper—does this tarnish their indie cred?

Some refer to the indie spirit as a defining factor. There, John Blow fits the mold. He's making unusual games for unusual people, with a spirit that often flies in the face of the traditional industry. Where does ThatGameCompany come in, then? That team comes at games from a decidedly different direction, espousing ideals of experimentation with genre and development practice. But with a three-game Sony-funded deal, is it harder to call them indie?

A ROSE BY ANY OTHER NAME...

» Our terms and designations are failing us. We distinguish between types of games in order to help categorize our jobs, describe what we like, and as badges of honor. The term "indie" is most often a drive for authenticity, which is a nebulous and subjective term. That's likely why it's so consistently debated. Perhaps it's best to define "indie" as any company that's independently funded, but that leaves a lot of triple-A developers eligible as well.

But why not? It's clear that the lines between "indies" and triple-A studios are getting very blurry. Where does one end and the other begin, if they're all operating in a very similar sphere? Facebook games are tiny, and often made by small groups of people, but I can't foresee anyone calling them indie.

Then there are the IGF awards. MINECRAFT is nominated for several awards there—but it's also nominated in the main Choice Awards competition. Is this a conflict? In the current game climate, I dare say it isn't. Nowadays, one can be indie and compete with the "big boys"—especially when the big boys are looking to you for their next idea.

The term "indie" has lost its meaning as the scope of games has expanded. Maybe we need new terms—or maybe they're now irrelevant. What's clear is that the opportunity for making games is wider than ever before, and indie or not, that can only mean good things. ☺

—Brandon Sheffield, twitter: @necrosofty



United Business Media, 600 Harrison St., 6th Fl.,
San Francisco, CA 94107
t: 415.947.6000 f: 415.947.6090

SUBSCRIPTION SERVICES

FOR INFORMATION, ORDER QUESTIONS, AND
ADDRESS CHANGES
t: 800.250.2429 f: 847.763.9606
e: gamedeveloper@halldata.com

FOR DIGITAL SUBSCRIPTION INFORMATION
www.gdmag.com/digital

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

PRODUCTION INTERN
Tom Curtis

CONTRIBUTING EDITORS
Jesse Harlin
Steve Theodore
John Graham
Dave Mark
Soren Johnson
Damion Schubert

ADVISORY BOARD
Hal Barwood Designer-at-Large
Mick West Independent
Brad Bulkeley Neversoft
Clinton Keith Independent
Brenda Brathwaite Lolapps
Bijan Forutanpour Sony Online Entertainment
Mark DeLoura THQ
Carey Chico Independent

ADVERTISING SALES

GLOBAL SALES DIRECTOR
Aaron Murawski e: amurawski@think-services.com
t: 415.947.6227

MEDIA ACCOUNT MANAGER
John Malik Watson e: jmwatson@think-services.com
t: 415.947.6224

GLOBAL ACCOUNT MANAGER, RECRUITMENT
Gina Gross e: ggross@think-services.com
t: 415.947.6241

GLOBAL ACCOUNT MANAGER, EDUCATION
Rafael Vallin e: rvallin@think-services.com
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER
Pete C. Scibilia e: peter.scibilia@ubm.com
t: 516-562-5134

REPRINTS

WRIGHT'S MEDIA
Ryan Pratt e: rpratt@wrightsreprints.com
t: 877.652.5295

AUDIENCE DEVELOPMENT

TYSON ASSOCIATES Elaine Tyson
e: Elaine@Tysonassociates.com
LIST RENTAL Merit Direct LLC
t: 914.368.1000

MARKETING

MARKETING COORDINATOR Nahal Agahi
e: nahal.agahi@ubm.com



Cape Town, South Africa

Visit us at **Booth #613** during GDC 2011 and receive an exclusive DVD containing jaw-dropping, high-resolution sample satellite images.

The imagery is breathtaking.



The level of realism has been known to wipe out social calendars for months.

Take visual entertainment to a higher level of reality. GeoEye puts a world of high-resolution satellite imagery at your fingertips for the kind of realism gamers are buying. You can easily integrate actual locations and cities on a global scale to create photo-realistic environments for greater engagement, depth and immersion. Give your business the edge seen in today's top-rated video games. Look to GeoEye for jaw-dropping imagery.

For more information or to receive a quote, call 303.254.2245 or email gamer@geoeye.com



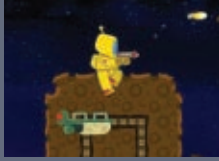
www.geoeye.com

© 2011 GeoEye. All Rights Reserved.
© 2011 Ubisoft Entertainment. All Rights Reserved. H.A.W.X.2, Tom Clancy's, the Soldier icon, and Ubisoft are trademarks of Ubisoft Entertainment in the U.S. and/or other countries.



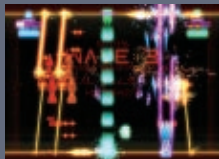
best xbox live indie games of 2010

/// THE XBOX LIVE INDIE GAMES SERVICE HAS FINALLY STARTED TO DITCH ITS NOT-SO-IMPRESSIVE IMAGE OF MESSAGE AND AVATAR GAMES, PROVIDING SOME OF THE MOST MEMORABLE GAMING EXPERIENCES OF 2010. WITH THE XBLIG COMMUNITY EVEN RUNNING ITS OWN WINTER UPRIISING PROMOTION, THE SERVICE IS REALLY COMING INTO ITS OWN, AND CAN ONLY GET BETTER FROM HERE ON IN. HERE ARE OUR PICKS FOR THE 10 BEST XBOX LIVE INDIE GAMES OF 2010. —MIKE ROSE



ASTROMAN
StarQuail Games

+ Astroman's spaceship has crash-landed on an alien planet, so naturally it's up to you to navigate Metroidvania-style levels and search for pieces of the wreckage with naught but a simple laser gun for company. This lovingly-crafted exploration platformer features plenty of puzzles to solve, aliens to blast, and planets to explore, along with a gorgeous comic visual style. As you collect pieces of the ship, Astroman can fly further out into areas unknown, discovering new levels as he goes.



**RADIANGAMES
CROSSFIRE 2**
Radiangames

+ The sequel to an already popular arena blaster on the Xbox Live Indie Games service, CROSSFIRE 2 honed the best parts of the original version to create the best Radiangames release to date. One or two players control a nimble ship that can jump from the bottom of the screen to the top and back again, allowing it to shoot enemies from behind. Along the way, there are opportunities to upgrade your ship, giving an advantage against the later, more powerful enemies.



**BREATH OF DEATH VII:
THE BEGINNING**
Zeboyd Games

+ BREATH OF DEATH VII is definitely not the seventh in a series of retro role-playing games, but rather a parody of classic RPGs of old. Taking control of a team of old school era stereotypes, players journey through an undead world with turn-based battling, character customization and plenty of exploration that will please both younger and more experienced gamers alike, although the older players may appreciate the well-written and constantly humorous dialogue a lot more!



PRISMATIC SOLID
Yoi Kimori

+ PRISMATIC SOLID is a

forced-scrolling space shooter, with the player following a set 2D plane while environmental elements move around a 3D space. Huge boss battles duck and dive around the hero, while each level sports a unique visually-impressive theme, matched with music from Namco composers Shinji Hose and Akayo Saso. While the harsh difficulty curve may put the more casual gamer off, PRISMATIC SOLID is easily one of the most gorgeous, visually stunning and intense titles on the Xbox Live Indie Games service.



**PLATFORMANCE:
CASTLE PAIN**
Magiko Gaming

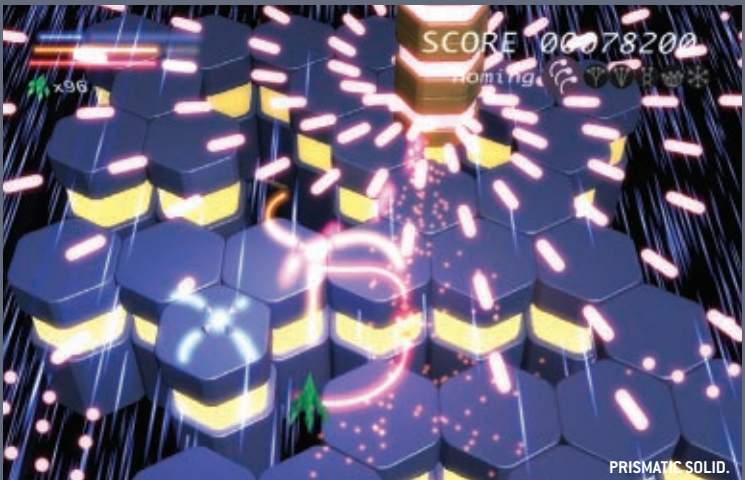
+ If you're looking for a hard-as-nails platformer to tide you over between SUPER MEAT BOY sessions, then PLATFORMANCE should be your poison. Your mission is to guide a small knight through a series of

difficult and utterly mental obstacles, with plenty of deaths guaranteed. Once you've got the hang of it, there are harder difficulty modes that introduce an instant-death ghost who follows your route, laying the pressure on as you dodge those swinging knives and deadly fireballs.



CHU'S DYNASTY
Tribetoy

+ Imagine a version of the Nintendo classic SUPER SMASH BROS in which players could mess with time, and you've got a rough idea of what CHU'S DYNASTY is all about. Up to four players battle it out over a series of platforming environments, laying into each other via STREET FIGHTER style combo attacks. On top of that, each player has their own special time-shifting abilities, allowing you to confuse your opponents and completely alter the flow of battle.



PRISMATIC SOLID.



EPIC DUNGEON

Eyehook Games

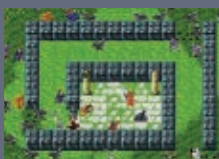
+ There's never a moment to blink in this fast-moving roguelike RPG, with enemies constantly attacking as you make your way down through the EPIC DUNGEON. Along the way your character can level up, grab stronger weapons and armor, and upgrade his or her skills to keep the nasties at bay. Every floor is randomly generated, so no two playthroughs are the same. You can even find gravestones marking where your last character died, obtaining an item from that playthrough's inventory.



SHOOT 1UP

Mommy's Best Games

+ SHOOT 1UP is a shooter that revels in complete and utter chaos. One ship just isn't enough, so why not deploy and take control of more than thirty ships at the same time? With so much mayhem to keep watch over, players can then arrange their fleet into formation and fire off a huge Plasma Auger attack, annihilating the enemy with extreme overkill. Throw a second player in to help deal with some of the strangest backdrops and enemies you'll ever see, and watch the armada double.



SOULCASTER 2

MagicalTimeBean

+ As fantasy dungeon crawlers go, SOULCASTER 2 is definitely one of the more unique offerings you'll come across. Players take control of a mage who cannot attack enemies directly, but must instead call on mystical guardians to act as both his attack and defense, like a cross between an RPG and a tower defense game. Each guardian has its own strengths and weaknesses, so placing them down in the optimal formation is a must, especially against the huge and powerful hordes.



PROTECT ME KNIGHT

Ancient

+ Created by video game music composer Yuzo Koshiro's own development studio Ancient, retro-themed PROTECT ME KNIGHT is a cross between RAMPART and an action RPG. Up to four players work together to keep a princess safe, as hordes of ruthless enemies attempt to take her prisoner. Barricades can be put in place to hold them back, while magic points are used to upgrade your stronghold and level up your character's abilities for the more challenging levels.

canabalt goes open source

SEMI SECRET SOFTWARE commemorated a \$25,000 iPhone indie game charity sale by making its popular iOS "auto-runner" game CANABALT open source.

Studio co-founder Adam Saltsman said in a blog post that the CANABALT source is now open, including the game's engine and the Flixel framework.

The game released in 2009, and according to Saltsman, has sold 225,000 copies to date. Saltsman created a prototype for the game in five days, while the studio's Eric Johnson ported it to iOS in 10 days.

Saltsman offered a disclaimer for those who plan to use the code: "We wanted to offer our condolences to everyone who downloads this and goes poking around in there," he wrote. "This



was a rushed Flash game, ported, in a rush, to the iPhone, before iPads or iPhone 4s even existed."

"We try very hard to stay up to date and do good work, but we're just two dudes—it's possible if not likely that some of the way we do things is not ideal or optimal," he added.

The indie developer plans on supporting the code as time goes on. The current code that's available provides 60 frames per

second performance and iPad and iPhone 4 Retina display support.

Saltsman clarified that while the source is open for CANABALT and users can copy-paste engine code and sell games based on the code, developers cannot redistribute CANABALT's specific game code, art or sounds. "Engine stuff is ok to distribute, CANABALT-specific stuff is not," he said. —Kris Graft

game tie-ins boost local tourism

A NUMBER OF SMALL rural Japanese towns have struck up partnership deals with video game publishers, hosting events for games that have some relevance to the area in order to attract gamers and boost local tourism.

Japanese newspaper The Mainichi Daily News reported that Capcom chose the Shibu hot spa resort in Yamanouchi, Nagano Prefecture, as the location for a MONSTER HUNTER 3 event thanks to its similarities to the fictional town found within the game.

The event attracted a large number of attendees

including Noriko Hasegawa, a 26-year-old video game fan from Kobe, who said to the paper: "I may have stayed out of this town had it not been for the event."

The MONSTER HUNTER 3 event is just the latest in a slew of similar themed events held at locations around the country seen to be relevant to a particular video game.


Last summer, Konami struck a deal with a hotel in the Atami hot spa resort in Shizuoka Prefecture to promote the company's virtual dating game LOVE PLUS while Yunin Co. and the Hokkaido town of Yuni

promote their specialty crops there through a farming video game.

Meanwhile, The Shakunagenomori botanical park in Mimata, Miyazaki Prefecture, has seen huge spikes in attendance since striking a cross-promotional deal with Colopl Inc.'s game COLONY NA SEIKATSU PLUS.

Players of the game who visit the park and buy its products can receive special cards only available at the location. Park head Yoshinori Ikebe said he has been "amazed" to see the enthusiasm for the game lead to strong sales.

—Simon Parkin



Esc... to a place where graphics come from life.

Newfoundland and Labrador. A place where you can restart, refresh and, of course, recharge. It's where inspiration was born and a **creative and unique culture** reside. And it's not far away – just off the east coast of Canada, bridging the gap between North America and Europe. Game developers are already discovering that creativity lives here, and that our workforce is highly productive and loyal. Every year our education facilities release top talent from highly recognized programs such as **Digital Animation, Computer Science, Software Engineering and IT.** Run the numbers and you'll find that Newfoundland and Labrador is a leader for economic growth in Canada. And, that operating a business here can help you leave the competition behind. **We're ready to start whenever you are.**



Department of Business Government of Newfoundland and Labrador, Canada
877.727.6353 / business@gov.nl.ca / www.nlbusiness.ca

it's happening here.

lessons learned
from two
years of game
development
self-reportage

dissecting the postmortem

A R A S H I R I N I A N

AFTER HAVING PARTICIPATED IN NUMEROUS GAME PROJECTS throughout my career as a designer, including many failures and successes, I noticed that certain types of development mistakes appeared to recur with surprising frequency. Was this just another *Twilight Zone*-inspired idiosyncrasy of my career? Or is there something more going on here? I imagined a development hell where throngs of teams all ran into the same pitfalls over and over, without knowledge of what they were doing wrong, and without the realization that anyone else might be making similar mistakes.

As developers, we have our own career histories to depend on for knowledge and experience about the rights and wrongs in game development. Beyond that, we can only rely on other developers' willingness to relate their own experiences, mistakes, and solutions to us. To that end, things like postmortems, conferences, and just plain open discussion amongst developers are great tools. But these are only vignettes in a sense, and often highly contextually dependent. For example a team's troubles with Lua integration are not helpful if you never use Lua. Beyond that, I was curious whether there was a bigger picture, what it looked like, and if the results could help us learn something new about game development in general. That, in a nutshell, is the motivation behind this analysis. >>>>

INCLUDED POSTMORTEMS:

AGE OF BOOTY, AION, AKRASIA, BRÜTAL LEGEND, THE CONDUIT, DARKSIDERS, DEADLY CREATURES, FAR CRY 2, FINAL FANTASY CRYSTAL CHRONICLES: MY LIFE AS A KING, FREE REALMS, GOLDEN AXE: BEAST RIDER, INFAMOUS, LITTLE BIG PLANET, THE MAW, N+, PENNY ARCADE ADVENTURES: ON THE RAIN-SLICK PRECIPICE OF DARKNESS, SAINTS ROW 2, SCRIBBLENAUTS, TALES OF MONKEY ISLAND, TOMB RAIDER: UNDERWORLD, TRIALS HD, UNCHARTED 2: AMONG THIEVES, WIZARD 101, THE WORLD ENDS WITH YOU



collecting data about game development

Without any expectations about whether I would find any interesting trends or commonalities across game development projects (or whether I'd even find anything coherent in the first place), I decided that *Game Developer's* extensive history of postmortems was the best and most consistent source of data.

Conveniently, postmortems written for *Game Developer* all share a similar structure: the developer writing the postmortem is required to select and illustrate five things that went right during the project, and five things that went wrong. Despite the enormous range of project types and sizes, this made it relatively straightforward to collect and organize data about the good and bad things that were reported about projects.

The data set for this analysis consists of 24 successive postmortems published in *Game Developer*, covering a period of two years, from articles that were published from February 2008 to January 2010.

collecting data about common issues

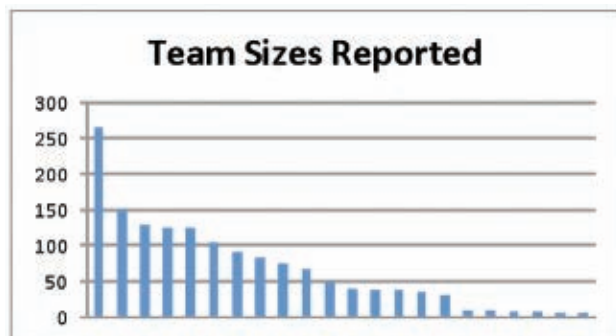
Before data collection began, I defined a set of various issues or situations that I thought would be interesting to track across all postmortems. For example, whether the postmortem mentioned using scrum or some agile process, whether a successful experience with outsourcing was reported, and so on. For each of these items, each postmortem was scored for mention of that issue or situation. I didn't just search for specific words; collecting the data required a lot of extensive re-reading to ensure that if the postmortem was counted in a certain category, there would be no question about its inclusion.

results part 1: postmortem metrics

Here are some metrics that characterize the projects whose postmortems were included in this analysis.

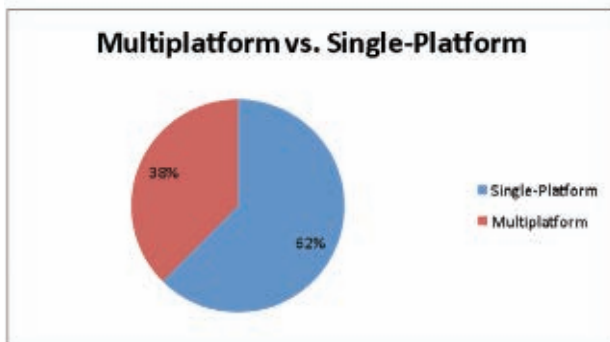
team sizes reported

The largest project was FAR CRY 2, boasting a team of 265 members. The smallest projects were AGE OF BOOTY and N+, both reporting just 5 team members. Four postmortems reported a range of team size; of those, the maximum was recorded. Two postmortems did not report a team size and were excluded from this section. Team sizes tended to fall into four clumps, with the smallest teams consisting of 5–10 members. The next five largest teams consisted of 30–40 members. There was another clump representing 68–90 members, and six projects reported over 100 team members.



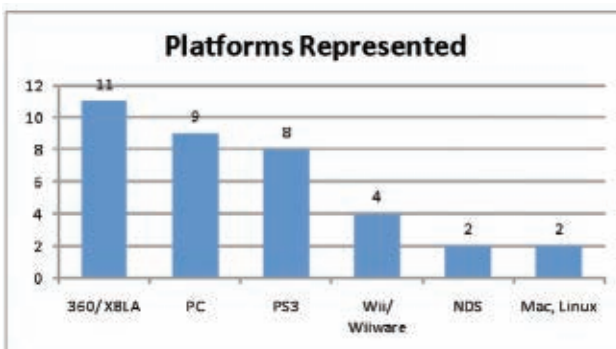
multiplatform vs. single-platform

Slightly more than 1/3 of the projects were released on multiple platforms.



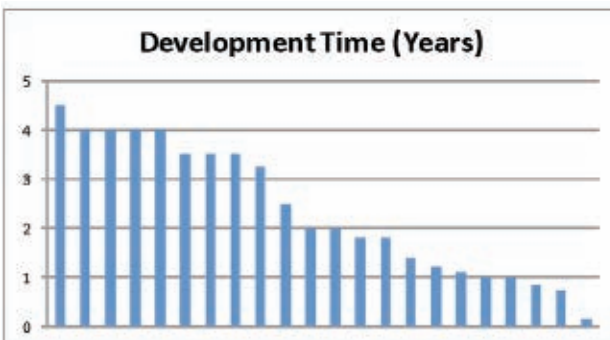
platforms represented

Multiplatform projects are counted once for each platform they were released on (there were no PSN games represented in this selection of postmortems).



development time

The average development time for projects was 2.4 years. Two projects did not report a development time and were excluded from this section. Interestingly, development time also seems to clump at yearly or half-year marks, although this is probably an artifact of how durations were reported. There is also a large jump between the 2-year mark and the 3.5-year mark.



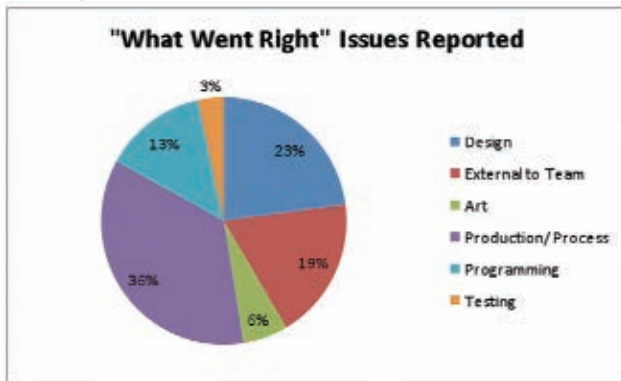
use of contractors or outsourcing

✓ Eleven of 24 postmortems reported utilizing contractors in the project. Only 7, or about 29 percent, reported outsourcing work to separate companies.

results part 2: how things went right and wrong

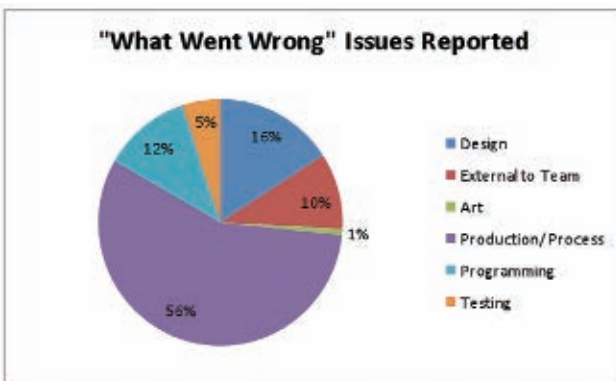
As described in the Methodology section (Pg. 11), each “thing that went right” and “thing that went wrong” that was reported in each postmortem was classified into one of six categories. Each of these categories generally corresponds to a major discipline involved in game development. In total, this supplied 240 data points: 120 “rights” and 120 “wrongs.”

If we look at the all of “right” issues under this classification, we see some interesting results:



What's going on in this pie chart? When things go well, production is most often credited for it, with design running second place and issues external to the team a close third. Art issues appear to take a disproportionately small slice of responsibility for the good things that happen on projects, and testing gets the least amount of representation.

Now, let's take a look at how issues were classified when things went wrong.



Immediately we see that production issues take an enormous share of the problems when things go wrong. What this means is that when things go wrong on a project, most often it's not inherently because of design

cavents

It's important to mention some qualifications that are inherent in the nature of postmortems intended for public consumption. The first thing that's called into question is the sincerity of the report itself. Is the writer deliberately holding back certain information that would be considered “bad PR” for the company to reveal? Some writers appear to be more forthright about dramatically bad outcomes of their projects than others. It seems safe to say that there is likely some amount of this type of informational restraint, but of course we can never really know who is writing with candor and who has one or more fingers tied behind their backs for whatever reason. Fortunately, I have been pleasantly surprised about the willingness of many authors to expound on some quite disastrous situations in their postmortems.

The second major qualification is one of completeness of information. Because postmortem authors are free to pick their favorite five “bad things” and “good things” about their projects, and because each postmortem is written independently, there is no guarantee that any given topic will be mentioned at all in any given postmortem. What's more, the absence of mention of any given topic gives us no information about that topic one way or the other. For example, if there is no mention of “working crunch” in any way, we still don't know if the team worked crunch or not. Either way, if the team did work crunch and it wasn't mentioned, we wouldn't even know if it was a brief finish-line crunch with everyone cheering to complete their most polished work, or a morale-obliterating death march with far-reaching casualties.

In short, we're at the mercy of the collective of postmortem authors, as the quality of information presented here is a function of their ability to present honest and complete information about their own projects.

Reliable Pathfinding Technology. Havok AI.



Physics Animation Behavior Cloth Destruction AI

havok[™]
www.havok.com

*Turning Creative Aspirations
into Technical Realities[™]*

methodology and definitions

Data from the 24 postmortems was collected and organized in two different ways. These were the methods undertaken for each.

organizing and summarizing things that went right and wrong

✓ The body of each postmortem consists of five expositions about things that went right and five about things that went wrong. Each of these atomic “things” was then classified into one of the following seven categories, defined as follows:

design Relating to game design, level design, gameplay and rule designs, and overall game vision.

external to team This category covers all situations and decisions that were made that are clearly external to the direct team and development process, including business logistics, hiring, partnerships, funding, marketing, studio-wide decisions, and so on.

art Relating to art decisions, direction, or specific art processes.

production/ process This relates to scheduling, work prioritization, production methodologies, development plans and processes, scope, team morale, team communication, team assignment, team management, and so on.

programming This category covers all technical issues, including tools, technology implementation, and anything code-related.

testing This category includes

all traditional QA functions, including bug testing, gameplay/usability testing, localization, gameplay data collection, and metrics.

other There were only two items that did not fit any of the above categories, and were omitted from the results. Together they represented less than 0.5 percent of the data overall. One was a “right” item about sound direction (interestingly, this was the only item that could be categorized as relating to audio).

The other was a “right” about obtaining nice office space.

✓ A couple of specific types of challenges were encountered in categorizing each reported item from the postmortems. First, sometimes the original title of the item was not a good characterization of what was explained. This issue was avoided by categorizing the item based on the content of the explanations, not on how the explanations were titled. Second,

sometimes the discussion on a certain item would have crossover with or would otherwise veer into other subjects. For example, an item ultimately classified as “design” might make mention of some production process or something about art, or vice versa. In these cases, a good faith effort was made to select a category based on what the primary gist of the entire explanation appeared to be about.

mistakes, art mistakes, or coding mistakes, but problems in the way the process of development is prioritized, conducted, and managed.

In general, this seems to indicate that development teams are just much worse at planning, coordinating, and conducting the work required to produce a game as a whole, more than anything else. I think this finding will resonate with a lot of developers—poorly managed projects unfortunately appear to be much more common than well-managed ones. Mind you, our sample of postmortems was all games that shipped, and at least by that account can be considered successfully managed projects.

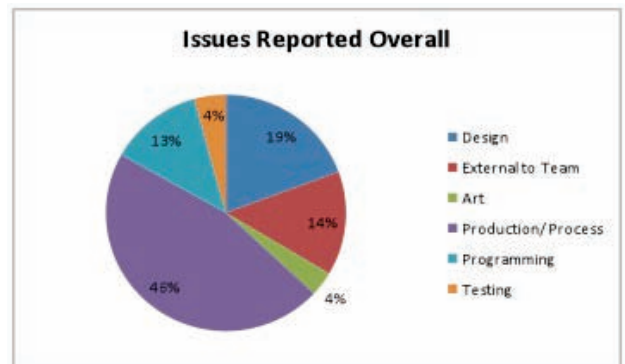
There were a total of 68 individual “what went wrong” issues classified under the “production” category. Within this group, the most common issues were related to scope, feature creep, and resource problems; this accounted for 16, or about 23 percent of production problems. The second-largest subset of problems within production was team-communication related, which accounted for 8, or slightly more than 10 percent of the issues. Six issues involved various critical events happening later than they should have in the development cycle.

While production increased dramatically in the “wrongs” versus the “rights,” all other issue types decreased in frequency of reports (except for testing, which nearly doubled, but was still a small number).

It should be noted that the “wrong” issues related to testing were all about planning, managing, and being unprepared for the logistical burdens of the testing process, and not related to testers doing a bad job. For these reasons, most of the testing-classified items could arguably also be included in the production category instead.

Another interesting statistic is that programming issues decreased only by 1 percent, while design, art, and external categories decreased by 7 percent, 5 percent and 9 percent respectively. This also seems to make sense, as when things do go wrong, technical issues can easily have much more salient effects on the health of a project. However, design issues are still the second most represented type, which also seems to indicate that designers may be creating more problems for projects than the other traditional disciplines.

Here’s what the distribution of issues looks like if we combine all the rights and wrongs together:





This aggregate graph might be best understood as the amount of overall impact or significance a certain discipline can make on the smooth-running of a project's development. As expected, production is almost half of the pie, with design taking the number two spot again, and coding taking roughly equal weight to external issues. Note that since postmortems are explicitly about the process of game creation, this is not an indication of what is most important to making a game successful or artistic—just to completing it.

results part 3: common themes

are small teams important?

✓ Seven, or just under a third of postmortems, reported a “small team” as a positive aspect of the project. Interestingly, three of those projects actually consisted of 30–40 member teams, with the other four being reported from the smallest teams.

developer-publisher relationships

✓ Fifteen out of the 24 projects were developed under a typical developer-publisher system, where the entity publishing the end product is not the same as the entity developing it. Projects that were developed at a separate studio entity, even if owned by the publisher, were included in the count of 15.

Of those postmortems, we found that 40 percent, or 6 out of 15, reported a positive developer-publisher relationship. Three out of 15, or 20 percent, reported problems with their relationship with the publisher, while the remaining 40 percent did not report anything good or bad about their publisher.

planning for the team

✓ Five out of 24, or 21 percent, reported deliberately planning their game with respect to the capability or expertise of the team. All of the projects that were in this category completed their development in under 2 years, and the majority of development cycles were completed in a year or less. These teams were not all “small,” however—two of the projects had 37- and 50-person teams respectively.

crunch, time extensions, and scope

✓ Nine projects, or 38 percent, reported receiving a time extension to finish their project. A few also reported the length of the time extension, which was anywhere from “a few extra weeks” to a whopping total extension of 17 months (BRUTAL LEGEND).

The same number also reported some manner of a crunch period during their project, although few actually reported the duration of the crunch, which varied anywhere from six months to “almost a full year” to “always in crunch mode” (MY LIFE AS A KING).

Most interestingly, 17 postmortems, or 71 percent, reported scope problems where there was either not enough time or resources to complete the game, or there was too much design that had to be cut, often repeatedly throughout the project. This is the most obvious trends across the postmortems: teams are consistently underestimating the required amount of time and resources needed to create their titles. However, it's unclear how many of these scope problems are due to pressure to complete the game under a convenient timeframe for the publisher, and how many are the result of poor estimating from the developer.

Along similar lines, half of all projects reported making last-minute or exceptionally late feature additions or changes.

development agility

✓ Five projects, or 21 percent, reported using some flavor of scrum or other explicitly agile development process.

Nine projects reported using a deliberately flexible design approach to at least one of the game elements they described.

Eighteen, or 75 percent of projects, reported iteration or rapid prototyping as a valuable component to development.

Conversely, a surprising 29 percent, (7 postmortems) actually reported that they committed to an inflexible design or plan partway through development. Five of those projects were 2.5 year-long cycles or longer.

management and communication

✓ Eleven projects, or roughly half, made mention of some variety of team management problems, which included problems like overwork (separate of crunch mentions), lack of focus, problems with staffing, and morale issues.

Nine projects, or 38 percent, reported some kind of significant problem with communication across teammates, which included deliberate refusal of team members to communicate with each other, confusion about game vision and direction, and the ineffectiveness of leaders to adequately convey changes to the rest of the team.

pipeline problems

✓ Nine postmortems also mentioned explicit problems with asset pipelines. These include pipelines making work unusually time-consuming or painful, not coming online early enough in the project, or otherwise not adequately supporting the actual work process of team members.

so how did the outsourcing go?

✓ Of the seven postmortems that reported outsourcing some of their work, three reported an overall successful experience, and four reported problems of some kind. The variety of problems included a lack of preparation, starting the process too early with respect to the overall cycle, underestimating the amount of management involved, and hiring a company to perform work that they did not have sufficient expertise in.

conclusions

✓ From looking at postmortems over the past two years and considering the data presented here, it would seem that the biggest takeaway is the importance of managing the development process itself. Of course, we're not talking about just the “existence” of management, but good, methodical, careful management that keeps the interest of the game and the team at the forefront. Unlike some of the other disciplines, project and team management, when done well, may be able to compensate for inadequacies in other areas. However, when executed poorly, it seems as though project management has the potential to unravel and destroy even the best creative work.

Beyond that, one of the most common and disturbing trends is the inability for game development projects to be properly scoped and scheduled. We developers are constantly fighting a battle (that we aren't winning as often as we should be) between the resources we have available and the end product we are attempting to realize—whether those factors are imposed by ourselves or by those paying our checks. 📌

ARA SHIRINIAN is a game designer and writer. Notable games he has worked on include THE RED STAR, NIGHTSKY, and DODD'S BIG ADVENTURE for the uDraw Wii tablet. He can be reached at www.shirinian.net.

UNREAL TECHNOLOGY NEWS

MOBILE EDITION



UNREAL ENGINE 3 FOR MOBILE: INFINITY BLADE KICKS OFF THE PARTY

Anyone looking for proof that Unreal Engine 3 is ready for the world of mobile game development got their answer in December 2010. Epic and ChAIR Entertainment's own *Infinity Blade* created a major stir the moment it was released for the iPhone, iPad and iPod touch for a simple reason: it showed that mobile gaming is ready for console-quality graphics and gameplay.

"We were able to make *Infinity Blade* in five months using a team of 10 developers," said ChAIR Entertainment Creative Director Donald Mustard. "The advantage of using Unreal Engine 3 to make a game like this is that it saves you years of work – you spend your development time creating a fun game as opposed to creating underlying technology."

Upon release, *Infinity Blade* shot to the top of the iTunes App Store charts in countries around the world. IGN named *Infinity Blade* the best iPhone game of the year. It won TouchGen's Best Action Game, Best Graphics and Game of the Year honors along with many other awards. Fast Company has named *Infinity Blade* among the seven most innovative iPad Apps of 2010 and many sites reported *Infinity Blade* to be the "fastest grossing app ever". The game's first free update has already shipped adding in-app purchases to the mix, and more features, including multi-player, which is on the way.

The floodgates have been opened for a new phase of game development for mobile devices. Not only has *Infinity Blade* shown that mobile devices can support high-fidelity graphics, it's also proved that there's a large market of mobile users whose trigger fingers have been itching for the kind of quality mobile games made possible for the first time with Unreal Engine 3.

Dungeon Defenders: The First Wave

Infinity Blade isn't the only UE3-powered mobile game making a splash. Trendy Entertainment's *Dungeon Defenders: The First Wave* is the first UDK-powered game to debut on Android as well as the iOS devices.

"Developing mobile games with Unreal technology allowed our small team, with no prior experience on mobile platforms, to produce several titles in just a handful of months – just in time for the holiday season," said Trendy Entertainment Development Director Jeremy Stieglitz.

"Unreal's combination of robust tools, seamless art-driven pipelines, and powerful graphics has given our team the power to create high-end games which clearly stand out from the crowd."

And that's just the beginning. Plenty of developers are using the Unreal Development Kit (UDK) for free to test the waters of mobile gaming, now that it's clear they won't have to compromise their vision or creativity to go mobile.

"UDK for mobile development brings the power of Unreal Engine 3 to virtually anyone, allowing them to realize their own game ideas, and make them come to life on their own mobile devices," said Epic's Shane Caudle. "Now with UDK, anyone can be a game developer and sell their games on Apple's highly successful App store."

A "Jazzy" tutorial: Using UDK for iOS

Caudle created a tutorial to demonstrate how to use UDK to prototype an iOS game using the visual scripting language Unreal Kismet without touching code. If nothing else, it's worth watching just to see *Jazz Jackrabbit* in action again – Jazz starred in one of the first games Epic ever made, back in 1994. In the tutorial, the game is designed to be a top-down dual-stick shooter.

"This Jazz tutorial is just a simple example of how quickly you can get something fun up and running without having to write any code, due to the power of Kismet," said Caudle. "It's amazing how quickly you can iterate on your game using Kismet, the Mobile Previewer, and our UDK Remote app. UDK Remote is freely available on the App Store, and allows you to test your game on your computer while using your iPhone, iPad or iPod touch as a remote input device. If you're into mobile game development, I suggest giving UDK a shot!"

After getting Jazz armed and running, Shane takes the game further by adding multiple enemy types, pickups and powerups, particle effects, music and sound effects, HUD elements for health, and score and custom input zones.

It's GDC time!

If you're interested in using Unreal Engine 3, let's talk at GDC this year. Whether you're making a game for console, PC or mobile, you should see our latest tools and technologies in action. Contact us at licensing@epicgames.com with several day and time combinations that suit your schedule to arrange a meeting.



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

Epic's Unreal Engine 3 has won Game Developer magazine's Best Engine Front Line Award four times along with entry into the Hall of Fame. UE3 has won three consecutive Develop Industry Excellence Awards.

Epic is the creator of mega-hit "Unreal" series of games and the blockbuster "Gears of War" franchise.

Follow @MarkRein on Twitter.

UPCOMING EPIC ATTENDED EVENTS

DICE Summit
Las Vegas
Feb. 9-11, 2011

GDC
San Francisco
Feb. 28-Mar. 4, 2011

Please email: mrein@epicgames.com for appointments.

WWW.EPICGAMES.COM





Taking

Game Development

to the next level



DevTest Studio

The industry's #1 choice for test management and defect tracking

DevTrack

Use DevTrack to track defects/issues

- Track each issue through a definable workflow
- SCM integration-track fixes against their source code deliverables
- Deploy a resolution across multiple releases, versions and products
- Reporting and metrics to illustrate the entire defect lifecycle

DevTest

Use DevTest to manage your testing

- Create a central repository for your test cases, Knowledge items and automation scripts
- Schedule releases and test cycles using a wizard-driven interface
- Execute test assignments and submit defects from the same interface
- Track results with real-time dashboards and reports

TestLink

Use TestLink to automate your testing

- Add automated tests to the DevTest library
- Schedule automated tests along with manual tests
- Launch automated tests from the DevTest interface
- Track automation results with real-time dashboards and reports

Try DevTrack and DevTest live. Watch a recorded overview. Request an online demo.

www.techexcel.com
1.800.439.7782



postmortem

K R E A T I V E

It was January 2009. We had just spent over a year working on a free-to-play, four-player brawler titled *Sugar Rush*, and the game was scheduled to go to public beta in less than two weeks. After a couple stress tests, and running hundreds of fake clients on the Amazon EC2 virtual cloud environment, we were fairly confident that our architecture would hold up, and the game was getting decent numbers in attachment. Then the whole thing came crumbling down. Our publisher, Nexon Publishing North America, abruptly announced that it was shutting down, scuttling our game with it. It hardly seemed like it at the time, but this turned out to be the best thing that ever happened to us.

In retrospect, our studio had become complacent with the steady cashflow, and although I still think back fondly to the game and believe we executed at a high quality, it was clear the project was losing its independent identity.

The meltdown forced us to re-examine our studio goals. Eventually, we decided to forge on and create our own game, exactly the way we wanted to create it. Through that journey, we created an unforgettable experience and rediscovered our identity as an independent studio.

>>>>

shank



PUBLISHER EA Partners
DEVELOPER Klei Entertainment
NUMBER OF DEVELOPERS
Average of about 10
LENGTH OF DEVELOPMENT 18 months
RELEASE DATES August 24, 2010 (PSN), August 25, 2010 (XBLA), October 26, 2010 (PC)
BUDGET More than EETIS, less than SUGAR RUSH
SOFTWARE Adobe Flash, Adobe Photoshop, FMOD Designer, Visual Studio 2008, Hudon CI, The Shanker, SVN, Python, and Scite, among others
PLATFORM Xbox LIVE Arcade, PlayStation Network, PC

WHAT WENT RIGHT

1) PAX 2009. Exhibiting at PAX and announcing the game with a playable demo was one of the best decisions we made during development. It forced us to polish our game and really nail down what the core concepts were, and gave us far more publicity than we thought possible. Honestly, it was unclear to us what the reception would be; of course, we loved the game and felt others would enjoy it, but the actual response far exceeded our expectations.

Because we didn't yet have a publisher and the game wasn't approved for any console, we hid the consoles under question-mark boxes. I remember a particularly interesting conversation with a potential publisher, who requested and cautioned us not to show a playable demo at the show for fear that it would impair their ability to market our game later. However, in my experience, publishers tend to market your game exactly as much as it's already gained traction, so we felt strongly about getting out early and often.

Thus, throwing caution to the wind, we showed our entire hand, and with the magical help

of Carolyn Carnes (our wonderful PR representative who has moved on to become Digital Marketing Manager at Microsoft), we created a dialogue with the press and a solid following within the game community. This gave us the momentum to continue building the product and ultimately sign on with EA Partners.

2) TOOLS AND PIPELINE. One of the benefits of being independent, of course, is that you can develop with a constant focus in the long term. Without impending milestone deliverables, it's okay to have months at a time where nothing seems to be getting done while the tools and engine are being created. While the back-end engine was being built, we also lovingly created three essential tools for content creators:

→ The Shanker—our level creation tool, to allow level designers and level artists to create levels and script the experience.

→ A custom animation pipeline from Adobe Flash to allow artists to work naturally in Flash while still pushing the

boundaries of what can be rendered on screen.

→ A stategraph engine for game object behavior, to allow designers to create enemy behaviors, interactive objects, and SHANK controls.

In the end, we built 13 levels in three months using these tools (this shows up in the "what went wrong" section), and the fact that our programming team worked a total of possibly three overtime weekends throughout the entire project is a testament to how robust our tools are.

3) AUDIO INTEGRATION. The audio component of SHANK was one of the most painless integrations we've ever encountered. It was so painless that I almost didn't want to include it in this section, as most of the staff never had to care about it, but that's also exactly why it was something that went right. From a technical perspective, we spent a couple days integrating FMOD, and from then on had only very small issues all the way up until launch.

When it came to music, we auditioned the role to a number of different composers, but none of them seemed to really fit our

theme. Then Vince de Vera and Jason Garner—two local Vancouver musicians—gave us a sample tune, and we knew immediately that they were perfect for the role. Neither had worked on games before, and our collaboration was incredibly refreshing. Music transitions were done using the Music Designer in FMOD, transitioning between segments at key points in the gameplay.

We also worked with a local sound designer to help with some of our integration, including leveling the sounds. On one eventful day, Alan suggested that I add reverb and ambience to different sections of the levels, and within a day our entire world felt full and wonderful as SHANK stepped in and out of different environments. For a team with no expertise, the audio was a fun and painless ride, and with the experience we gained from SHANK, we're confident we can create an even better aural experience for our next outing.

4) A CLEAR VISION. From the beginning, it was clear what SHANK was meant to be. We set out to revive the 2D beat-em-up experience, inspired by the Rodriguez-created *Desperado*,



Get picked first.





GAIKAI™

www.gaikai.com/jobs

shank

post mortem



among other modern Westerns. We dubbed it a “cinematic brawler,” and within a month of development, the art style, aesthetics, and feel of the controls were set. Being an independent project, there was no self-censorship, no political meetings, and no second guesses. This clarity allowed us to focus all our efforts on fine-tuning that experience, and to not waste our efforts trying to find the “hook.” We knew what it was—now we just had to execute it.

This is in sharp contrast to EETS. In 2003, I built a small physics engine in my free time, and then posed the question “Okay, how do we build an actual game out of this?” This is not necessarily a poor way to build a game—the experimentation was extremely fun—but it did lead to a very long development cycle (three years!) before we understood why anyone would play the game.

The last time I worked on a game with as much clarity as SHANK was WARHAMMER 40,000: DAWN OF WAR. Jay Wilson led that project as lead designer; there was never any confusion as to what the product was, and the entire team worked together to achieve the goal. It went on to become the most efficient, profitable, and crunch-light project Relic had shipped to date.

5) RELATIONSHIP WITH EA PARTNERS. Our relationship with EA Partners has completely changed our view on how publisher-developer relationships can work. During our negotiations with different publishers, we were adamant that we keep creative control of the project. In this case, EA Partners clearly trusted our abilities. They gave us the freedom to execute our vision, and perhaps more importantly make tough decisions—some of which would make a producer cringe. It was clear that we were in this together, and that any problems we had were not simply our own, but the entire team’s.

The clearest example of this is that our producer, Mike Doran, would answer the phone saying, “How can I help?” and after hearing our current blockers, would respond by saying, “On it.” EA Partners actually

felt like a resource to our studio rather than overhead that needed to be managed. Obviously, we worked hard to earn this relationship, and in the end, both parties came out incredibly satisfied with the whole experience.

When a budding independent developer asks whether they should strive to obtain a publisher, the answer is usually “It depends.” In this case, due to the size of the project, we needed additional financing, but we were able to find a partner that enhanced the entire process. I believe this relationship proved to us that it’s possible to stay creatively independent while working with a large publisher.

WHAT WENT WRONG

1) SCHEDULE AND ART LOCK DOWN.

One of the biggest issues we had was the late stage at which the art was locked down. Although we had great tools, many of the levels were done in parallel and the final art was delivered all at once, scant weeks before we went into certification. In addition, cutscenes were delivered right up to the final days, and UI artwork was also relegated to the end.

This caused no end of cascading headaches; in particular, programmers had to deal with content blowing our budgets at the last moment. The late FX we implemented used features that were not fully tested, and days were spent debugging and optimizing tools that should have been used months earlier. The UI became a huge headache due to synchronous loading of textures, and the cutscenes deadline for the ratings board tested the endurance of the art team.

The designers got the worst of the problems. It was their job to polish the levels after the final content was delivered—making sure the cameras were framed, the enemies were in place, the cutscenes transitioned properly, and so on. As the content slammed in, suddenly the designers had a Herculean amount of work. In the end, the team pulled through because of the strength of our tools, and because we worked our butts off.

2) THE PLAYER LEARNING

EXPERIENCE. For the two PAX demos we created, we tailored them specifically for manned demos, where we could guide the users with very simple instructions on how to play. Since we were guaranteed that we would be next to them, we didn't worry as much about the learning curve as the overall experience.

This was a valid strategy for the exhibitions, and worked incredibly well, but turned out to be a downfall for the final game. As the primary demo man, I quickly learned that only very few key tips needed to be mentioned for a player to really understand the combat mechanics—after all, responsive combat was something we're passionate about. Letting them know that weapon transitions are key, for example, or that pressing grapple opens up a whole new moveset was all we needed to get players to "grok" the system and have a lot of fun.

What we didn't realize was how hard it would be to translate that into an in-game tutorial. Those

simple phrases, precisely placed just as I could feel a player's need to hear it, were nearly impossible to implement, especially as the tutorial was done after the level was already art-complete. What we needed was a more thoughtful approach to the whole learning experience, and the tutorial is only a small piece of that equation.

Given additional time, greater full-experience playthrough testing would have been immensely useful in finding out common blockers, pacing problems, and control frustrations in the game. For example, although we found offensive players enjoyed our control scheme immensely—often calling the combat system the most responsive they'd ever played—defensive players felt that they were sticky and unresponsive. This could have been easily resolved had we done rigorous, empirical playtesting with a wider range of players earlier in the process.

Sadly, this seems to be a common theme in postmortems, and it's something we're extremely mindful of for our future titles.



3) MULTIPLE PROJECTS WITHOUT THE RESOURCES.

For the majority of the time we were developing SHANK, we were also actively trying to build another project. However, we simply didn't have the resources to do so, and the result was that the staff was pulled in multiple directions.

I had probably one of the worst months in my professional life in my attempts to have both projects live up to the quality that we were proud of. For weeks we pulled some unspeakable hours in the office

switching between two projects, working with the staff to try and make them both work. Thankfully, our partner on the second project worked with us to put it on hold, allowing us to focus on delivering one great product at a time.

There are many reasons why this came to pass. Ultimately, it was a combination of incomplete information—we didn't know when or if projects would be greenlit—as well as poor resource management on our part. We also made the classic mistake of underestimating



shank



the differences between the two projects, thinking that our in-house engine and tools would support both projects. Indeed, there were definitely efficiencies to be had, and features from one very much benefited the other—but there were enough differences that significantly more effort than planned was needed to move the projects along.

4) CASH FLOW. For the first half of development, the company slowly bled the war chest we had saved up over that first four years in business. Doing contract jobs here and there, we kind of bobbed along, but ultimately spent a lot more than we earned. This caused us to really back-load our tasks, since we couldn't hire, and a lot of our staff was working on other projects that brought in some money.

At the end of 2009, running on fumes, we asked the shareholders (who are also employees) for short-term loans, temporarily reduced salaries, and Jeff and I signed an agreement to take out a loan against our personal assets to keep the studio afloat until our finances were figured out.

Many of the cons in this section were exacerbated by the lack of resources, and once the cash starting flowing in again, we had to try and catch up on all the lost time. In particular, the cutscenes were delayed to the very last moment because we simply didn't have the funds to pay for contractors to help out earlier in the project.

Thankfully, we've fully recovered to past our previous peak, and I'm looking forward to future projects benefiting from not being resource starved early in the project.

1) NO INTERNAL QA. SHANK benefited both from an outsourced QA studio as well as EA Partners' small QA staff. However, as the builds went out for testing more frequently, it became more and more obvious that we needed a better process for smoketesting builds before they left our office.

We use continuous integration in the office, and our turnaround time for an internal build was counted in minutes—fast enough that people didn't really think about it. However, when it came to doing a deploy for the QA team, we went through a multistage process of

rebuilding all three platforms, pre-processing data, running the platform package scripts, and finally packaging it all up. This takes about an hour. Because we didn't want the team to simply twiddle their thumbs and wait for an hour, often everyone would go home as the designers continued working and waiting for the build. When the build finished, we'd smoketest it, and only briefly since by then it was getting really late. If we could fix it, then we'd have to start the whole cycle again. Two things happened more often than I liked.

First, sometimes we didn't do a good enough job at smoketesting, and a broken build was delivered. I'd get a call very early in the morning to let me know that this was the case, since our test team was based on the East Coast and we're on the West Coast. Second, whoever smoketested (usually the designers) went home extremely late, causing us to be less effective the next day.

In retrospect, we needed dedicated in-house QA, a staggered delivery schedule, and preferably an external QA team in the same time zone.

CONCLUSION

/// After two years of developing a game in tandem with a publisher and their designers, it was incredibly satisfying to once again work on a project that is entirely our own. For much of the team, this is the first game they've ever shipped, and for all of us, it's been an incredible journey filled with amazing lessons.

The words in this article can't possibly capture all our learnings, but hopefully over time we can share them with everyone and can all continue to improve the medium that we care so deeply about.

The concept of SHANK happened to be conceived by Jeff Agala and I, but it was the entire team that brought it to life. I think the most exciting thing for me is to see what games we'll build together in the future. 🎮

JAMIE CHENG self-funded Klei Entertainment in 2005. Prior to founding Klei, Jamie was an avid AI programmer for Relic Entertainment. He also co-wrote an article on planning systems for *Game Programming Gems 5* and *the Best of Game Programming Gems*.



AWARD WINNING GAMES

AWARD WINNING DEVELOPER

INSOMNIAC GAMES

BIG STUFF ON THE HORIZON

www.insomniacgames.com/careers

facebook

Become a fan:
Insomniac Games

twitter

Follow us:
insomniacgames

MARKUS PERSSON

mine

I had been working as a game programmer at king.com for four and a half years, making games both at work and during my free time. When I started that job, we were eight people, and when I left, we were 10 times as many. Initially, my hobby game development wasn't a problem for the company, but once they informed me that they technically owned all games I made during my free time and any prizes I won in competitions, I quit.

I got a job at a more liberal company as a web developer, so I could focus a bit more on my hobby. Initially, MINECRAFT started out as a top-down strategy game where you moved characters around in a dynamic cube-based world, kind of like a crossover between DWARF FORTRESS and ROLLERCOASTER TYCOON.

While playing around with a first-person mode, I realized the world was much more interesting as a first-person adventure game. The low-resolution textures I had used got really blurry and awful, so I thought I had to try to get some higher resolution art. It wasn't until playing INFINIMINER that I realized I could just turn off the texture smoothing and end up with a charming pixely look, and that's how the seeds of what MINECRAFT is today were born. After talking about it on a couple of Internet forums I frequent, and putting out a free alpha version for a few weeks, I decided to try to charge for the game, and added premium accounts. Initially, these had very minor benefits over free accounts. I sold about five to ten per day. Over time, the sales increased to 15, which was enough to support me full time, so I dropped down to part time on my day job and spent three days per week on MINECRAFT.

Then it sold 30 copies per day. Then 50. I quit my day job and went full time on MINECRAFT exactly one year after leaving the job at king.com. Now it's seven months later, we just released the beta version, I've started a game studio with a couple of friends, and we've hired a few talented people.

If you've never played it, MINECRAFT is a sandbox fantasy adventure game set in a world made up entirely of one-meter blocks of different materials. The player can pick up those blocks and move them around, and use them to craft items and tools. Monsters can spawn in dark areas and during the night, which plays nicely into a general fear of the dark. During alpha, the development focus was on experimenting with features and seeing what works. In beta, we will focus more on cleaning up the game, fleshing out existing content, and improving performance and stability.

The game is currently selling over 6,000 copies per day, and we've gotten lots of awards, including *PC Gamer UK's* 2010 Game of the Year, and Machinima User's Choice Game of the Year. We've also been nominated in three categories in the Independent Games Festival. This postmortem covers the early days of MINECRAFT,

craft

alpha





specifically the period during which the game was in alpha.

WHAT WENT RIGHT

1) LETTING THE GAME FUND ITSELF. MINECRAFT's always been fully self-funded. I developed the initial versions in my free time as a hobby, and hosted it on my private servers. Once the game made enough money to pay for a server on its own, I signed up for a real server and moved everything there. My income was coming from my day job, which I kept working at part time until working on MINECRAFT full time was risk free.

It turns out I could've gone full time earlier, so perhaps I was a bit too careful, but since I had a lot of free time then, there wasn't a problem with me working on MINECRAFT during my spare time. As a result of never taking any risks and letting the game fund itself, MINECRAFT and Mojang are fully independent with no external investors, so we haven't made any promises to anyone other than the players. This means I can keep focusing on making the game I want to make, and to work with the people I want to work with.

The success of the game also meant we got hundreds of applications once we started looking for more developers, and we ended up snagging some really talented people like Jens Bergensten from Oxeye and Junkboy, an awesome pixel artist.

2) OPEN DEVELOPMENT. From the start, I was very open about MINECRAFT's development. I

talked about it on forums, primarily those on TIGSource, and told people what I was doing and where I wanted to take the game. Fairly soon, we set up an IRC channel for MINECRAFT for more rapid discussion, and after a while, I set up a Tumblr blog in order to get information out to more people more easily.

Discussing with the players and listening to suggestions, I learned a lot about how the game could be played and what directions were most interesting to others. Usually, people played it in completely different ways than I did. For example, when I added more complex game rules to the basic game engine, it turned out a lot of people really liked the free building from the engine test, so I kept it around and called it "creative mode." Sometimes players manage to convince me that something I originally thought was a bad idea actually is a great idea, like with lighting and custom texture packs. With the texture packs, players were hacking the client to replace the textures for a long time, and I resisted the change until I saw a PORTAL server mod that basically was a simple version of PORTAL by Valve Software. It wouldn't have been nearly as cool if it weren't for the custom textures that really helped set the mood [see www.youtube.com/watch?v=4PBlqoBP_y4].

Another example of the players being right is the ladders. I resisted this for a long time on the basis that I've never ever enjoyed ladders in any game ever, but gave up after being convinced that having huge stairwells took up too much space. It turns out ladders don't get used as frequently as I feared.

3) NOT LISTENING TO ADVICE. While I did appreciate the advice I got from people who were supposed to know what they were talking about, a lot of it would've severely limited what I was able to do. People tend to give you advice based on personal experience, and just because something worked or didn't work for them in the past, it doesn't mean that it's valid for me now.

If I had listened to advice I never would've left my day job to start work on MINECRAFT in the first place, nor would I have charged for the game as early as I did. Without the early funding from early adopters, MINECRAFT would never have taken off. Another piece of advice I shouldn't have listened to, but unfortunately did, was to not quit my day job too early. It provided some economical security, but it turns out that really wasn't needed. I've ignored some advice that would've helped as well, such as suggestions to change the company form, which is expanded on later on under "what went wrong."

When someone starting their own thing asks me for advice, I usually tell them not to listen to advice. Then I give them advice anyway!

4) MAKING A GAME FOR MYSELF. I've always made games that I myself want to play and that I think are missing from the market, partially out of frustration of not being able to play those games. The great thing about making games where the target audience is people like yourself is that it's really easy to know whether you're appealing to that audience with your game.



changing THE GAME

Preparing Students for 21st Century Careers

DeVry University's bachelor's degree program in Game & Simulation Programming (GSP) positions students for success with an innovative experiential education.

Our graduates are well-rounded and ready to make an impact on today's ever-changing, demanding simulation and video game industries. The GSP curriculum includes training in a broad range of programming languages and software applications. These courses are integrated with a general education curriculum to reinforce essential critical thinking skills.

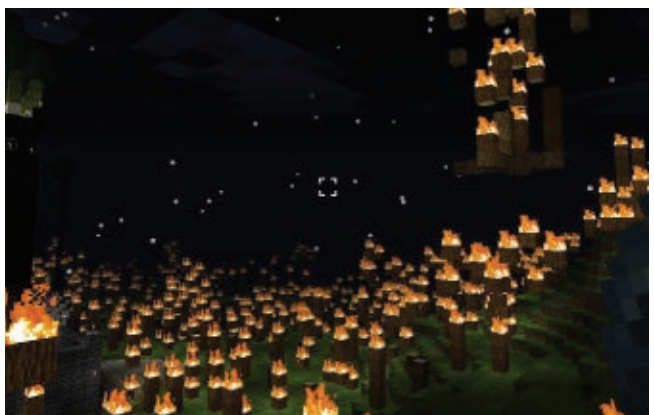
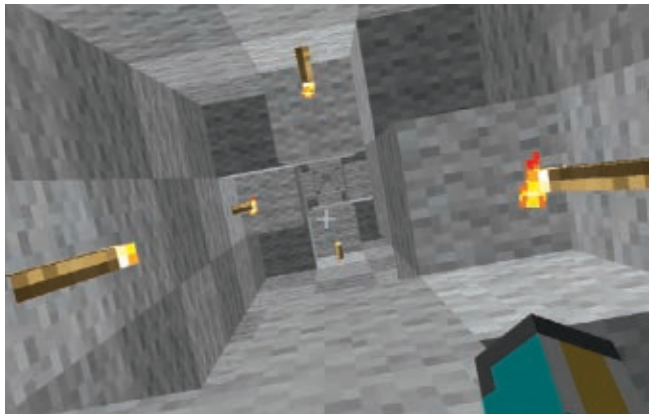
Discover education working at devry.edu

Visit us at The Game Developers Conference, booth #2408

February 28–March 4, 2011 | Moscone Center | San Francisco, CA



Creed



It seems obvious, but a lot of people seem to try to make games that appeal to a vague notion of a “general public.” If I like my own game, there probably are more people out there who would like the game as well, and the Internet is a huge market. That translates into quite a lot of potential customers. And fortunately for me, I seem to represent a fairly profitable target audience.

5) PURE LUCK. If you're not making a sequel, it's basically impossible to have any idea how well a game will do, especially if you don't have many years of experience like most publishers. I've tried to analyze why MINECRAFT has sold so well several times, and I come to slightly different conclusions each time. For example, I think it's usually fun to watch other people play the game, so that will drive the viral aspect, or that the random levels in the game made it feel more personal, so people would be more prone to talk about their experiences. However, I can't escape the fact that a large factor is that I just happened to make the right game at the right time just as the audience was starting to warm up to the idea of paying for indie games.

Platforms like the iPhone, XBLA, and Steam meant players were getting used to paying for games made by small teams, and MINECRAFT happened to come around at the same time. I had never even considered this, so it came down to just pure luck. I like to tell myself the success is mostly because the game is awesome, though.

The only paid marketing I tried was throwing about \$500 dollars into Google AdWords, which resulted in the game getting a very small number of clicks at a cost per user that was way higher than the conversion rate. Talking about the game with media and being public about my content patches always seemed to have the greatest results.

WHAT WENT WRONG

1) THE WRONG COMPANY FORM. Initially, I had a sole proprietorship, which meant that I was the company. In that company form, you can't keep money in the company at the end of the year and have to take out all the money as salary. In Sweden, getting a salary that high means you pay a lot of tax. Like over 60 percent. If I had started a limited company (“Aktiebolag” in Sweden) earlier, I would've been able to save more money for future development, and we wouldn't have had to spend several weeks working out the details for the new company.

Long hours with lawyers, waiting for papers, and talking to banks really take the fun out of running a company. Fortunately, now that everything's in place and we've got a really talented CEO to help run the company, I can just focus on the products again.

2) SUPPORT. Email doesn't scale well. At all. Initially, I replied to all emails I got. Then I started just reading them all and replying to the things that needed replying to. Then I started getting several hundred emails per day and was unable to keep up with them all. Unfortunately, this meant that a lot of support issues also went unnoticed when I failed to see them while skimming through the subject lines.

Now that we've got a company, we've got more people reading email, and we're trying to set up a better support system that doesn't use email at all. We're still struggling to catch up to the support need. I'm not sure how we'll deal with this in the coming months. We're setting up some systems, but it might end up with us outsourcing support. Outsourced support might be bad, but at least you can get replies from them.

3) TOO MUCH STRESS. When I find inspiration and get in the groove of programming, I can be very fast. As a result, I sometimes set up goals that are too tight, and when something happens that delays development speed, like an illness, or just an extra tricky bug, I get really stressed trying to get things out in time. On top of this, not being able to keep up with email meant I had a constant feeling of having missed some important information.

The biggest crash I had was before going full time on MINECRAFT, which resulted in me taking a three-week vacation where I did absolutely nothing other than play games and sleep. And recently, during the push to release beta, half the company, including myself, got sick. I did a couple of 12-hour days full of painkillers and coffee, and it really drained me. This is obviously not healthy. So for beta, if I realize a deadline will be hard to reach, I will say so instead of trying to reach it. We're putting some proper development methodology in place.

4) CODE REBOOT. After many months of working on the same code base, I was growing frustrated with it and decided to start over on a new engine mostly written from scratch. I guess this happens to most programmers. You get frustrated with some structure you put in place early on, or you think of a feature you don't think you can work into the current code base, and you decide to do a rewrite. This is almost never a good idea. Not only do you waste time duplicating work, but you end up with fewer features, because you either forget to add them or just never get around to them.

Sure, there were a bunch of cool new features as a result of the code rewrite, like the lighting engine and the infinite map size, but this could've been added to the old code as well. It might've taken some restructuring to do so, but it wouldn't have taken as long as the rewrite did and I wouldn't





PUBLISHER None
DEVELOPER Mojang
NUMBER OF DEVELOPERS 1, but I got help with music and sound effects
LENGTH OF DEVELOPMENT 18 months
RELEASE DATES The first version of Minecraft went live in June 2009, and the Beta version was released in December 2010
LINES OF CODE 80,000
DEVELOPMENT TOOLS LWJGL, Eclipse, Paint.net, Audacity, Red Bull
PLATFORM Mac, Windows, Linux

have lost important features like multiplayer. If you ever get the urge to do a rewrite, resist it!

5) WAITING TOO LONG TO ADD MULTIPLAYER. As a result of the code rewrite, I split up the game into “classic,” which was the old code, and “alpha,” which was the new game. I hadn’t ported over the multiplayer code to alpha. I decided to just focus on the single-player version of the game and cram in new features at a really high pace. This was probably my most creative time during development, but once it was time to implement multiplayer, the amount of work I needed to do was massive.

Huge features had been written without any concern for distributing it, game logic code was mixed with rendering code, and lots of assumptions had been made that don’t hold up in multiplayer, such as the idea that there would only be a single player for monsters to target. The huge workload meant that my motivation dropped, and so did development speed.

One particularly bad example is the inventory. The version I first implemented was basically just a UI hack to try out how it could be done, and when it came to moving this server side, there was a lot of work just simply separating logic from rendering. Even after that, no effort had been made to prevent clients from cheating and using items they hadn’t made yet. It ended up with me having to rewrite most of the code, and server-side inventory alone took several weeks to fix. These days, when we add a new feature, we try to make sure to test it

in both single player and multiplayer as soon as possible. Moving forward, we will probably change the game so that even when playing single player, you’re actually playing the multiplayer version of the game against a local server.

JEEPERS CREEPERS

/// The development of MINECRAFT has been chaotic and organic from the start, with me adding features that I felt were missing and fixing bugs when they were annoying. Some bugs, like pigs looking like tall pillars with four small stumpy legs, I kept and made part of the game. The way they moved had a very creepy feel to it, so I named them “Creepers” and painted them green. Turns out this was a good move, as Creepers have become something of an icon for the game by now.

When I started work on MINECRAFT, I was expecting it to be a 6- to 12-month project, and that it would hopefully make at least enough money to fund itself and the development of the next game. Now it’s been over a year and a half and we only just now got out of alpha. It’s time to focus on the full version, and the first step is for me to try to define what the full version actually is. We’ll keep releasing expansions and keep the game alive, but there needs to be some kind of final version that you can point at and say, “I did this!”

I’m not sure why I feel a need to have something to call the final version if we’re just going to keep updating it, but it just feels wrong to never have reached some kind of goal. Having the game

constantly be under development also seems to confuse the press. I’ve heard people question whether the game is actually released, and if so, what year did we release it? It feels like a good idea to have a point in time that represents the cut-off between building toward a fully self-contained game, and adding new content to that game.

We recently started looking more into integrating social media into the game, like perhaps the ability to record videos in-game to post to video sharing sites, or Facebook integration to allow players to find friends who also play the game, but it really seems to me like social media has its own ways of organizing itself around content it likes. There are several hundred thousand videos about MINECRAFT on Youtube, there’s a subreddit dedicated to it, and at least one 24/7 live streaming channel for it. While it can certainly help to have better integration to those services, I don’t think it’s a good idea to start with that aspect. If the core game isn’t interesting enough to show up on those channels, having good integration is just a waste of time.

The company has grown fast, and we’re seven people strong now in an office that could potentially house up to 15. We’ve got another game in the prototype stage, and more game ideas for the future. ☺

MARKUS PERSSON is 31 years old, and developed MINECRAFT from his home office. In the past, he worked on WURM ONLINE, and has made several games for game development competitions.



THE LARGEST GLOBAL PUBLISHER OF FACEBOOK GAMES!

6waves

70+ million monthly active users!

www.6waves.com

“ We aim to work with our developers to publish the best social games to cover every genre, language and platform. ”

-CEO, Rex Ng

Our Services

- Distribution**
 Promote games to our 70+ million active users on Facebook
- Marketing**
 Spending on Facebook and other Marketing activities to promote games
- Localization**
 Translate, adapt and launch games in major countries and languages
- Monetization**
 Maximize monetization potential by having local and global payment providers
- Hosting**
 Procure scalable, reliable, and affordable hosting with optimal architecture
- Optimization**
 Provide valuable feedback on virality, retention and monetization

Published Games



Ravenwood Fair



Resort World



A Thousand Suns



Birdland



Legacy of Rome



Big Business

Reviews

"We have been very impressed by the high level of personal service 6waves has provided so far and have not been shy about telling people about it. You guys are awesome and thanks for everything!"

Robert - COO, 5th Planet Games, Inc.

"I'm surprised our game could get so many players in just a few days! 6waves really saved us months, if not years, to build up our user base, not to mention the marketing dollars that would have been spent. My hats off to the 6waves team!"

Marcus - CEO, wizQ Interactive Inc.

Contact us now and make your game the next Big Global Hit!

Email us: bd@6waves.com

BLIZZARD

ENTERTAINMENT



WORLD
WARCRAFT



STARCRAFT



DIABLO

BLIZZARD IS HIRING

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

IRVINE, CALIFORNIA | AUSTIN, TEXAS | VELIZY, FRANCE | CORK, IRELAND
SINGAPORE | SHANGHAI, CHINA | TAIPEI, TAIWAN | SEOUL, SOUTH KOREA
SAO PAULO, BRAZIL | BUENOS AIRES, ARGENTINA | MEXICO CITY, MEXICO

jobs.blizzard.com

Visit us in the GDC Career Pavilion at Booth #2432.

©2011 Blizzard Entertainment, Inc. All rights reserved. World of Warcraft, Diablo, StarCraft and Blizzard Entertainment are trademarks or registered trademarks of Blizzard Entertainment, Inc., in the US and/or other countries.



pressed by the dark

building emotions with high-stakes play

C H R I S P R U E T T I L L U S T R A T I O N S B Y U N O M O R A L E Z

Respect for the player is a common theme in modern game design. Ramp difficulty up slowly. Secretly assist the player with aiming and jumping. Allow cut scenes to be paused or skipped. Reduce difficulty dynamically to allow failing players to progress. Let the player save anywhere. These methods are intended to lower player frustration, and to allow an audience wider than hardcore players to enjoy a game. Like good user interface design, this approach removes friction from the experience and lets the player focus on the content of the game.

But often this idea also manifests as easier gameplay. After all, one way to reduce frustration is to simply scale back the level of skill that the game requires. One recent example is KINECT JOY RIDE, a launch title for Microsoft's new motion controller, which has received some criticism for the dramatic levels of steering assistance that it provides (in one humorous online video, a player is able to place third in a race without actually steering at all).

There is some evidence, however, that harder, or perhaps more intense play can cause the player to be more emotionally affected by game content. The evidence is found in the Two-Factor Theory of Emotion, defined by Stanley Schachter and Jerome Singer in a psychological study in 1962.

The theory states that the brain normally generates emotional responses based on external stimuli, but can sometimes be tricked into generating a false response based on the body's physiological state. When the body is aroused and the brain does not have an obvious way to explain that arousal, it can misread the situation and cause us to respond emotionally in a way that we normally would not. I first read about Schachter and Singer's theory in a 2007 Gamasutra article by Dan Cook (see Resources, pg. 37). In "Constructing Artificial Emotions: A Design Experiment," Cook makes the case that the Two-Factor Theory of Emotion has a number of interesting implications for game designers. In this article, I'd like to delve into the theory itself, and consider how it might apply specifically to difficulty in games.

TWO FACTORS OF EMOTION

/// Schachter and Singer's theory defines emotion as the combination of two "factors:" physiological change and a cognitive label to explain that change. Their research grew out of a string of experiments in the '40s and '50s that attempted to establish a link between aggression and sexual arousal. Some research suggested a link, like the 1965 study by psychologists Barclay and Haber that compared levels of sexual arousal in students who were angrily berated by their teacher to those who were not. Schachter and Singer's approach was more general: they postulated it was the physiological effects of being the subject of aggression, rather than aggression itself, that lead to a rise in arousal.



Our bodies react to a lot of different types of stimuli, but often the mechanics of that reaction are similar. For example, fear, trauma, and sexual arousal can all lead to the release of adrenaline and an increased heart rate. Schachter and Singer's test attempted to separate this type of reaction from obvious context to see how the brain would respond.

The experiment they performed involved giving two groups of college students injections: one group received a shot of adrenaline and the other a placebo. Some students were told about the contents of the shot, but the rest were not told or were misled to believe it to be something else. Once injected, each student was paired with another student who was actually a confederate of the psychologists. The confederate was instructed to act in a certain way, either as though they were experiencing a euphoric high or as if they were very angry. The three important data points (psychologists call them "dependent variables") were the actual contents of the shot, what the student was told about his shot, and how the student behaved in the presence of the confederate.

The results of the test showed that students tended to follow the emotional response of the confederate. If the confederate acted angry or high, the students reported



feeling the same. But this only worked on the students who had received adrenaline and were not told what it was; those who received a placebo and those who were informed that they had been given adrenaline showed no increased emotional response. The effect was particularly strong in students who had been told to expect bodily responses not associated with adrenaline, like dullness and headache.

Schachter and Singer explained this result with the Two-Factor Theory. They suggested that when the body becomes excited and the brain does not know why (as was the case with the students who were unwittingly injected with adrenaline), it looks to external stimuli for clues on how to best emotionally label that response. In this case, subjects who were in the same room as a peer who acted high or angry misidentified their body's reaction to the adrenaline as an emotional response, and felt high or angry themselves. We might also guess that the students who were verbally abused in Barclay and Haber's experiment may have mislabeled physiological change that resulted from anger or fear as sexual arousal.

This is an interesting cognitive theory, but it gets really fascinating when we consider Donald Dutton and Arthur

Aron's 1974 experiments involving the link between fear and sexual arousal. Building on earlier Two-Factor studies, Dutton and Aron sought to test whether lust could be instigated in people by putting them in a scary situation. If the Two-Factor Theory of Emotion is correct, they thought, it should be possible to cause a person's body to react to fear and then confuse the person into thinking that they are aroused by introducing sexual symbols.

Their first experiment took place on a high bridge, the Capilano Suspension Bridge in Vancouver, BC. This is a wobbly bridge with low handrails that spans a 70-foot drop to a ravine, and as Dutton and Aron observed, most people cross it very slowly and very carefully because it's quite scary. They placed an interviewer on the bridge itself and asked men between the ages of 18 and 35 to answer a questionnaire and write a short story. At the end of the survey the interviewer provided their personal phone number in case the interviewee wished to call back later to learn more about the experiment.

Dutton and Aron performed this procedure twice, with two different interviewers: a man and an attractive woman. They also repeated the experiment on another bridge that was not so scary: it had solid construction, high handrails, and rose only about ten feet off the ground. The dependent variables for this experiment included whether the subjects accepted the interviewer's phone number, whether they called back later (probably looking for a date), and the level of sexual content present in the subject's survey answers.

For men who met the male interviewer, the responses were pretty clear: low sex content scores in the survey and few bothered to call back (most subjects did not even accept the phone number). This result was consistent regardless of whether the survey was performed on the scary bridge or the safe bridge. The responses from men who spoke with the attractive female interviewer on the safe bridge were similar: There was some minor increase in sexual content, but results were overall consistent with those of the male interviewer.

But those subjects who met the female interviewer on the scary bridge behaved very differently. The sexual content in their surveys increased dramatically, almost every person interviewed accepted her phone number, and 50 percent of them called her back later. By each measure, the subjects on the scary bridge were considerably more aroused by the attractive woman than they were on a safe bridge. This appeared to be strong evidence to support the Two-Factor Theory; Dutton and Aron suggested that the physiological effects of being scared by the bridge were indeed being misidentified by the male subjects as lust when the attractive woman was added to the situation as shown in Figure 1. (It's worth noting that the psychologists did not control for sexual orientation, but did select men that were not with a female companion. Their results suggest that the majority of their test subjects were heterosexual.)

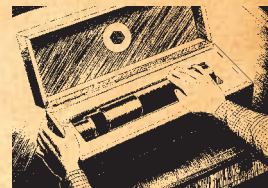




Image courtesy of Codemasters

CODEMASTERS

USES MORPHEME

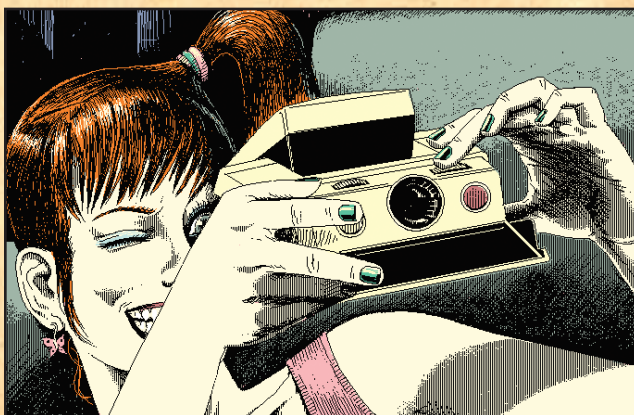
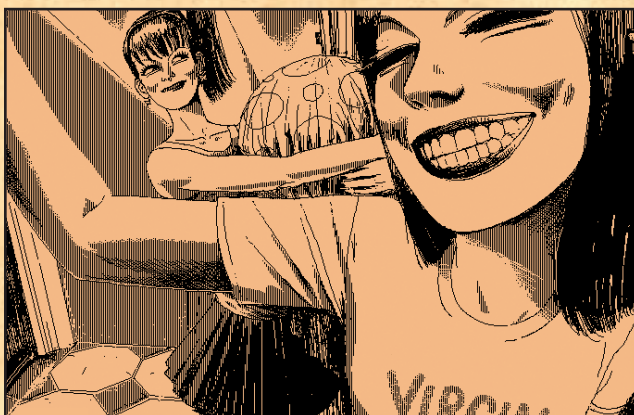
“The Morpheme engine and tools enable fast and compelling content creation, driving the animation quality in *bodycount* to new heights of realism.”

Bryan Marshall, CTO at Codemasters



bodycount





But Dutton and Aron are careful scientists, and performed a couple of follow-up experiments to confirm this result. One potential flaw of the previous test was the “lady in distress” syndrome—the idea that men might be responding to the situation because the female interviewer on the high bridge appeared to be in danger or helpless.

To address this problem, their last study took place in a lab. It tested the sexual content of questions answered by male college students who believed that they were about to be painfully shocked. In this study, college men were led to believe that they were participating in an experiment about the effects of pain on learning, and were paired with an attractive female student, who was actually a confederate. Both students were told that they would receive either a painful shock or a light shock, but before that they were asked to privately answer a questionnaire. The theory here was that men who experienced anxiety about the upcoming shock might mislabel their body’s reaction to that anxiety as sexual arousal due to the presence of an attractive woman.

The results of this study confirmed the previous experiments. Men who believed that they were about to receive a painful shock reported higher levels of sexual interest in their counterpart, and scored higher on sexual content in their survey results. The “lady in distress” syndrome was shown not to be a factor because men responded with higher levels of arousal regardless of whether they believed that the confederate would receive a painful shock as well. Men who were about to receive only a light shock showed no significant arousal.

These studies support Schachter and Singer’s original thesis: Dutton and Aron argued that the men in their studies reacted with lust because their bodies mislabeled physical responses to fear. They also showed that the same effect did not occur in safe situations; lust could only be generated when a sexual context was paired with anxiety about something else.

FROM THEORY TO PRACTICE

/// The Two-Factor Theory of Emotion is an interesting idea, but what does it have to do with game design? Consider this: have you ever become really excited while playing a game? I mean, the type of intense play where your heart rate is up, the blood is pumping, and you are acutely focused on the screen? I know I’ve felt this way; my hands used to cramp from holding the GameCube controller too tightly during particularly intense games like *RESIDENT EVIL 4*. If a game can increase your heart rate and cause the release of adrenaline, it’s already accomplished the first of the Theory’s two factors. If Schachter, Singer, Dutton, and Aron are correct, all the game needs to do now is to provide some context for your brain to respond to; conveniently, you’re already looking straight at the screen.

The Two-Factor Theory suggests that there is a very good reason to make games hard, or at least intense: By stressing the player out, the game has a better chance of causing an emotional response (see Figure 2).

Consider a horror game. If the goal of the game designer is to scare you, the Two-Factor Theory suggests that you’ll be much more likely to feel afraid if you are in an elevated physical state because your brain may mistake that state as the result of the images on the screen.

Indeed, many of the best horror games also feature extremely unforgiving mechanics. Classic *RESIDENT EVIL* games strictly limit player ammunition, health, and even saves. Games like *DEAD SPACE* are often reported as being more effective when played on harder difficulty levels. Combat in *FATAL FRAME* is about high-precision timing in an interface that severely limits your peripheral awareness. And games like *SIREN* and *MANHUNT* induce stress with sneaking and punishing combat mechanics. And at the same time, these games are filling the screen with zombies, flesh monsters, and long-haired ghost girls. They are providing both of the factors that the theory requires: difficult mechanics to prompt a physiological change, and scary content to provide a label for that change.

RESOURCES

“CONSTRUCTING ARTIFICIAL EMOTIONS: A DESIGN EXPERIMENT,” by Dan Cook, Gamasutra, 2007: www.gamasutra.com/view/feature/1992/constructing_artificial_emotions_.php?page=1



FIGURE 1

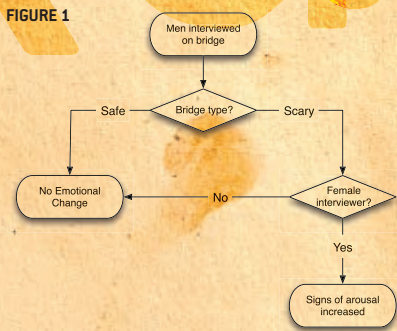
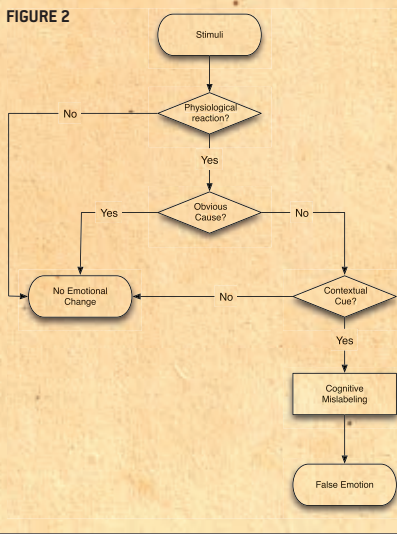


FIGURE 2



But the implications of Two-Factor do not end at horror games. Games like *SUPER MEAT BOY* and *DEMON'S SOULS* are widely loved despite their reputation for intense difficulty, and perhaps we can attribute that to a combination of stressful play and rewarding visuals. Is the effect of the trite sexual content in *GOD HAND* increased because of that game's hardcore combat system? How about the feeling of urgency caused by *METAL GEAR SOLID 4*'s memorable finger-destroying button mash finale? Reviewers of *MAXIMO: GHOSTS TO GLORY* consistently praised that game's unrelenting difficulty as well as its ability to make you feel incredibly awesome with each completed challenge.

Heavy difficulty can, of course, lead to heavy frustration, which is the enemy of fun. But if the goal is to increase the player's heart rate, perhaps difficulty is not as important as intensity of play. *ICO*, for example, is not a particularly difficult game, but the threat of losing Yorda to the shadow monsters is stressful. Every time the player must leave her somewhere to go complete a puzzle, the level of tension starts to increase because we know she can be abducted quickly.

The mechanics of *ICO* are not unforgiving, but the threat of failure is strong. Perhaps anxiety induced by that threat increases the player's connection to Yorda, and strengthens the impact of the final bridge scene. *CONDEMNED*, another horror game, features intense combat but also provides plentiful health items all over each level. The level of stress in the midst of a fight is very high, but the game is not difficult overall. Still, the Two-Factor Theory suggests that this intensity makes the player more vulnerable to the game's frightening content.

Maybe a better way to apply the Two-Factor Theory is to think about high-stakes game play. Even if the mechanics themselves are easy, players may react to stress when the cost of failure is very high and the threat of failure is imminent. *MAXIMO* and *RESIDENT EVIL* both ration saves. *FATAL FRAME*, *SUPER MEAT BOY*, and *RESIDENT EVIL 4* all put a huge emphasis on precision

maneuvers. Even short-term periods of stress, like a challenging boss fight, may be enough to get a physical reaction out of the player. From that point, the research suggests that content designed to provoke a specific emotional response, particularly a response that can be associated with an elevated physical state, has a better chance of affecting the player.

Or how about this angle: if exciting the player's physiological state makes it easier to convince them that they feel a certain way, why not accomplish that with actual exercise? Between Sony's *Move*, Microsoft's *Kinect*, and Nintendo's *Wii*, there are a lot of opportunities for players to break a sweat while playing a game. As long as the player does not consciously associate their movement with their body's arousal, the Two-Factor Theory suggests that physical exercise may open the player up to cognitive mislabeling. In this light, *Marvelous'* recently announced *IKENIE NO YORU* (lit. "NIGHT OF SACRIFICE"), a horror game that somehow involves the *Wii Balance Board*, might actually make sense.

Modern game design theory places a premium on respecting the player. But the Two-Factor Theory of Emotion suggests that games that err on the side of easy and relaxing gameplay may be shutting the door to a potential tool for emotional manipulation. Curiously, this means that one gateway to emotional relevancy (another recent buzz-word) may be play that encourages physical stress or exercise. Though it may be counter to many contemporary ideas about accessibility, high-stakes mechanics combined with suggestive content might actually be a better way to get an emotional message across. Perhaps the real way to a player's heart is through the adrenal gland. **RU**

CHRIS PRUETT is a game developer advocate at Google, focused on Android. Under the cover of night he writes indie games and blogs about horror game design. The views expressed in this article are his alone and not those of his employer. Chris lives with his wife and daughter in Cupertino, California.



Penn Engineering's

MASTERS IN COMPUTER GRAPHICS AND GAME TECHNOLOGY

offers recent Engineering and
Computer Science grads:

A unique, one-year graduate-level program
combining **art** with **engineering** and balancing
theory with **practice**

A **wide array of interdisciplinary courses** from
UPenn's School of Engineering and Applied
Science, School of Design, the Wharton School
of Business and the Annenberg School of
Communication

Access to the **LiveActor Motion Capture Studio**
for student game projects

Cutting edge research opportunities at the
Center for Human Modeling and Simulation

A world class education at University of
Pennsylvania's Philadelphia-based
Ivy League campus

GET IN THE GAME

www.cis.upenn.edu/grad/cggt/

COME JOIN THE

- *Team up with the creators of League of Legends, PC Gamer's Free-to-Play Game of the Year.*
- *Join a fun culture of passionate gamers.*
- *Take on leadership and responsibilities in a fast-paced work environment.*

We are looking for all-stars
in the following fields:

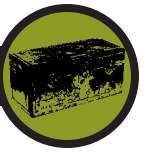
ENGINEERING
DESIGN
ART
QA
OPERATIONS
MUCH MORE!

Apply today at Careers@riotgames.com

or view all job opportunities at

www.riotgames.com/careers





TRINIGY VISION ENGINE 8

REVIEW BY SPELLBOUND ENTERTAINMENT TEAM

SPELLBOUND IS A LONG-

established German PC and console game development studio. We have completed a number of games using Trinigy's Vision Engine over the years, the most recent of which has been ARCANIA—GOTHIC 4. We have also been working extensively with the latest Vision Engine release (Vision 8.0) that came out in spring 2010.

As a studio, Spellbound mainly develops large-scale games for next-gen consoles and PCs, but we've also begun working on small, web-style projects. We find that the Vision Engine provides us with a practical and full-featured solution for getting game prototypes up and running, and then provides the flexibility that we require to add in our own optimized solutions for our projects' specific needs.

GETTING ACQUAINTED

» Focused primarily on high-performance rendering, the Vision Engine is a software development kit comprising a 3D engine and a set of tools. The SDK and tools are all actively maintained and improved with regular minor updates, containing miscellaneous fixes and enhancements, and one or two major upgrades per year. The runtime engine is multiplatform; it runs on PC, Xbox 360, PlayStation 3, and Nintendo Wii.

In addition to support for building classic stand-alone PC applications, the main new feature of Vision Engine 8.0 is called WebVision. It allows studios to integrate full 3D applications into web pages with minimal effort. WebVision is promoted as being compatible with most major

browsers, and although we haven't tested this extensively ourselves, we had no problem with the Internet Explorer and Firefox browsers that we use in-house.

The Vision Engine provides a toolbox of state-of-the-art graphical features for building rich 3D environments and content. The engine is well suited for large outdoor scenes as it natively handles levels of detail and streaming of large areas. We have prototyped a web-based project that makes use of these techniques to very good effect, streaming the data from the Internet in the background as the player progresses.

The engine supports a wide range of graphics hardware and comes with all the common features that one might expect, including an animation system, 2D overlays to display GUI objects, a particle system, shadowing technologies, and so on. The core engine also supplies solutions for multithreading, data streaming, and optimized memory management, as well as a large collection of ready-to-use shaders for both forward and deferred rendering. For those of us who like to hone the engine to our needs, there is support for custom render loops that allows studios to directly control the rendering process.

The Vision Engine's interface is highly modular and easily complemented with external libraries. The SDK ships with plugins for many popular third-party middleware packages ranging from physics (PhysX and Bullet Physics) to AI (Kynapse and xaitment) and dynamic vegetation (SpeedTree). At the time of writing of this article,

there were 18 of these integration modules available and more that we know of in development.

Taking tree rendering in GOTHIC 4 as an example, the SpeedTree integration provided by the Vision Engine was used during prototyping. It was great for the small test scenes in the prototype, but we knew that we needed to support upwards of 50,000 trees in the full game. With the Vision Engine, we found that we could effortlessly extend the integration package to support the additional features that we wanted to add to make this possible.

Trinigy provides a wide variety of licensing options depending on the needs of the development studio. Smaller indie studios can take advantage of the fact that Trinigy factors in a project's budget when determining the licensing fee. They also provide a free, time-limited fully featured evaluation version.

A LOOK UNDER THE HOOD

» Before going any further, it is worth noting that the Vision Engine is a SDK for C++ application development and not a stand-alone scriptable game engine. Although there is support for LUA scripting, including a LUA debugger, the main API is in C++ and is clearly designed for programmers with a good grasp of object-oriented programming techniques.

From a technical perspective, the Vision Engine provides a great deal of flexibility and has allowed us to do pretty much anything that we've been able to dream up. This has been one of the biggest reasons why we continue to work with the game engine today.

New developer-specific features are implemented by extending Vision base classes and registering their types in the engine's type management system which encompasses all objects based on Vision's API, including both the built-in types and the user-defined entity types. New features can also be exposed in dedicated plug-ins that can be used by the WYSIWYG editing tool vForge.

Trinigy VISION ENGINE 8

STATS

Trinigy GmbH
INKA-Businesspark
Arbachtalstr. 6
72800 Eningen
Germany
www.trinigy.net

PRICE

License costs are negotiated on a sliding scale based on project budget. No royalties.

SYSTEM REQUIREMENTS

Microsoft Windows operating system. Visual Studio 2005 or newer.

PROS

1. Highly flexible.
2. High performance.
3. Top-notch competency and availability of support team.

CONS

1. Requires familiarity with basic math and rendering concepts.
2. Documentation sometimes lacks precision.
3. Running out of addressable memory space in 32-bit version of the vForge editor.



On the subject of documentation, our views are a little more mixed. The reference documentation is succinct, sometimes to the point of lacking the details that one is looking for. This can be a little irritating at the beginning, but the SDK does ship with numerous samples, showcasing nearly every feature of the engine, which is really helpful when it comes to getting a better grasp on the Vision Engine's inner workings. Trinigy also hosts a developer site with a support forum that provides quite a lot of useful information.

The place where Trinigy really shines is in technical support, which is exemplary. The support team is reactive, professional, knowledgeable, and in direct contact with the engineers writing the software.

Just to illustrate, during the development of *GOthic 4* we had several situations where we

discovered an issue that was critical for us and that required a change or fix on the engine's side. Even when we reported the issue quite late at night, we typically ended up with a fix or work around for the issue the same day! This gave us real peace of mind.

Before closing the hood, it's worth mentioning engine updates (major releases once or twice a year). In our experience, integrating a new Vision Engine version has required only a minimal amount of work on our side, and we have never seen an update introduce major bugs. Trinigy's testing and version release processes are clearly sound.

DOWN THE PIPELINE

» The Vision Engine comes with a comprehensive set of tools to generate and modify game data: an animation tool, a bitmap font generator, viewers to display

textures, 3D models, 3D scenes and visibility zones in a scene, as well as a tool dedicated to profiling and runtime tracking of in-game resources. The engine also includes a script debugger to track errors in LUA script.

An editor called vForge serves as the main content integration platform for the Vision Engine. It's a nice tool that provides an array of different dedicated editing layouts that allow studios to manipulate different types of game content efficiently. Because vForge uses the game code to display the 3D scene, studios get true WYSIWYG editing, which cuts down enormously on the number of iterations that are required to achieve the desired result. Its interface is user friendly and we find that new recruits are able to learn to use it very fast.

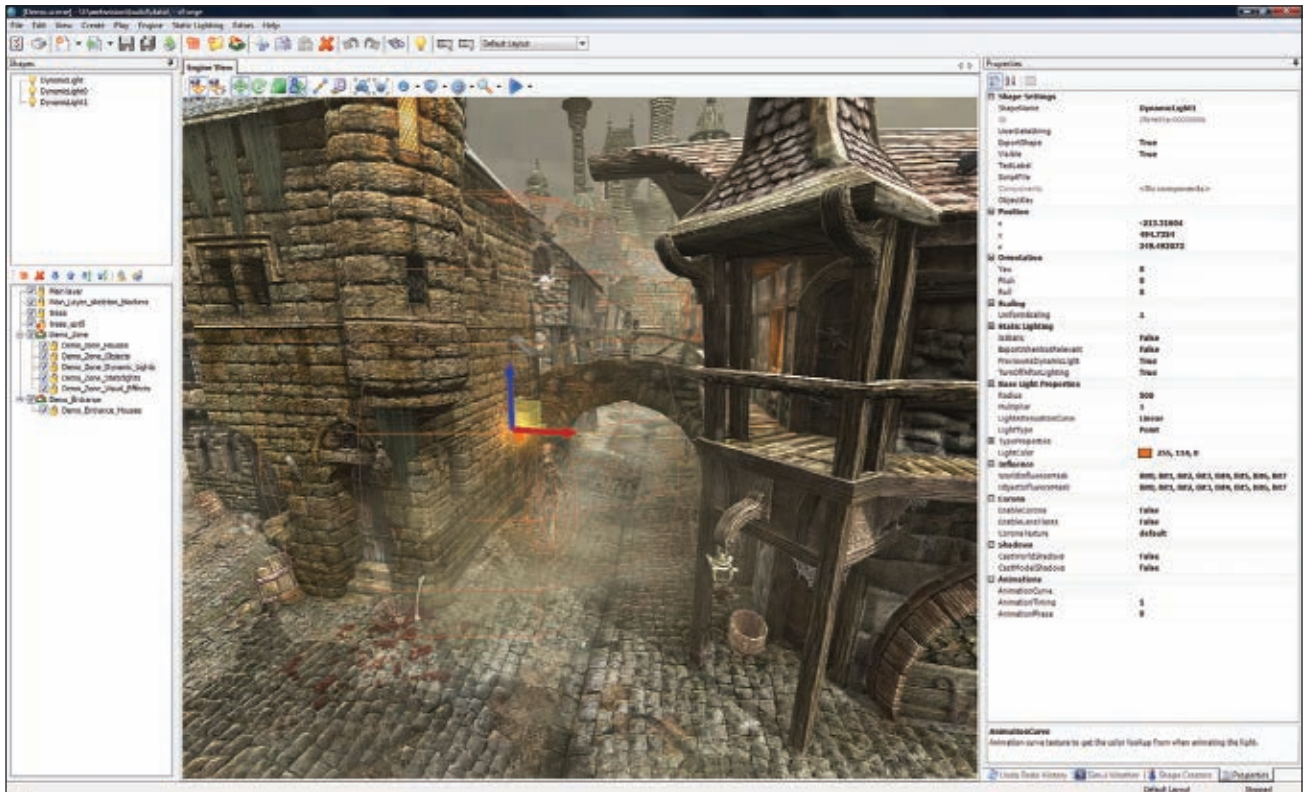
The tools are ergonomic, and Trinigy has always proven to be receptive to input and requests from our content creators. They include a lot of cool options to help studios find objects with a smart search function, to help modify properties of several objects at a time, to modify the lighting result on an object by placing a virtual position for the object and taking the lighting from this position instead of the real position, and so on. There are far too many of these very useful small options to list here, but they all make one's life easier and contribute to our team's enthusiasm for the tool set.

vForge also offers a lot of really helpful profiling and debugging information, such as displaying scenes with debug shaders that show only the lighting result, the normal map result, the amount of overdraw, and texture mip levels. It also has debug flags, which add the number of polygons, the object render order and suchlike, over each object in the scene.

The complete pipeline for level creation in Vision Engine is composed of the following elements:

- Creation of a height field terrain and the painting of textures over it.
- Placement of small objects and vegetation on the terrain with the aid of randomized systems that allow one to paint zones and choose different types of random distribution for each object type.
- Height field editing in vForge (for games using height fields).
- Scene assembly, integrating objects exported from 3ds Max, Maya, or Softimage.
- Building level logic with the Vision Engine's component and script editors by placing actors with custom classes and properties in the scene, assigning them paths, and so on.
- Configuration of both dynamic and fixed pre-calculated lighting (for light maps and light grids) in the vLux tool.
- Creation of particle effects in vForge's particle editor (another tool that our team likes), and previewing the results directly in the scene.
- Creation of new shaders using either an integrated editor designed mainly for programmers and technical artists or a new node-based, artist-friendly visual shader editor (which we don't have significant hands-on experience with).
- Assignment of shaders to objects and configuration of rendering properties (specular intensity, color, and so on).
- Placement of visual effects of different types such as volumetric cones, sun glares, water effect with real reflection, cloth simulated objects, and so on—a library of these effects is provided with the engine.

FINAL BUILDER
www.finalbuilder.com/game



Dynamic light placement can be easily adjusted in the vForge editor.

For all the good we have to say about the Vision tools, our experience during GOTHIC 4 was a little more difficult. We used a very high-resolution height field [25 cm grid spacing] for a terrain of several kilometers by several kilometers in size. We were using the 32-bit vForge executable, as the 64-bit version had not yet been developed, and we ran into big problems with vForge running out of addressable memory space, and crashing. This problem would not appear when using a 64-bit executable, so one can consider it fixed for the future, but it was extremely troublesome at the time.

For GOTHIC 4, we also developed our own editor for setting up AI, quests and the like. At the time, this made sense (quite apart from anything else, to avoid running out of memory), but it did introduce some overly complicated workflows. For the future, we are working on integrating our own

editing code back into vForge (via plug-ins) in order to profit from the tool's advantages.

VISION ON THE WEB

» As mentioned above, Vision Engine 8 comes with the possibility to build embedded games for the web. A WebVision game is basically a Vision Engine game that runs in a web page, with its data being streamed over the network and decompressed in the background on the fly. The WebVision plug-in handles the communication with the browser and forwards external events via a messaging system to the user application. Although fairly new and not yet quite as mature as some of its competitors, it offers many of the same performance and features found in stand-alone game engines, such as forward and deferred rendering, post-processing, special effects, and streaming of large environments, among others.


Via the WebVision system, any project based on the Vision Engine may also be deployed over the Internet and played within a browser, provided that the game logic supports streaming of game data which, packaged into encrypted archives, are streamed from the server and uncompressed at runtime in system memory.

Trinigy provides WebVision plug-ins for nearly every leading PC web browser, including the most recent versions of Internet Explorer, Firefox, Opera, and Chrome. Currently, WebVision is free of charge for studios that license the Vision Engine.

CLEAR VISION

» The Vision Engine has been used successfully in the past by Spellbound on multiple projects and platforms, the most recent being GOTHIC 4 on PC and Xbox 360. It has provided the flexibility, toolset, and support that we've required.

Trinigy's licensing options are more flexible than some of its competition, and middle-to-large game developers will definitely be enticed by its feature set, toolset, multiplatform availability, excellent support, and affordable price.

WebVision is a new middleware in the web space, providing a flexible and feature-complete game engine capable of exploiting the full power of a user's PC. Where competing middleware focus more on accessibility, providing solutions where simple games can be assembled with a few mouse clicks and a few scripts, Trinigy's solution is designed for creating larger and more complex applications—paving the way for a new generation of browser-based games. 

SPELLBOUND ENTERTAINMENT is the creator of the AIRLINE TYCOON series as well as the developer of ARCANIA - GOTHIC 4, the studio's latest title, which was released in October 2010.



product news

Asus Using PrimeSense Tech www.asus.com

Computer hardware maker Asus has announced it will team up with Kinect hardware maker PrimeSense to bring a similar controller-free, natural motion interface to a PC-based living room system later this year.

The WAVI Xtion system will use a dual camera sensor—similar to that used in the Xbox 360's Kinect—designed to hook up to a living room TV and connect wirelessly to a PC up to 25 meters away. Using the system, Asus promises users will be able to browse PC multimedia, internet, and gaming content from the comfort of their living room using nothing but their body movement.

Asus is targeting a 2nd quarter 2011 commercial release for the setup, though a "Pro" software development kit is being readied for February release. The company is also readying an online store for distribution of Xtion-powered software.

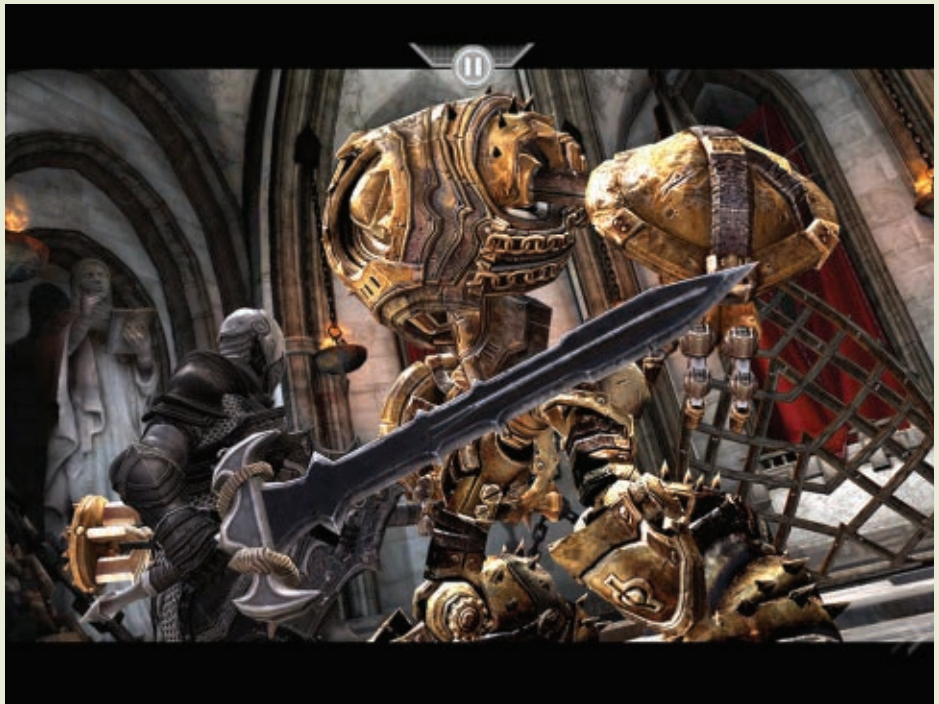
The Xtion development environment is built on top of PrimeSense's OpenNI initiative, a set of open source natural motion middleware and drivers PrimeSense released to the development community in December.

Unreal Dev Kit Arrives For iOS www.unreal.com

Epic Games' Unreal Engine dev kit, used to create the App Store graphical powerhouse INFINITY BLADE, is now available for other developers on iOS devices.

UDK for iOS will allow iPhone, iPod Touch, and iPad developers to create games using the same tech behind games including GEARS OF WAR, MASS EFFECT, and dozens of other Unreal Engine-powered PC and console titles.

Like standard UDK agreements on other platforms, developers can experiment with the engine or make free Unreal Engine-based games without incurring a licensing fee. Developers that want



INFINITY BLADE.

to sell UDK apps for iOS will have to pay a \$99 licensing fee and 25 percent royalties after the first \$5,000 in sales.

Epic recently released its first iOS game, INFINITY BLADE, an action-RPG developed by subsidiary studio Chair Entertainment, creators of the Xbox 360 downloadable title SHADOW COMPLEX.

NaturalMotion Releases Morpheme 3 www.naturalmotion.com

The latest release of NaturalMotion's animation middleware, Morpheme 3.0, is available now, with support for Microsoft's Kinect and Sony's Move controller interfaces.

NaturalMotion describes specific nodes for Kinect that filter and re-target data onto existing rigs or allow for indirect gesture-based character control. The company also says Morpheme 3.0 features performance enhancements like reductions in memory usage, improved workflow

and more debugging tools.

Morpheme 3.0 also features new transitions including "Active State" and "Self," aimed at making the creation process faster and more transparent. The tool's preview functions have been enhanced, according to NaturalMotion, with the aim of easing prototyping and testing interactions among multiple characters.

A new asset management system has been introduced with the intent to help developers better organize their clips and build physical characters more quickly. A "Node Wizard," intended for programmers, has been introduced to allow for more customization and better visibility on runtime executions and debugging within Connect.

OnLive Built Into Vizio TVs, Blu-ray Players, Mobiles www.onlive.com

Cloud gaming company Onlive is partnering with the rapidly-growing

HDTV company Vizio to incorporate the game streaming service into internet-enabled TVs, alongside apps like Netflix and Pandora.

TV-integrated OnLive will be a new venue for consumers to play games including WORLD OF GOO, BRAID, MAFIA II, BATMAN: ARKHAM ASYLUM, BORDERLANDS, and other major releases available on the service.

OnLive's cloud-based infrastructure relies on remote servers that host games that users access through broadband connections.

These servers handle the brunt of the processing for games, and don't require a local download, installation or disc-based consoles like the Wii, Xbox 360, or PlayStation 3. OnLive offers various payment methods, including a full "PlayPass" and a Netflix-like subscription plan.

Onlive CEO Steve Perlman said because Onlive is cloud-based and dependent on constantly-upgraded remote servers instead of local

hardware, the TV may age over the years, but the service will continue to improve as datacenters get more and better servers.

Sales of the recently-released MicroConsole will continue following the Vizio deal. That device, released in 2010, connects televisions to OnLive's remote servers via a broadband connection.

Rad Game Tools Launches Telemetry

WWW.RADGAMETOOLS.COM

Rad Game Tools recently launched Telemetry, a new tool the company describes as a programmer-driven profiling system that visualizes execution flow for game developers.

The Kirkland, WA-based company said Telemetry utilizes a client/server architecture in which the game relays performance information to a server, which then processes that data.

Telemetry's client, Visualizer, lets programmers more easily identify performance issues, Rad stated. The firm expects Telemetry to reveal performance problems that game makers otherwise wouldn't know exist.

Currently, the tool is only available for Windows, but Rad said support for other platforms will be announced at a later date.

Rad is also the company behind the Bink video codec, the Miles Sound System for game audio, and the Granny 3D animation toolkit.

Sony Bravia TV Includes Built-In PS2 In UK

WWW.SONYSTYLE.COM

Just before the holidays, Sony released a new 22-inch, 720p Bravia television with a unique feature—a full-fledged disc-based PlayStation 2 integrated into the television's stand.

Also onboard the unit is Bravia Internet Video, which allows access to select on-demand media. As of press time, there was no word of plans for a U.S. release of a similar PS2 TV.

The television features two PS2 memory card slots, two controller ports and two USB ports built

into the TV/console's face, plus a USB port behind the monitor. The integrated console also has its own Ethernet port, separate from that of the television monitor. Additionally, the TV itself has four HDMI ports.

The new model could serve as a rudimentary hint of the future, as companies examine the role of gaming- and internet-ready television sets in the marketplace.

SoftKinetic Coming To GameTree TV Set-Top Boxes

WWW.SOFTKINETIC.NET

After Microsoft's Kinect and ASUS' Xtion, Transgaming has announced a new, purely gesture-based control system for its on-demand GameTree TV games, which are available on many TV set-top boxes.

Using iisu-branded middleware from SoftKinetic, a new software development kit will allow developers to create and distribute gesture-controlled games through select set-top boxes equipped with Intel CE Media Processors, including many cable boxes and Blu-ray players.

SoftKinetic's iisu middleware is compatible with Adobe Flash, Unity 3D and 3DVIA Virtools development environments and allows for easy multi-platform development of games with a gesture-sensing component, the company says.

Transgaming's GameTree TV service launched last Fall with 30 casual games that can be controlled using a remote control, including PLANTS VS. ZOMBIE and WORLD OF GOO. The platform is currently available in select European markets, though the company says it's planning for global expansion this year.

Intel Announces Sandy Bridge Microprocessor

WWW.INTEL.COM

Intel recently unveiled the new, second-generation core i7 processor at the Consumer Electronics Show, where Valve placed its vote of

confidence behind the new "Sandy Bridge" microprocessor, and even designed PORTAL 2's PC version with the processor in mind.

Valve's Gabe Newell claimed the integrated CPU and graphics technology would allow for a "console-like experience on the PC."

The new 32-nanometer microprocessor includes sophisticated built-in capabilities for 3D graphics, as opposed to previous CPU-heavy Intel chips.

Intel claims it performs tasks like Microsoft spreadsheet work hundreds of times faster than the previous generation chip.

Mad Catz To Release Cyborg Gaming Lights

WWW.MADCATZ.COM

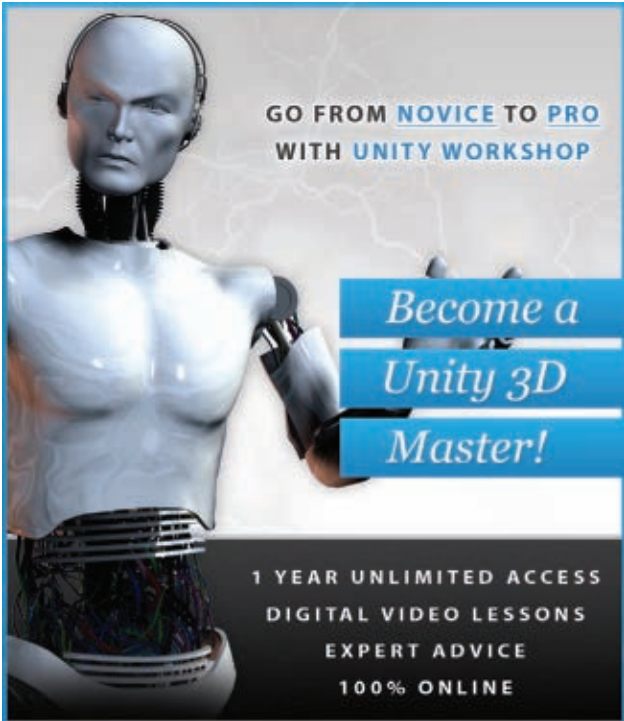
Peripheral manufacturer Mad Catz announced a partnership

with entertainment lighting company ambX to release the Cyborg Gaming Lights, a system for PC games that synchronizes colored lighting effects with in-game events.

Users place a series of lights around their monitor or desk, connect them to their PC, and the system will light up to provide ambience and spatial awareness.

Mad Catz says the system will help players identify the location of enemy fire, for instance, as the lights illuminate in certain areas to communicate direction.


To fully take advantage of the system's features, games must include support for the abBX technology. Titles already supporting abBX include COMMAND AND CONQUER 3, FAR CRY 2, and CRYSIS.



GO FROM **NOVICE TO PRO**
WITH **UNITY WORKSHOP**

Become a
Unity 3D
Master!

1 YEAR UNLIMITED ACCESS
DIGITAL VIDEO LESSONS
EXPERT ADVICE
100% ONLINE

 **unityworkshop**

WWW.UNITYWORKSHOP.COM | 1.877.895.6882



THE POWER TO ACHIEVE YOUR VISION

Visit us at
GDC EXPO

Real-time all the time with CryENGINE®3 – even in Stereoscopic 3D.

CryENGINE®3 Sandbox™ gives developers full control over their multi-platform creations in real-time, with improved tools enabling the fastest development of games on PC, PlayStation®3 and Xbox 360™. All features of CryENGINE®3 games can be produced and played simultaneously with Crytek's "What You See Is What You Play" and LiveCreate™ systems, newly enhanced with an innovative solution for developing high quality Stereoscopic 3D (S-3D) games, in real-time, without compromise on quality or performance.

If you are interested in evaluating CryENGINE®3 for your next project, contact us at cryengine@crytek.com



BEHIND THE MIRROR

ADDING REFLECTION TO C++

Reflection is a programming language feature that adds the ability for a program to utilize its own structure to inform its behavior. Reflection has its costs, but those are often outweighed by the ability to automate the serialization of objects into and out of a file, cloning, comparison, search indexing, and network replication, type conversion (copying base data between derived class instances), and user interface generation.

Of course, all these tasks can be accomplished without reflection capabilities, but you will likely pay higher costs having to write code that is very rote and prone to error. A good implementation of reflection can provide a platform on which each of these problems can be solved without glue code in every class that desires these features.

At the highest level, reflection can encompass many different features, such as runtime knowledge of class members (fields and methods), dynamic generation and adaptation of code, dynamic dispatch of procedure calls, and dynamic type creation.

However, for the purposes of this article, I will define C++ Reflection to mean "having access at runtime to information about the C++ classes in your program."

RTTI

» Before diving headlong into how to add reflection to C++, it's worth noting what type of information is already built-in. The C++ language specification provides minimal information about the classes compiled into a program. When enabled, C++ Run Time Type Information (RTTI) can provide only enough information to generate an id and name (the typeid operator), and handle identifying an instance's class given any type of compatible pointer (dynamic_cast<>).

For the purpose of game programming, RTTI is often disabled entirely. This is because its implementation is more costly than a system built on top of C++. Even if a program only makes a handful of RTTI queries, the toolchain is typically forced to generate, link, and allocate memory at runtime for information about every class in the application (that has a vtable). This significantly increases the amount of memory required to load your program, leaving less memory available for face-melting graphics, physics, and AI. It's better to implement your own RTTI-like system that only adds cost to the classes that need to utilize it. There are plenty of practical situations where vtables make sense without needing to do runtime-type checking.

Thus, the first step in implementation of a reflection system is typically a user implementation of RTTI features. This can be accomplished with only a couple of steps. Type information can be associated by a static member pointer (which also makes a good unique identifier for any given type within the program). In addition, some virtual functions allow querying an object's exact type, as well as test for base class types:

```
// Returns the type for this instance
virtual const Type* GetType() const;
// Deduces type membership for this instance
virtual bool HasType( const Type* type ) const;
```

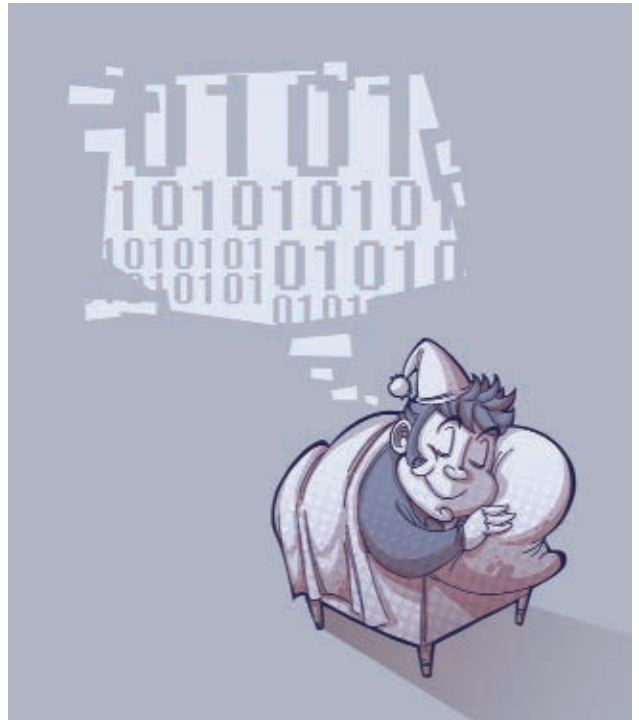


ILLUSTRATION BY JUAN RAMIREZ

GetType returns a pointer to the static type data, and HasType compares the provided type against its static type pointer as well as every base class' type pointer. This gives us all the information needed to reimplement dynamic_cast<>, but it only adds overhead to classes that are worth paying the added cost of type identification and type checking.

VISITOR PATTERN

» The simplest technique for implementing reflection is to take a purely programmatic approach. Virtual functions can be a mechanism for the traversal of all fields in a class. The visitor design pattern provides an abstraction for performing arbitrary operations on the fields as in Listing 1.

This is a textbook implementation of the visitor design pattern. Objects deliver the visitor to each one of its fields and the visitor gets an opportunity to transact with each field in series. It offers excellent encapsulation since the object does not know or care about any implementation details of what the visitor is trying to accomplish.

This technique does not require data from an external tool to do its job since it's implemented entirely in the code compiled into the program. It's simple to step through and debug, and extensible since many operations can be implemented as another class of Visitor.

With this approach, the development cost is small. A single line of code for each field in every class in your codebase is a fair price to pay to attain the benefits reflection can provide. However, there are some drawbacks with



INDEPENDENT PROPELLER AWARDS

* at South by Southwest *

indiePub is accepting indie game submissions for the First Annual Independent Propeller Awards Competition which will come to life at South by Southwest's ScreenBurn in Austin, Texas this March. Finalists will get the chance to demo their game at SXSW where \$150,000 in cash and prizes will be given away, and developers will get a shot at having their games published by indiePub through its sponsor, Zoo Publishing.

Check out indiePubGames.com to see how your game could become the next big thing!

Register Today at
WWW.INDIEPUBGAMES.COM
Submit by Feb 18th



using a visitor function for reflecting upon your objects. There are a lot of virtual function calls happening to interact with each field in a class. This is a concern for performance-critical code, and on certain platforms. Also, this technique is best suited for operations that want to visit every single field of a class. There are many situations where this work is not required, and iterating over every field just to access a few is wasteful and time consuming (depending on the size of the object).

DATA MODEL

» To really take reflection to the next level, it's necessary to be able to address specific fields and read and write data without iterating over every field in the class. A data model that represents the classes and fields specified in the code is needed to accomplish this. At runtime, your program can reflect upon this model to interface with objects and their field data.

This data model is owned by a central registry of type information. This singleton object owns all the type information in the program and can have support for finding type information by name. It's also a central point where a map of the entire inheritance hierarchy of classes can be built. The registry can be populated by employing a parser tool to analyze your source code, or by adopting a method similar to the visitor function approach to populate this data model at program startup.

TO PARSE OR NOT TO PARSE...

» Using a parsing tool to analyze your code introduces a lot of complexity. C++ has a very complex syntax. While there are some tools you can take off the shelf to do the parsing, there is still a lot of work to do to make that data usable at runtime. Typically, you want to extract just the necessary data from the abstract parse tree and write out a meaningful representation of only the data that is required for what you want to reflect upon. Templates, typedefs, functions, and other language features are generally overkill for the purpose of reflecting upon fields in a class.

A parsing tool is probably going to do one of two things: write a data file to be loaded at runtime (or packed into the executable as a global variable or resource section), or generate some code that gets compiled into your program.

If you choose the data file route, you have the added task of computing member size and offset information. This information is compiler specific and target platform specific. By choosing this approach, you are committing to abide by the padding and alignment rules of whatever compiler you use to build any given version of your program. Another source of complexity comes from the existence of two independent pipelines processing information about your code: the compiler and the parsing tool. This necessitates synchronizing the data output from the tool with the specific version of the compiled program, which will make packaging and deploying your program harder. Synchronization is a very important problem to solve in this approach because not detecting out-of-sync reflection information can cause nasty bugs (and potentially mangled data).

If you choose to generate source code to be compiled into your program, you inherit the burden of the complexities that come with creating a code generator that is most likely specific to your particular needs. The code generation tool will probably need to make a bunch of decisions about how your code needs to be decorated and organized. These requirements will change as your codebase evolves, and it will require you to be diligent about releasing and configuring your own build tool. Also, maintaining a tool that governs the ability to compile your game is risky because it has a tendency to break at the worst possible time (during a milestone).

The reward for using these approaches is tangible. You don't have any code that needs to be written by hand to reflect upon your classes. If you choose to generate code, then you will also probably get great performance since you can generate function bodies that do specific operations on every field of your classes, just like you would have done if you weren't using reflection at all.

In reality, there are a ton of moving parts when using this approach. Things can break in hard-to-trace ways if any step of the pipeline doesn't work as

LISTING 1

```
// a base class for any object that wants to reflect upon any class'
// fields
class ObjectVisitor
{
public:
    virtual void VisitField( int32_t&, const char* ) = 0;
};

// an example of a class that would write/read from/to each
// field to/from a file
class SerializeVisitor : public ObjectVisitor
{
public:
    virtual void VisitField( int32_t& value, const
        char* name )
    {
        // do serialization work
    }
};

// a base class for some of your reflection-aware objects
class Object
{
public:
    virtual void Accept( ObjectVisitor& visitor ) = 0;
};

// an example of a derived class that has a reflected field
class Foo : public Object
{
public:
    virtual void Accept( ObjectVisitor& visitor )
    {
        visitor.VisitField( m_Number, "Number" );
    }

private:
    int32_t m_Number;
};
```

expected. Having implemented and maintained this technique for many years, I can tell you that there are days when it feels like the planets have to align for all the parts in this complex pipeline to actually work together in harmony.

HAND CODING

» Alternately, code can be written to populate the reflection data model when our program starts up. This code creates class information structures, populates them with information about every field within the class, and adds them to the registry. Writing this code sounds arduous, but C++ template support provides some excellent tools to accomplish this with remarkably concise and manageable code. A good goal for this is to extract as much information as possible in a single function call per field, per class (just like our visitor function). This allows us to avoid any time spent at build time processing source, managing dependencies on build tools, dependency checking generated code, and synchronizing externally loaded data.

POLYMORPHIC DATA

» Because containers in C++ are template types instead of concrete types, function overloading can only take us so far. Since each template instantiation is a completely different type, trying to support containers using a visitor pattern could lead to a combinatorial explosion in the number of overridden



HOMO SAPIENS



TEXTURE SAPIENS

TEXTURES JUST GOT A WHOLE LOT SMARTER.

**DISCOVER THE EVOLUTION OF TEXTURING AT GDC
BOOTH #3014 SOUTH HALL.**

Substance Designer is our complete, smart texturing solution for creating dynamic textures, noises and advanced processing filters for your projects. For more information about our smart texturing solutions, and to request evaluation licenses, visit www.allegorithmic.com



Allegorithmic's
**Substance
Designer**

Texturing Just Got Smarter

 **allegorithmic**



LISTING 2

```
// Class stores all of the information for a class of object in your code (client object or
// data object):

struct Class
{
    const Class*      m_Base;    // our base class
    Array< const Class* > m_Derived; // our derived classes
    const char*      m_Name;     // our name (user-friendly)
    Array< const Field* > m_Fields; // fields of this class

    Class( const char* name )
        : m_Base( NULL )
        , m_Name( name )
    {
    }
};

// Field stores information about a particular member variable in a class. Fields are stored
// in an array in the Class object that owns them.

struct Field
{
    const Class*      m_OwnerClass; // the class this is a field within
    const Class*      m_DataClass;  // the class of data that serializes the field
    const char*      m_Name;        // name of the field
    size_t           m_Size;        // the size of the field
    uintptr_t         m_Offset;     // the offset to the field

    Field( const Class* owner, const Class* data, const char* name, size_t size, uintptr_t offset )
        : m_OwnerClass( owner )
        , m_DataClass( data )
        , m_Name( name )
        , m_Size( size )
        , m_Offset( offset )
    {
    }
};
```

LISTING 3

```
template< class ObjectT, class DataT >
Field* AddField( Class* owner, DataT T::* field, const char* name,
const Class* data = NULL )
{
    // call out to a template function that is specialized to return the appropriate data
    // class for this type of field
    // also compute the offset from the base pointer using the pointer to the member variable
    Field* field = new Field( owner, data ? data :
DeduceDataClass< DataT >(), name, sizeof( DataT ),
GetFieldOffset( field ) );
    owner->m_Fields.Push( field );
    return field;
}
```

functions. Enumerated data types present the same challenges. It's not easy to support them via overloading, since every enum in the entire game would need a different overload.

A solution to this shortcoming is to delegate the handling of any piece of data to a separate class of object that can interface with individual fields using a pointer. This will give us the ability to operate on any data in a polymorphic manner, including integer, floating point, and enumerated data types. Many languages that require derivation from a canonical Object class do this already. Adding support for treating simple types with polymorphism doesn't mean that it's necessary to use the polymorphic versions of these types everywhere in your code. They will only be used to abstract away the implementation details of dealing with serializing, comparing, and converting data to and from human-readable strings (which is very handy for generating property UIs).

Truly polymorphic data can solve many edge cases and provide extensibility for user types like enums and exotic containers. It can also support user data types that need custom processing during serialization. If these data classes store a value in addition to working through a pointer, they can be used to interface with fields and store standalone data. This allows for interoperability between versions of the program that have slightly different fields without discarding this "unknown" information. This is a major coup for game development tools that revise sets of properties frequently between releases. You can publish a test release with a very different set of properties and know that, if content creators check in some of those files, they probably won't break anything for folks still using the stable production tools (since the stable tools data is still there in the files).

Every field in the reflection information will specify a class of object that will handle the details of reading and writing the necessary data to a persistence interface or other objects of the same type. With this in mind, it's time to declare some data structures to store Class and Field information, as seen in Listing 2.

POPULATING THE DATA MODEL

➤ To help populate the data model, some template functions can help extract useful data via template parameters (see Listing 3).

A template function with parameters for

RESOURCES

Helium is an open source game engine toolkit that contains an implementation of C++ Reflection. Much of the code in this article was derived from it. It uses a BSD-style license, and is available at www.heliumpjproject.org. The reflection system itself is located in the Foundation/Reflect folder within the source repository.

- Fully hosted turnkey voice solution

SURROUND SOUND FOR GAMES VOICE CHAT

- Flexible API provides creative control

- Seamless integration and low-cost implementation

- Multilayer, 24/7 engineering support

Deliver an immersing and realistic voice experience for your game. Match voice communications to the game environment and allow your players to experience clear, realistic audio without clipping, echoing, or leveling issues.

With a conveniently flexible API, Dolby® Axon adapts to a multitude of game architectures for seamless integration and low-cost implementation—giving you more opportunities to explore and expand game-play options.

In addition, Dolby hosts servers and offers multilayer support and guaranteed uptime, eliminating the hassle and cost of maintaining server infrastructure. Highly scalable and bandwidth efficient, Dolby Axon is the most comprehensive choice available for in-game voice chat.

games@dolby.com

Dolby and the double-D symbol are registered trademarks of Dolby Laboratories.
© 2011 Dolby Laboratories, Inc. All rights reserved. 511/23768





the object type and variable type provides an easy way to extract the size of the variable and its offset from the base instance pointer (using a pointer to member variable), while also supporting the use of template specialization to deduce which type of data object is applicable to this field. Three important things are happening in this function to extract data for our reflection data model: pointer to member variable C++ syntax, translation of this syntax into an offset from a base object address, and the use of deduction using explicit specialization.

POINTER TO MEMBER VARIABLES

» Pointer to member variables are a pretty infrequently used aspect of C++. It does what you might expect, but its syntax is strange if you haven't seen it before:

```
int32_t Object::* pointer_to_member_variable =
&Object::m_Member;
// These are typically dereferenced with an instance of the object
// type (just like member function pointers):
Object object, *pointer = new Object;
int32_t value1 = object.*pointer_to_member_variable;
int32_t value2 = pointer->*pointer_to_member_variable;

// To compute the offset from a pointer to a member variable
template< class ObjectT, class DataT >
uint32_t GetFieldOffset( ObjectT DataT::* field )
{
    // a pointer-to-member is really just an offset value
    // disguised by the compiler
    return (uint32_t) (uintptr_t) &((ObjectT*)NULL)->*field;
}
```

This function doesn't bother with allocating an instance to dereference the pointer to member variable. It substitutes a NULL pointer, dereferences the pointer to member variable, and uses the address operator to yield the offset (from NULL) at which the pointed member exists. Some of this syntax may seem strange, but it's a perfect fit for maximizing what information is needed to describe a field in a single function parameter.

EXPLICIT SPECIALIZATION

» DeduceDataClass is a good example of template deduction using explicit template specialization. This deduction technique is a way of using the C++ template mechanism to allow for the automatic selection of some information by the template compiler based only on a template parameter. The default template function's implementation returns NULL, indicating that the deduction failed since no specialization was found to find the associated data, as below:

```
template< class DataT >
Class* DeduceDataClass()
{
    // unknown data!
    return NULL;
}
// Then create an explicit specialization for every type that can
// be deduced:
template<>
Class* DeduceDataClass<uint32_t>()
{
    // this specialization associates the uint32_t built in
    // type with an object class that can
    // process data of type uint32_t with respect to other
    // persistence / cloning / mining code
    return SimpleData< uint32_t >::s_Class;
}
```

LISTING 4

```
template< class ObjectT >
static Class* CreateClass( const char* name )
{
    Class* result = new Class( name );

    // populate the field information for this class
    ObjectT::Populate( *result );
    return result;
}

Finally, an example class and main that will put all of this code to
work:

class Foo
{
private:
    uint32_t m_Number;

public:
    static Populate( Class& c )
    {
        // AddField is a template function that will deduce
        // everything but what the desired name is.
        Field* numberField = c.AddField( &Foo::m_Number,
"Number" );
        // Its easy to imagine Field having extra
        // information to inform all sorts of program behavior
        // numberField->SetRange( 0, 10 );
        // numberField->SetCategory( Advanced Settings );
        // filePathField->SetFileFilter( *.png );
    }
};

void main()
{
    Registry::RegisterClass( CreateClass< Foo >( Foo ) );

    // program

    Registry::UnregisterClass( Registry::GetClass< Foo >() );
}
```

In this case, a pointer is returned to the class reflection information for the type of data object to be used when dealing with the built-in type passed into the template argument. One more template will help keep the code that registers classes at startup concise, as seen in Listing 4.

CONCLUSION

» Reflection can imbue an enormous amount of flexibility to your game engine, but this flexibility doesn't come without cost. However, the extra memory reflection data consumes is balanced by the time saved implementing features more rapidly. The ability to deliver changes to your users quickly, and with minimal engineering overhead, will pay dividends as your user base grows and your production time stretches across multiple titles. 🎮

GEOFF EVANS is a senior engineer at WhiteMoon Dreams. He was a founder of the Nocturnal Initiative open source project at Insomniac Games and is a founder of the Helium Project at WhiteMoon Dreams, which aims to build an open source commercial quality game engine. Contact him at geoff@heliumproject.org and follow him on Twitter @gorlak.

THE 2011
gametree™ TV
DEVELOPER COMPETITION



WIN GLORY
WIN \$50,000

ENROLL NOW AT [HTTP://GAMETREE.TV/COMPETITION](http://gametree.tv/competition)
OVER \$50,000 IN PRIZES | CELEBRITY JUDGES | SHOWCASE AT GDC
JOIN OUR GAME DEVELOPMENT SEMINAR AT GDC: MARCH 2ND, 3-4PM, ROOM 302





SIGNS OF LIFE

DUAL QUATERNIONS, A NEAT NEW SKINNING TECHNIQUE

IT'S THE BEGINNING OF THE YEAR, and tender new shoots of life are slowly stirring all around. Perhaps you're still actually going to the gym and watching your carb intake—2011 is young enough that anything is still possible. So, in the spirit of the moment, let's stop and look at the landscape of real-time graphics and see where the new sprouts of green are coming up.

Looking for interesting new bits of graphics tech is a harder business than it used to be. As we've chronicled before, the breakneck pace of game technology has slowed a lot in recent years. The current console generation started with a frantic arms race as we assimilated a host of unfamiliar tech, but in its sunset, things have slowed down a lot. Throw in a slower economy and the fact that what growth we do see is coming from underpowered platforms like mobile phones and lowest-common-denominator browser games, and you've got the makings of a graphics bust.

This doesn't mean that nothing interesting is going on, though. Instead, what's happening is a lot of long-standing sore spots are gradually being addressed. Our focus this month is a good example of some new tech that doesn't change the world, but can make a very positive difference in artists' lives.

Dual quaternion skinning ("DQS") has been around for a couple years in the academic world. This year both the Unreal Engine and CryEngine have added support for it, but it's still unfamiliar to many artists.

As the name implies, dual quaternion skinning is an alternative to the traditional skinning method we use to bind

meshes to skeletons. It's almost the perfect paradigm of a late-cycle technology, offering a refinement of an existing technology rather than a radical innovation. Nevertheless, it's got some really compelling advantages over the familiar system. All this means it's worth a closer look.

OUT WITH THE OLD

» The workings and limitations of traditional "linear" skinning techniques are pretty well understood, but a quick review is a good way to understand what the new dual quaternion technology does.

The conventional linear skinning technique is pretty simple. Every vertex is "bound" to one or more bones in the skeleton. As the skeleton moves, the vertices maintain their positions relative to the binding bones, moving along with the animation of the skeleton (see Figure 1).

The well-known drawback to linear skinning is the way it handles vertices that don't want to move rigidly with a single bone. Large changes in pose tend to produce unpleasant collapsing artifacts. Highly mobile joints like elbows and knees are especially susceptible, as are bones like the wrist and neck which twist around their own major axis.

These painful-looking artifacts come from the way the linear skinning interpolates the position of vertices that are bound to more than one bone. The algorithm is extremely simple: the skinning system or shader simply calculates the vertex position for every bone that affects the vert and then does a weighted average between these positions to get the final "smooth-skinned" result. Unfortunately,

that average position will always be closer to the joint than the un-averaged vertices, hence all those collapsing elbows and knees. That's why so many game characters have flat butts and sloped shoulders.

The customary fix for this behavior is to add extra bones that disguise the collapsing joints. These "fixup" joints have been a standard part of the rigger's toolkit for a long time now (we covered them back in the February, 2004 edition of *Game Developer*) and can be done with a variety of tricks. The most common fixups are done entirely inside of Max or Maya, using expressions or constraints to subdivide the problem rotations. Binding verts to these fixup bones produces a more reasonable behavior than the vanilla linear skinning. Just as important, riggers and character artists can add them in with no help from engineering.

Fixups have a long and honorable track record, but they come with significant memory cost. It only takes between 20 and 25 bones to completely describe a full-body pose (excluding fingers, toes, and facial bones). But a complete set of fixups may almost double that number, once you factor in things like twist bones in the biceps and forearms, and multiple graduated fixups for tricky areas like shoulders. Even a less complete set will be expensive, because each of those fixups means a complete set of animation frames. Even a minimal set of fixups on shoulders, elbows and wrists can add 20 percent to the overall animation cost of the character!

IN WITH THE NEW

» Here's where dual quaternion skinning comes in. Although the name is somewhat scary, the

idea is simple enough for artists. Where linear skinning blends vertex positions, the dual quaternion technique blends rotations. A vertex weighted equally between the bicep and forearm, for example, will rotate around the elbow joint as the forearm moves (see Figure 2). Essentially, this is how we've been using fixups for the last decade, only with no extra joints, expressions, or the weight painting hassles that come with fixup bones sitting exactly in the same place as major joints.

If it's all so simple, you might wonder why it's taken so long for this idea to appear. The description is, indeed, pretty straightforward, but the magic that happens behind the scenes to make it work does verge on rocket science (the mathematically fearless can check out the original SIGGRAPH paper that introduced the idea with a flurry of equations here: <http://isg.cs.tcd.ie/kavanl/papers/sdq-tog08.pdf>).

Luckily for us mere mortals, it's not necessary to understand how it works to appreciate the benefits. A quick glance at Figure 3 will show you the basic appeal of the dual quaternion technique. While it's still not a fancy Hollywood muscle system, it definitely sucks less. The key advancement is, of course, that the dual quaternion interpolation doesn't collapse around the joints. If you compare the width at the bend in Figure 2, you'll see that the volume of the linear skinned version has shrunk by about a quarter, while the DQS version has maintained its cross section nicely. The improvement is particularly pronounced for things like shoulders and wrists, which rotate on three axes at the same time as seen in Figure 2.



REALITY BITES

» Before getting too excited, remember that DQS is still, fundamentally, a very simple approximation of what happens when a living creature moves a limb. It doesn't create the really difficult, eye-catching effects like the bulge of a muscle or the way

skin slides over underlying bone. As we have already said, it's an improvement, but not a revolution. The example in Figure 3 is skinned with very smooth weights to clearly accentuate the limitations of DQS. The DQS version doesn't show the annoying shrinkage you see in the linear skinning

example. Instead, it shows an almost opposite behavior: the rotational blending above and below the bend actually cause the bent area to inflate a bit. Moreover, on the inside of the fold, both systems show the familiar pinching that comes from overly smooth weights inside an elbow or

knee, although to be fair, the DQS pinch is less obvious.

Luckily, these artifacts can be controlled with some attention to the vert weightings. By tightening up the weights on the outside, the bulging can be reduced to manageable proportions. It doesn't take much fiddling to get something that looks like the compression of a muscle rather than an irritating flaw in the math. (see Figure 4). On the inside of the bend, use the same kind of tricks you'd use for linear skinning: fairly tight weighting and a lower vertex density to hide the pinches. The result is still not going to look like *Avatar*, but it's a lot better than you could do in the old linear skinning without a lot of dedicated rigging.

As with any piece of art software, technical examples like these can give you an introduction, but only first-hand experience will tell how this technique suits your aesthetic. Fortunately, after several years as an almost mythical rumor, DQS has recently become easy to check out for yourself: Maya 2011 now supports dual quaternion skinning as a standard feature. You won't even need to re-weight an existing model—just open your character and flip a switch on the skin cluster node to see the difference between vanilla skinning and DQS. Softimage XSI also supports dual quaternions, and again allows you to toggle between the new method and the old. The current version of Max does not offer DQS support out of the box, but there are some free plugins floating around on the net [here, for example: <http://klaudius.free.fr/download.htm>].

TRADEOFFS

» Of course, this new ability doesn't come for free. Dual quaternion skinning requires a more complex vertex shader than the old-fashioned linear method. Implementations will vary, of course, but as a reference point, you might note that the dual quaternion shader introduced

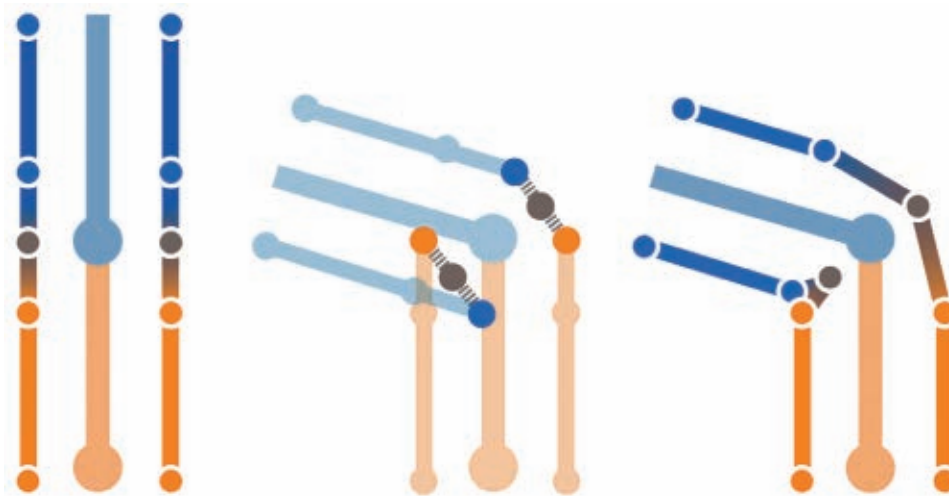


FIGURE 1 Why joints collapse: The brown vertices are weighted 50 percent to each bone. When the blue joint bends, the position of the brown vertices are calculated by averaging the position they would have had if they were attached solely to orange or blue bones. This produces the collapsing and pinching typical of linear skinning.

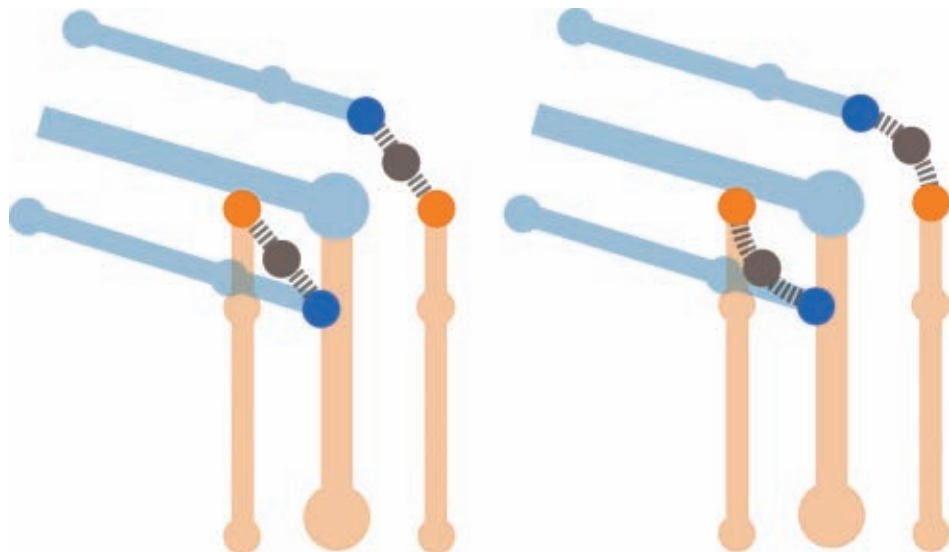


FIGURE 2 DQS-style interpolation: the vertices blend rotationally rather than linearly, so a vert weighted half to one bone and half to another doesn't collapse toward the joint.

in the original SIGGRAPH paper back in 2008 used 52 shader instructions, as compared to 36 for a conventional linear blend shader. Depending on the way your engine pushes animation data to the engine, there might also be a cost for converting animation data into the correct format.

At runtime, the choice between linear and dual quaternion skinning is really just the latest incarnation of the classic game technology trade-off between memory and processing power. DQS will consume more of the horsepower of your GPU for the same number of characters on-screen. However, it will relieve you from the necessity of animating the host of fixup bones you need to combat the limitations of linear skinning. If your game is heavy on sophisticated shader effects or has tons of vert-heavy characters on-screen, DQS may not be fast enough for what you need. On the other hand, if you're worried about your animation budget or the cost of pushing lots of bones, DQS can be a godsend: it delivers better looking visuals while cutting down on animation storage and decompression costs.

In the rare case where the question "Do we need more CPU or more memory?" doesn't

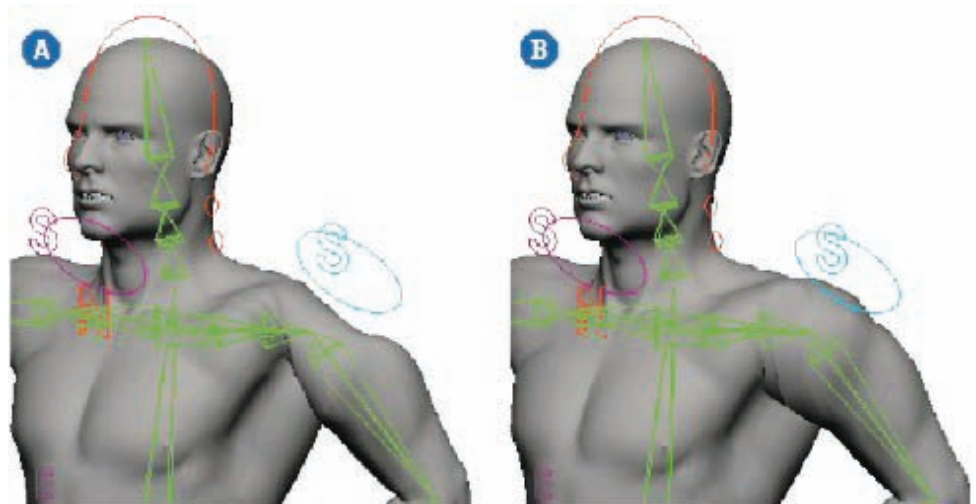


FIGURE 3 This example from the Softimage documentation clearly shows the advantages of DQS (right) for traditional problem areas such as shoulders. The traditional implementation is shown on the left.

answer itself, DQS offers one more important advantage. Fixup systems, though they are meat and potatoes for experienced riggers, do represent an ongoing support cost. A DQS-based pipeline will free up valuable tech artist time for other tasks, such as building tools or fetching coffee. Character rigs will be simpler, animation retargeting more robust, and vert binding more straightforward. It's rare for studios to pick technology based solely on the convenience factor

for artists, but if they did, DQS would be the preferred choice for most of us.

FUTURE'S SO BRIGHT ...

» That's a quick overview of what dual quaternion skinning does, and what it might do for you. It's a neat new approach, even though it won't set the world on fire. You can certainly expect to see more of it in the coming year. As with many graphics techs that have gone before, you can also expect to hear some

pretty cringe-worthy gushing from fanboys and marketing types. But hey, spring's around the corner and everybody has to get their jollies somehow. [@](#)

STEVE THEODORE has been pushing pixels for more than a dozen years. His credits include *MECH COMMANDER*, *HALF-LIFE*, *TEAM FORTRESS*, *COUNTER-STRIKE*, and *HALO 3*. He's been a modeler, animator, and technical artist, as well as a frequent speaker at industry conferences. He's currently the technical art director at Seattle's *Undead Labs*.

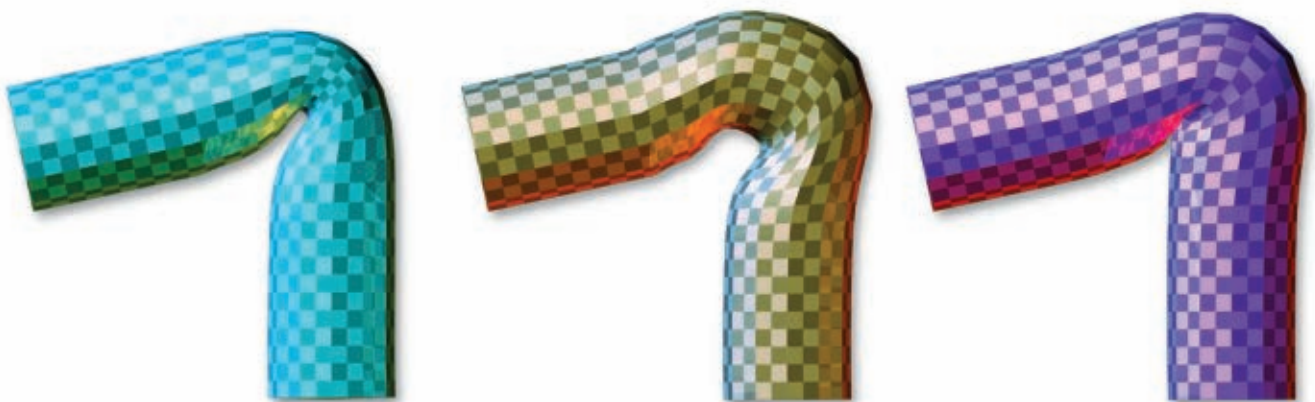


FIGURE 4 DQS is not perfect. Using the same weightings, the DQS technique (center) does not collapse like the linear skinned example (left), but it does exhibit an annoying bulge above and below the bend. With tighter weightings (right), the effect is fairly easy to control.



NARRATIVE AND PLAYER AGENCY

PLANNING FOR CHOICE

CONTINUING ON FROM LAST

month's column, there are many ways to leverage story in games, ranging from passive background information to being the primary driver throughout all the game content. However, quite often that story is passive; it is useful for guiding people through the content, but gives the player little avenue to actually change the flow of the story. Many great story games, such as UNCHARTED 2 and STARCRRAFT, present stories that the player might find deep and engaging, yet provide very little ability to make changes.

Some games try to go further, allowing players to make choices that shape and change the narrative. The patron saints for these games are, of course, tabletop RPGs like *Dungeons & Dragons*, where a room full of dice-rolling adventurers are free to take their quest in any direction they choose while a frantic dungeon master tries desperately to get them to the front door of the dungeon he brought to the table.

This freedom is a hallmark of these tabletop RPGs, and it should come as no surprise that developers try to bring the tabletop experience to life on the PC and consoles. I work at one of these companies today, and seeing BioWare put these games together up close has given me a new appreciation for the remarkable design intricacies involved in their construction.

HOW MUCH CHOICE?

»» How much choice can you actually give the player? In a tabletop game, players can go in any direction they want to seek content and narrative, with a Dungeon Master there to steer them in the direction of certain ideas and to decisions. In a tabletop game, the

answer may not necessarily be to save the princess or sacrifice her to save the village, but instead to find a middle road, an ingenious grey area where both might be possible. This kind of emergent problem solving provides some of the most magical moments of the tabletop experience.

In video games, this third path is likely one you can't afford to provide the player. Designers have to account for and predict all of the player's likely choices; if you provide a third door to open, there had better be content behind it. Each choice you allow the player to make is content that you have to create, support and run through QA. And when choices are stacked on other choices, you're creating a spider web of content, of which your average player will only see one path. Each mediocre or lukewarm choice you offer reduces the odds that players will see your best stuff.

And choice must be constrained for other reasons. In a tabletop game, players can feel free to kill the king that grants the quest because the gamemaster is there to rescue the narrative, perhaps spinning the story towards a lesser duke or rival who might have similar reasons to send the players into the Dungeon of Whatever. Providing this level of responsiveness, and still providing content delivery (cinematics, VO) at a high level of polish, is nearly impossible in the computer RPG.

INVALID CHOICES

»» In the tabletop game, the gamemaster could just let the game be broken, and declare the dungeon lost to the sands of time. The players chose to invalidate the quest, so let them, right? Players are less patient with electronic narrative that stalls because they

made some unfortunate choice, regardless of whether it was accidental. I suspect the \$45 dollar price tag might be a factor.

Most writers who aspire to join the ranks of BioWare understand that narrative choice is a cornerstone of our game design philosophy. Still, many applicants stumble with the pitfall of ensuring that all choices are valid. Letting the princess die or the village burn might be an unfortunate choice. It might have deep ramifications that affect the players further within the game, or require the player to perform supernatural acts to atone for his decisions. But a choice cannot leave the player's game in a broken state.

Even if this were somehow desirable, it wouldn't particularly be good game design. Sid Meier once said that a game is a series of interesting choices. Choosing between a viable story path and a narrative dead end is not a particularly interesting one, because there is so obviously a right choice to be made. If you let the princess die or the village burn, there must be value to that choice.

COMPARTMENTALIZE

»» Choice is hugely important, but it only really truly begins to shine once the player can see the results of their actions in-game. Choosing to save the princess and letting the village burn doesn't carry much weight if the village is still there after the fact. Seeing the results of your actions bear fruit is what truly makes these choices matter, whether it's inside the narrative (i.e., turning an enemy into an ally for the endgame) or through mechanics (such as when MASS EFFECT 2's conversations and quests increase your standing

with your companions). Otherwise, these choices are transparently inconsequential.

But having those reactions appear in the world can be tricky, especially when they can compound. Consider that you've completed three quests for the king, and in one, you saved his son, in another, you killed his treacherous daughter, and in a third, you erred and let a village burn to the ground. Even if each choice were only binary, there are eight possible outcomes the player could choose, which means even crafting a return conversation with the king is likely to be a heroic task. Writing dialogue when you've done one good thing, one terrible thing, and one tragic thing is almost entirely doomed to be a contradictory, muddled narrative mess—especially if it must be spliced together at runtime through a dialogue tree.

One way to work around this is to compartmentalize your big decisions. Instead of having one king grant all three quests with momentous decisions, divide them up between three nobles, each reacting strongly to the results of the quest of most interest to themselves but are relatively complacent about the troubles of his other nobles. One can see this in action in DRAGON AGE, for example: the elves are quite concerned about their own problems in the forest, and care only for how the player handles their woes. They care little about the player's interference with the likes of wizards or dwarves. The narrative is compartmentalized.

Too much compartmentalization can undermine the sense that the player's choice matters all that much if overdone, but it doesn't take all that much for the



designer to grant that sense of accomplishment. A couple of well-placed NPCs or the pronouncements of a town crier can create the sense that the player's actions have consequences. In *DRAGON AGE*, they go a step further by making the choices you make helping the elves, dwarves and wizards affect which allies you have by your side in the final confrontation.

PLAYER ALIGNMENT

» It's impossible for NPCs to react to every combination of choices that the player has made, especially if the design of these stories has been heavily compartmentalized. As such, many designs centering on narrative agency include a summary score that describes the player's actions so far. *Dungeons & Dragons* had alignment, of course, but it was meant to be more prescriptive than descriptive. Since

then, dozens of games have had similar systems.

The *ULTIMA* series is one early notable game—its virtue system tracked the actions of the player thus far. However, this system was not really about narrative agency—the player had little choice but to be virtuous in order to beat the game. Since then, though, many games (mostly RPGs) have attempted to provide some sort of moral choice, quantifying all choices combined into a single meter.

KNIGHTS OF THE OLD REPUBLIC, for example, gave the stark choice between the Light Side and the Dark Side that fans of the *Star Wars* movies would expect, but choices are frequently more interesting when they aren't strictly good or evil. *VAMPIRE: BLOODLINES* allows players to choose between embracing their humanity or their budding bestiality. *MASS EFFECT* allows the player

to choose between playing as a do-gooder Boy Scout, or a Jack Bauer renegade bent on getting the right thing done no matter the cost. *RED DEAD REDEMPTION* allows the player to choose between honor and dishonor.

These systems must be designed with care. Reaction to both sides of the meter must be given by the game or the NPC inhabitants within, or the system will be seen as a waste of time. On the other hand, if spiking one of these scores give tangible, powerful rewards of some sort, then players will likely choose to "game" the system, choosing distasteful choices for material gain (whether this is a good design result is one that will vary from game to game). Worse, if spiking a score gives strong, tangible, and important results, the question quickly arises as to how to quantify and reward the "grey" player who walks a middle path.

THE STORY THUS FAR

» If games truly are about providing meaningful choices to the player, then allowing players to determine the flow of the narrative itself is one of the most powerful and effective ways that games can truly claim to be interactive. Interactive fiction is not about merely allowing the player to make a couple of inconsequential choices inside a dialogue tree that all leads to the same place. Interactive storytelling should make players feel like they aren't just agents in the world, but actively making decisions that shape it. 🎮

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

Infrastructure. Optimized. Now that's a bright idea!

- ▶ Windows® and Linux Dedicated Servers
- ▶ High Availability & Low Latency
- ▶ 24 / 7 / 365 Live U.S.-based Support
- ▶ Unmetered Bandwidth
- ▶ 99.9% Uptime Guarantee
- ▶ SAS 70 Type II Certified

Optimize your gaming infrastructure at:
www.codero.com/gamers

1.866.2.CODERO



Infrastructure Optimized



The Brass Tacks

2011 GAME DEVELOPERS CONFERENCE REVEALS FULL-DAY TUTORIALS

The 2011 Game Developers Conference features a packed full-day tutorial lineup—including notables from Epic, Blizzard, and Valve—for the 25th edition of the industry's leading event for game creators. • These lower-capacity, first-come, first-served tutorials will once again be held alongside the GDC Summits on the first two days of the San Francisco-based event, Monday, February 28th and Tuesday, March 1st. • They will be open to those with a Summits & Tutorials or All-Access Pass, and interested parties can select their preference during the registration process. • GDC 2011 tutorials include the following:

GAME DESIGN WORKSHOP

This two-day tutorial, hosted by Mind Control Software's Marc LeBlanc, will explore the craft of game design by putting attendees in the thick of the process, offering hands-on exercises and group discussions to break down game design into its core elements.

Attendees will play and analyze games as well as take part in activities that task groups with adding features, fixing design flaws, and more, providing applicable ideas for the iterative game design process.

ADVANCED VISUAL EFFECTS WITH DIRECTX 11

Hosted by a collaboration of the industry's leading hardware and software developers, the "Advanced Visual Effects with DirectX 11" tutorial will teach attendees how they can create impressive PC game graphics using the Direct 3D technologies in DirectX 11.

The tutorial explores the details of rendering advanced real-time visual effects, and also covers a series of vendor-neutral optimizations developers should keep in mind when designing shaders and engines.

LEVEL DESIGN IN A DAY: BEST PRACTICES FROM THE BEST IN THE BUSINESS

Bringing back last year's highly rated tutorial, some of the most experienced AAA-game level

designers in the industry will offer an all-new line-up of best practices, lessons learned, interactive audience participation, and case studies.

Titles discussed will span BRINK to DEAD SPACE, and GEARS OF WAR to FALLOUT 3, with takeaway lessons for all those "responsible for crafting moment-to-moment gameplay." Speakers include Epic's Jim Brown, Zipper's Ed Byrne, Bethesda's Joel Burgess, Visceral's Matthias Worch, Splash Damage's Neil Alphonso, and more.

EMERGING ISSUES IN GAME DEV DEALS—BUILDING ON THE PRESENT AS WE CAREEN INTO THE FUTURE

In this extensive tutorial, attorney Jim Charne will cover the ever-changing technologies and platforms that developers should understand when making a game in today's market.

By examining the emergence of online platforms, this session will explore the new legal and management issues that arise when dealing with global audiences.

AUDIO BOOT CAMP

The ever-popular "Audio Boot Camp" offers an introduction to the complex and evolving audio industry, focusing on harnessing the tools and technology to best serve the needs of a title.

SCE Europe's Dan Bardino and Microsoft's Scott Seflon will

discuss various realms of audio development, from musical scores, ambience, sound effects and dialog, and how to use AI to dynamically blend these various elements into a cohesive package.

LEARN BETTER GAME WRITING IN A DAY

Marvel Comics editor and writer Evan Skolnick will present the fifth "Learn Better Game Writing in a Day" tutorial, incorporating lectures and exercises to help beginning and intermediate writers learn the basics of story structure, character development, and more.

The tutorial will feature examples from several games including RED DEAD REDEMPTION, MASS EFFECT 2, DRAGON AGE: ORIGINS, and MARVEL ULTIMATE ALLIANCE 2, on which Skolnick was the lead writer.

PHYSICS FOR PROGRAMMERS

The intense, technical "Physics for Programmers" tutorial brings together some of the best practitioners in gaming physics—from Insomniac and Blizzard, through Sony and Havok—to continue this storied session's 10-year-plus tradition.

Over the course of a day, attendees will "get up to speed in the latest techniques and deepen their knowledge in the topic of physical simulation." These presenters will provide a toolbox of techniques for programmers interested in creating physics

engines, with references and links for those looking for more information.

PRODUCER BOOT CAMP

A special one-day "Producer Boot Camp" is being assembled by key GDC Advisory Board members including Laura Fryer, VP and General Manager of WB Games Seattle, Epic Games executive producer Rod Fergusson, and Media Molecule's Siobhan Reddy (LITTLEBIGPLANET franchise).

The trio will assemble a full day tutorial—including themselves and other announced speakers—that "focuses on some of the key skills required by producers, both new to the role and seasoned veterans, to be successful in this challenging industry."

TECHNICAL ARTIST BOOT CAMP: LESSONS IN HOW TO CREATE AND BE AN EFFECTIVE TA

In another first, "Technical Artist Boot Camp" will cover the sometimes-neglected technical art area, offering insight from well-respected TAs from Volition, Valve, Blizzard, BioWare, and other top studios.

The presenters' hope is that by exposing academia, studio management, and displaced industry professionals to technical art, they will "foster discussion and expand educational and professional boundaries."

CANADA

CREATIVE COMPETITIVE CUTTING EDGE

INVESTINCANADA.COM



INVESTINCANADA.COM

At GDC 2011

Come visit us at
the Canada Business Lounge
3rd floor, W Hotel
March 1-3, 2011



Government
of Canada

Gouvernement
du Canada

Canada



EMBRACING RISK

HOW AVERSION TO RISK CAN COST YOU TIME AND MONEY

ASK ANYONE OVER THE AGE OF

30 how many times they've had to "learn something the hard way." Most people can't count that high. Businesses are similar in this regard: they need to experiment in order to gather the data that will enable executives to make informed decisions—and experimenting often means failing.

Despite this, most game publishers and developers are profoundly averse to experimentation and risk. "Little" mistakes, like failed prototypes, are not embraced. "Big" mistakes, like failed attempts to capitalize on new markets, are assiduously avoided until those new markets "prove" themselves, by which point it is deemed necessary to spend a fortune acquiring a successful competitor.

Dan Ariely, the author of *Predictably Irrational*, has noted that there's plenty of research to explain this behavior. In his own words: "Experiments require short-term losses for long-term gains. Companies [and people] are notoriously bad at making those trade-offs."^[1] Put another way, short-term risk aversion is a major psychological handicap for businesses—one worth recognizing and confronting.

Case in point: EA's \$300m to \$400m acquisition of Facebook game developer Playfish. Whether EA paid a fair price for Playfish is probably irrelevant. The company decided that it needed to get into the social gaming space, and Playfish was a good option (not to mention comparatively cheap, relative to Playdom and Zynga). The more interesting question is whether this acquisition was necessary.

BIG MONEY, BIG PRIZES?

» The first social games that really took off generally cost less than \$100k to initially develop. EA could have funded 10 independent, tiny social gaming studios working on such games, empowered them to experiment with new business

strategies and game designs, while spending a tiny fraction of Playfish's acquisition price. Assuming roughly \$2m in cost per studio, that's about 1/20th the price of Playfish. And don't forget that unlike other publishers, EA already had a pool of experienced casual game developers within its Pogo group that it could have tapped to seed this initiative. So why didn't EA do that?

Some might argue that it was impossible to know social gaming would become so popular, and thus, that it was worth investing in. So let's say that for every emergent opportunity on par with social gaming, another four that look similarly appealing turn out to be complete duds. Now the price of attempting to create the next Playfish has increased by fivefold. Which, by my admittedly rough estimate, still means it would have cost 1/4th the price of acquiring Playfish.

I don't mean to pick on EA; in many ways, it has been one of the most forward-thinking publishers in recent years. I'm trying to illustrate the fact that, contrary to popular wisdom, it may not be more cost effective for publishers to acquire innovative companies than it is to actually innovate. And when you consider the fact that many research studies have demonstrated that somewhere between 50 percent to 80 percent of all big acquisitions end up being viewed as failures for the acquiring entity^[2], it becomes clear that growth by acquisition is *not* a low-risk strategy.

The other justification I hear for M&A spending sprees is that internal innovation is simply too hard for big companies. They can't hire the right people. They can't adapt their development processes. They can't learn new tricks. And worst of all, they can't protect innovative teams from the politics and bureaucracy that tend to doom groundbreaking projects. These are unquestionably major challenges that I don't mean to trivialize. And

yet, given the astronomical cost of recent high-profile acquisitions, and given the odds that those acquisitions will look bad in hindsight, it's time to reevaluate the cons of organic growth.

A PROPOSAL FOR STUDIO INNOVATION

» So what's the best way to encourage internal innovation? (Kim Pallister gave an excellent lecture on this at the IGDA Leadership Forum.^[3]) Here's my take:


First: Given the perils of internal bureaucracy, new teams should be spun up in separate locations and treated as wholly independent studios, while still benefiting from certain shared resources like legal counsel and financial services. They should be tasked with seizing an opportunity but be given the flexibility to attack that opportunity however they wish, even if that means stumbling through a few relatively inexpensive failures. And they should be kept small, as in four to six people. It doesn't take an army to experiment in most emerging games markets.

Second: The initiative needs protection from the top. Otherwise, the mini-studios will be cannibalized the instant a "more important" project comes along. It is not beneath a CEO or senior vice president to make this a priority, no less than deciding to greenlight a half-a-billion-dollar acquisition.

Third: The initiative needs to be overseen by a small group of people who understand that they are managing a portfolio of high-risk investments. It is not only likely but a given that a significant percentage of those investments will not pan out. In other words, preventing failure is not the key goal. Supporting promising new experiments and helping the mini-studios share learnings with each other is the goal.

This issue is not only relevant to large companies. Indie developers may not have EA's resources, but that doesn't mean they can't adopt a portfolio strategy. My studio, Spry Fox, amounts to just 18 people in total when you include partners and contractors. But as of the time of this writing, we have five F2P games in simultaneous development, with five completely independent, tiny teams working on them. Each team is experimenting with original game designs and/or new business strategies, and each team is fully aware that the experiments they are conducting may not ultimately be successful.

It is possible that all our projects will fail. But if we succeed, we'll have accomplished what very few large companies in our industry have been able to: a true portfolio process for developing innovative, original IP within new markets. I look forward to sharing the results of our efforts, be they successful or not, in my upcoming columns.

In the meantime, I invite you to ask yourself a question the next time you're weighing the pros and cons of conducting a business or game design experiment: "Am I focused on all the ways the experiment could go wrong, or am I focused on how to make the experiment as efficient and educational as possible?" 

[1] <http://donariely.com/2010/04/10/column-why-businesses-don%E2%80%99t-experiment>

[2] www.examiner.com/mergers-and-acquisitions-in-jackson/why-do-most-acquisitions-fail-to-add-value

[3] <http://blip.tv/file/4350642>

DAVID EDERY is the manager of the consulting firm Fuzbi and CEO of the Spry Fox game development studio. He is also an IGDA board member and a research affiliate of the MIT Comparative Media Studies Program. He was the portfolio manager for Microsoft's Xbox Live Arcade service and is the co-author of *Changing the Game: How Video Games are Transforming the Future of Business*.



LEVEL UP

MAKING THE MOST OUT OF THE GDC CAREER PAVILION

BY MATHEW KUMAR

2010 SAW THE GAMES INDUSTRY

in a period of transition. The rise of social gaming, new motion-control inputs, the promise of 3D, digital downloads, and cloud-gaming are not only challenging developers as they plan for the future, but also offer great potential to those looking to expand and grow their career by taking advantage of their established skills in new opportunities.

As a result, the Career Pavilion at the 25th Game Developers Conference—held this year on Wednesday, March 2nd to Friday, March 4th in Moscone South Hall (accessible with all GDC passes, including Expo or Student passes)—offers an ideal chance for developers, both established and new, to meet with recruiting studios and publishers.

We've talked to some of the top recruiters, from companies including Insomniac, Ubisoft, and Sony, to ask them what they're looking for, and with that knowledge at hand, we hope we can help you focus your search to discover a new role in this ever-changing industry.

BE THE BEST YOU CAN BE

» With an industry in transition, we were fascinated (though not surprised) to find that the recruiters we surveyed had a wide variety of differing needs they were looking to fill within a range of positions. One requirement was consistent, though: the need for experienced senior staff.

Despite industry upheaval, "the fundamentals haven't changed," said a recruiter for Ubisoft, whom is attempting to fill roles globally for franchises such as *SPLINTER CELL* (being developed in Toronto, Canada), *ASSASSIN'S CREED* (being developed in Malmo, Sweden; Annecy, France; and Sofia, Bulgaria,

as well as Montreal, Canada), and *RAYMAN: ORIGINS* (being developed in Montpellier, France.). "We have a larger proportion of positions that require previous multiplayer or related online experience," they explained, "but we are always looking for talented senior professionals who can bring something to the group. After all, the quality of our games reflects the quality of the teams behind them."

Angela Baker, a recruiter for Insomniac Games (currently working on titles including *RATCHET & CLANK: ALL 4 ONE* and *RESISTANCE 3*) said, "I think we [like everyone else] are always looking for senior talent to come in and really help solidify an already great team. While we may not have dozens of openings, the ones we have are fantastic opportunities to make a huge impact on the games we're making right now."

Of course, simple experience is not enough to stand out in a marketplace like GDC, with surveyed recruiters hoping that potential applicants can ensure that the "huge impact" they make is a positive one.

Ryan MacDougall, a recruiter for Capcom Studios Vancouver (néé Blue Castle Games, developer of *DEAD RISING 2*) said, "Over the last couple of years there have been numerous layoffs within our 'recession-proof' industry. The market is saturated with people with previous games industry experience, but many companies have had to shift to a different kind of model: leaner and more flexible; able to accomplish more with less."

As a result, the overwhelming advice from recruiters was, as in the words of Karen Chelini, Sony Computer Entertainment's Director of Talent Acquisition, "Be the best that you can be at where you are in your career."

"We need people who have high growth potential," she said, "While we may need to fill a current role, ideally we look for people who can move up and contribute at a higher level."

MEET BOTH YOUR NEEDS

» Amongst other new developments, the growth of social gaming has changed the playing field for game developers, with many recruiters looking to fill positions that offer different perks from traditional game development, such as alternative work-life balance via more flexible development schedules.

One such example is BigPoint, a social game developer that continuously develops its titles such as casual-focused *FARMERAMA* or shooter *RUINED ONLINE*. A BigPoint recruiter explains the company isn't about "meeting specific deadlines for putting a game in a box and getting it on a store shelf."

However, the company still requires traditional game development chops. "Skilled engineers (Unity, C++, C#, PHP, and Flash) and 3D artists are at the top of our list," he said, though with the expectation that prospective developers would recognize the "unique experience" of developing online, free-to-play browser games.

That's not to say that "traditional" game developers aren't keenly aware of the importance of work-life balance to potential hires—especially those with experience in the industry. Insomniac's Baker opined, "I think 'quality of life' means something different for every employee—and we certainly offer flexibility in response to that."

"It can mean working on a stellar title to someone, or getting

five weeks of vacation to another. One of our concerted efforts for everyone is to manage schedules and the scope of projects, so that the evil 'crunch' is minimal and short in duration. This is something that we work on every project, and have really seen positive results."

Ubisoft's recruiter agreed with the challenge in dealing with "crunch" mode and noted, "Each of our studios tries to adapt itself to its local reality, meaning both in terms of geographic location and studio size. For example, our Montreal studio offers an on-site daycare, and when it comes to having to put in extra hours during the final stretch of production, most studios offer specific post-project time-off so that people have the necessary time to relax with their friends and family."

Just as with examining if you can fit into a studio in a role that will meet your current needs, it pays to research if the companies you are interested in can offer you the kind of life and career you are aiming for. Some studios attempt to offer a "flat" structure—MacDougall offers that Capcom Studios Vancouver is "proud" that "everyone has a voice to speak their opinions and pitch ideas," while Baker says Insomniac "look[s] for ways for employees to grow within their jobs, not just up and out of them,"—while others offer more of a traditional hierarchy. At Ubisoft, it's "important to offer a clear career path," according to the company's recruiter, so "people have visibility and realize that they achieve their career goals within the group. We are able to not only offer a classic growth trajectory, but also propose mobility opportunities abroad or between projects. And of course, we try to convey the multitude of jobs that are open to

someone as they advance in their career in function of the type of path they wish to take and the evolution of their interests.”

PLAN AND PREPARE

» Whether you are fresh out of school, a developer who has suffered in a round of layoffs, or even just looking for a change, one of the main challenges facing you at the Career Pavilion will be standing out to recruiters as the best in your chosen field. While some of the tips our recruiters offer might seem like common sense, what is surprising is how often they claim they get overlooked:

Resume: Unless you are Shigeru Miyamoto himself, you aren't going to get anywhere without a stack of paper resumes to hand out (and no excuses if you think you're the next Miyamoto). Keep it clean (spell check!), tight (no more than one two-sided sheet), and ensure it highlights the very best of your career.

Portfolio: Even with multiple completed titles under your belt, it isn't enough to list them on your resume and expect that to show your potential. Delano Lobman, HR Manager for Guerilla Games (currently working on KILLZONE 3) states that they need to be “knocked off [their] feet” by the quality of art portfolios or code samples provided. Established developers need to be able to provide samples that show their importance to completed projects, and developers who have been out of work—even if only for a short time—would do well to show that they've kept their skills intact and up-to-date with samples of work created during the downtime.

Insomniac's Baker emphasized, “We see a lot of people.” Standing out “without being creepy” can be hard, she admitted, but offered an eye-catching tip: keep your best work to hand for quick demonstration. “With the advancement of portable tech these days, it's super cool to check out a reel in real time. The best way to stand out is to have a solid portfolio



GDC PHOTOGRAPHY BY VINCENT DIAMANTE

that you can show either digitally or in a book, or have a snippet of a game that people can play.”

Research: We can't emphasize this enough. Every recruiter we speak to is full of tales of meetings with job seekers that not only don't seem to know what their company does, but don't even know what they would like to do as part of it. Being familiar with a company's titles and where their studio location is a good start, said Ubisoft's recruiter, but being able to recognize a company's needs and sell yourself to them is key. “Check out developers' careers/recruitment pages and see what open positions and projects they are recruiting for; find out if you have experience in related platforms, technologies, or genres. And know if you are willing to relocate.”

Insomniac's Baker agrees, stating furthermore that successful applicants should be able to clearly state why they want to work for a company—and it is for more than just a job.

Don't overshoot, however. Capcom Studios Vancouver's MacDougall argues that while you must aim to be the best in your field, you should also recognize your current skill level. “People

applying for roles out of their reach, such as graduates applying for lead programmer or art director, happens far too often. Companies track that kind of thing: you are remembered for the roles that you applied to.”

The Early Bird Catches The Worm: Insomniac's Barker notes that “everyone in the booth is fresher at 10:00 a.m.,” so you're likely to make a bigger impression—and beat out those still nursing hangovers—if you turn up early. Indeed, one anonymous recruiter warned applicants away from trying recruitment booths towards the end of the day. “People have been on their feet all day and won't give you the time you might get otherwise,” they said.

Be Respectful: Get to the pavilion on time, but don't skip a shower to do it! While having fully researched companies that you speak to might seem to cover it otherwise, Guerilla Games' Lobman offers three important don'ts: “Don't act like you're 'all that,' and don't instantly talk about the salary we could offer.” The last one is potentially most important to established developers who might be nursing bad feelings over previous jobs: “Don't trash other

studios or former employers. It's very unprofessional.”

Follow Up: Even if you've wowed everyone you've spoken to with your portfolio and sparkling professional manner, it's essential to stay in contact with the companies you are most interested in—even if it's just to receive feedback on your meeting and portfolio. And don't be disheartened if the meeting didn't go as well as you hoped. Insomniac's Baker offers, “Just because we spend 5 minutes with you, doesn't mean we're not interested in following up with you. It just means that the line behind you has grown and we need to chat with other people. It's all good. So don't be offended!”

GO FOR IT!

» The Career Pavilion is an exhausting and packed three days not just for job-seekers, but recruiters too. By placing your best foot forward through research, preparation, and organization, you can make it easier not only for yourself, but for the very people you are trying to impress. Good luck! 🍀

MATHEW KUMAR is a freelance journalist based in Toronto and a contributing editor at Gamasutra.com.

Programmers & 3D Artists:
Are you the trailblazer
for our new frontier?

Friendly and small-feeling
\$1B company (on NYSE)
HIRING experienced
team members full-time
for our Naples, Florida studio

Apply online: dreamtools.com

... oh and the answer to your first question for us ... Ninjas.

DreamTools 

If it doesn't look real, we didn't make it.

© 2011 Simpson-Siringo-Tie-Company Inc. DTGAMEDEVAD

MAX YOUR EDGE WORKING AT WMS

**NOW
HIRING**

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

-Lead in the creation and design of video and reel-spinning slot machines

-Innovative designs and games that will lead the gaming industry into the future

-Join a company experiencing exponential growth

-Enjoy a fun work environment which allows you to continually learn and grow

WMSTM
MAX YOUR EDGE

Chicago . Las Vegas . Reno . London . Sydney

COME VISIT OUR BOOTH
CP 2326

www.wms.com



working 9 to 5

MARIEL CARTWRIGHT COMES IN OUT OF THE COLD

Marriel Cartwright has always lived the freelance life, working non-stop as an artist and animator for games like SCOTT PILGRIM in Montreal and BATMAN: THE BRAVE AND THE BOLD. As part of the art collective mechafetus.com, which includes cult indie artists such as Paul Robertson and Jonathan Kim, she developed her odd hybrid style of anime and Western illustration. Most recently, she's joined a new team in Los Angeles—her first office job—as lead animator on an upcoming original 2D fighting game for consoles.

BRANDON SHEFFIELD: After working on the critically acclaimed SCOTT PILGRIM game, how did you decide where to go next? Was location a factor?

MARIEL CARTWRIGHT: The project I'm currently on was started by a good friend of mine, and I had already been helping with animation on it while I was working on SCOTT PILGRIM. After SCOTT PILGRIM wrapped up, it made sense to move my focus onto this project. I was kinda worried about having to move back to LA for this job; I was working freelance and traveling a fair bit, but I figure I had to settle down eventually and I didn't want to pass up being part of this team.

BS: What about this project appealed to you as an artist?

MC: The style of the game lends itself to my own style really easily, and I'm able to take a bit of creative direction and be a big part of the project. I feel like I'm getting to work on something I would've volunteered to be a part of anyway, but it's still something that forces me to think a bit differently and work outside my comfort zone, so I'm learning a lot as well.

BS: How has the transition from artist to lead animator been? Was it at all intimidating? Do you have to manage outside contractors as well?

MC: It's not only a transition to lead, it's actually my first time working in an office too. I was totally nervous before it started. Being freelance meant I could work when I wanted and live anywhere, and I didn't like the idea of settling down for an office job. It's been okay though! It turns out I wasn't very good at scheduling my freelance work and felt like I was always crunching, so having hours makes things easier. I do have to manage contractors a bit, but our in-house team is small enough that we all provide some feedback to our outside artists. I just write more e-mails is all.

BS: You're now in charge of people you've worked alongside and known for years, how does that affect the working dynamic?

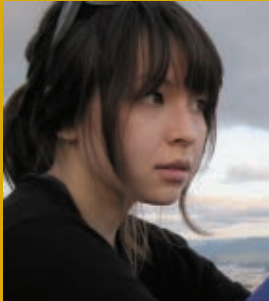
MC: It's been fine. I think because it's people I've worked with before, we all know what we're capable of and it makes work pretty efficient. I don't think technically being in charge has made a drastic difference ... because our team is so small everything is totally collaborative, being lead just means I have to manage a few more things.

BS: The popularity of 2D fighters fell a lot in the early aughts, but has recently been bolstered by SFIV, BLAZBLUE, and others. Do you see the 2D fighting revolution continuing? And if so, what will push the genre forward on the art side?

MC: I'm definitely not an expert on fighters, but I don't think 2D fighters are going to disappear anytime soon—the community supporting them is huge. On the other hand, I think fighters are still fairly niche compared to more mainstream genres. I'd love to see titles that are able to push the power of 2D with cool designs and fluid, fun animation that don't cater exclusively to niche audiences that many existing fighters do, and push things in ways that 3D can't.

BS: For a long time you have been part of the art collective Mechafetus. Would you say that association changed your art style or development practices at all? If so, how?

MC: Sure, I think I've taken a lot of influences from my friends. They've introduced me to a lot of weird, interesting stuff that I probably wouldn't have discovered otherwise. I think the past few years of knowing them has helped me settle into the cute/gross style that people seem to know me for. Of course, I'm always hoping to push my stuff even further just on my own, but being part of that team of sorts definitely impacted how my style developed.



new studios

Outsourcing firm Streamline Studios, which has worked on content for titles including GEARS OF WAR, GHOST RECON, and SAINTS ROW, announced it has founded a new studio in Kuala Lumpur, Malaysia.

Five months after Amsterdam-based game publisher PlayLogic filed for bankruptcy, the company said it would relaunch with "a focus on digital video game publishing" for consoles.

who went where

Industry veteran Graeme Devine has left a post at Apple's iPhone Game Technologies Division, where he oversaw the company's iOS gaming strategy, to pursue his passion for game development.

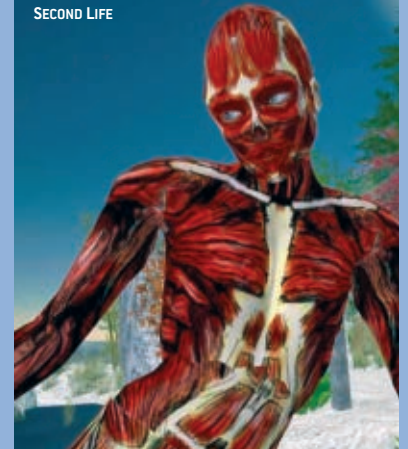
After departing from Codemasters in 2010, game industry veteran Stuart Black, senior designer on the 2006 Criterion shooter BLACK, will head up a new London-based in-house development studio at City Interactive.

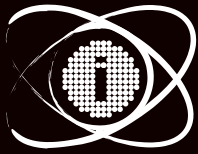
The PC Gaming Alliance named Intel's Matt Ployhar as its new president, as the two-year-old group continues its efforts to promote and advocate the PC as a gaming platform.

Former EA vice president Rod Humble has taken the reigns as CEO of SECOND LIFE developer Linden Labs after the departure of Mark Kingdon in 2010.

Atari's worldwide CEO Jeff Lapin has left the company, and deputy CEO Jim Wilson will expand his role to fill Lapin's position.

SECOND LIFE





THE 13TH ANNUAL
INDEPENDENT
GAMES FESTIVAL

AWARDS

WEDNESDAY, MARCH 2, 2011 • 6:30PM – 8:30PM

SAN FRANCISCO MOSCONE CONVENTION CENTER, HALL D



IGF AWARD CATEGORIES

IGF MAIN COMPETITION

- » Seamus McNally Grand Prize
- » Excellence In Design
- » Excellence In Visual Art
- » Excellence In Audio
- » Technical Excellence
- » Audience Award
- » Best Mobile Game

IGF STUDENT SHOWCASE

- » Student Showcase Finalist
- » Best Student Game

NUOVO AWARD



CELEBRATING OVER 600 INNOVATIVE GAMES ACROSS THIS
YEAR'S MAIN, STUDENT, AND NUOVO AWARD COMPETITIONS

VIEW THIS YEAR'S SUBMISSIONS AT WWW.IGF.COM

PLAY THE FINALISTS AT THE IGF PAVILION ON THE GDC 2011 EXPO FLOOR, MARCH 2-4, 2011

PLATINUM SPONSOR



GOLD SPONSOR



PLATFORM SPONSOR





chaos invaders

CHAOS INVADERS

<http://bitbattalion.com/games/chaos-invaders>

WHILE SIMULTANEOUSLY STUDYING FOR A COMPUTER SCIENCE DEGREE SASH MACKINNON HAS BEEN RELEASING SPEED-INFUSED GAMES THAT ARE A MODERN RETURN TO THE ARCADE ACTION OF YOUTH. HERE HE SHARES SOME TIPS FOR GETTING HIGH PERFORMANCE OUT OF FLASH.

Jeffrey Fleming: *How did you arrive at the core experience of CHAOS INVADERS?*

Sash MacKinnon: The core experience of CHAOS INVADERS is chaos, so I put a lot of work into making the game frantic, adrenaline filled, fast-paced, and visceral. Things like music and sound effects, the timer, large amounts of enemies on the screen, and even the user interface were all designed specifically to create this experience.

Deciding on this direction was actually a bit tricky. I had the idea to remake SPACE INVADERS, and within a few days had knocked out a prototype with all the mechanics in place, but it wasn't fun.

Originally I had planned on making it a slow-paced game with the emphasis on growing your ship into a huge tower that would reach out to new, larger enemies. I'd even thought of putting in some RPG mechanics based on the types of enemies you collected. Even with these mechanics in place, it lacked the stickiness that the final game has; I didn't know how I wanted the player to feel while playing it.

After I stumbled upon this idea of chaos, the game was instantly brought into a new light. I decided I wanted the player to feel excited, stressed, frantic, and above all, powerful. I wanted them to breathe a sigh of relief after every level. After deciding on that direction, ideas just poured out and CHAOS INVADERS was born. This struggle really taught me that mechanics alone often aren't enough to make a game fun. To make the game really appealing, it helps to know what experience you want the player to have while playing and build around that.

JF: *What has your game education experience been like?*

SM: I'm doing a degree of Computer Science at University of New South Wales and it's been great. I don't

encourage people to go in expecting they'll be taught exactly how to become an amazing designer, but it's really helped to give me the tools and knowledge to start out. It's also put me in contact with some really helpful people.

That being said, all the games I've made have been done individually, so my knowledge of game design and development has been mostly self taught. I've found that learning game design through courses and books can be great for table talk, but no matter how much you learn, when you really get stuck in the nitty-gritty decision making of designing your first game, it's going to be hard! There is a lot to be learned through experiencing the creative process personally. Formal education can build on this, but not replace it.

JF: *One thing your games seem to have in common is a real sense of speed.*

SM: Speed is something which really resonates with me as a gamer, but pulling that kind of performance out of ActionScript is tough as nails.

On my current project, MR RUNNER 2, I've really been pushing AS to its limits. I've got in-game motion blurs on a 700x400 screen, lots of particle effects and lots of layers. The truth is it's really hard to keep the frame rate up in Flash; you have to optimize every little bit of code you write and think things through to the nth degree. As if this weren't enough, there will always be computers which are slower than the one you're testing on. It can be a bit nightmarish, actually!

There are two really important techniques that I use when trying to bang out high performance. First, object pool everything that is created and destroyed more than a few times, especially if it is instantiating an embedded graphic of .swf; they have a tendency to memory leak and slow things

down pretty badly (there's a great object pooling explanation and library at: <http://lab.polygonaal.de/index.php?s=objectPool>).

Secondly, use libraries wherever you can. TweenMax for tweening and delayed calls, Box2D for physics, AS3DS for any tricky data structures, Away3D for 3D. They're all so useful and will save you a whole bunch of work. They'll also speed up your project quite significantly.

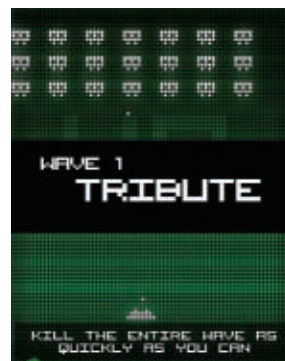
JF: *In screenshots, CHAOS INVADERS has a deceptively simple graphical look, but in play, the screen comes alive with motion blurs and transparency effects.*

SM: All of those effects—the blurs, glows, and transparencies, are actually built into AS3. The only trick is getting them to run fast enough to still have a playable game. There was one big trick which I used to speed CHAOS INVADERS up, and that was to use bitmap blitting (which means working with individual pixels) and to scale the game up five times. This achieved the pixel style that you see in game, but also meant that effects you see weren't very CPU intensive since they had to do five times less calculation! This really let me go wild with these effects, which were all part of the frantic experience.

The funny thing about this is when you look at my "enemy.png" files, they're all so tiny you can hardly see them. They look a lot less ominous at their original size!

JF: *What advice do you have for someone who is considering trying to make some money as an indie game developer?*

SM: Aiming to make money can be a bit overwhelming. Often, it puts a whole lot of pressure on the developer to go in directions which they wouldn't otherwise. That kind of pressure can really limit creativity. My advice is to make a game which you personally would love to play, and then do everything



you can to make it as unique and fun as possible. Don't worry too much about time or money, just stick with it until the end. If you have a good game, by the end of the long haul, you'll make money off of it.

The other important lesson I've learned is that the amount of revenue is unpredictable. CHAOS INVADERS made me almost 10 times less than MR RUNNER. I've heard of amazing games getting very poor sponsorships, and small games getting amazing sponsorships. It can be a little bit daunting.

If you're trying to penetrate the Flash game market, I would highly recommend FlashGameLicense.com, which is essentially eBay for Flash games. It allows sponsors from all around the web to view your game and bid on it. It's great for people who don't have connections in the industry just yet.

Even if all this seems overwhelming, don't lose hope! Despite these issues, developing Flash games part time has funded my education and social life so far throughout my time at the university, and has been an experience I would have paid for myself. Finally, my most important piece of advice is to just make a game! Get something out there! And if you're doing it for the first time, start small.

—Jeffrey Fleming



THE 11TH ANNUAL

game DEVELOPERS CHOICE awards

The **Game Developers Choice Awards** are the premier accolades for peer-recognition in the digital games industry. Every year at GDC, the Choice Awards recognize and celebrate the creativity, artistry and technological genius of the finest developers and games created in the last year.

AWARD FINALISTS WILL BE ANNOUNCED IN FEBRUARY.
Stay updated at: www.gamechoiceawards.com.

AWARDS ARE PRESENTED IN THE FOLLOWING CATEGORIES:

2010 AWARD CATEGORIES

- Best Audio
- Best Debut
- Best Downloadable Game
- Best Game Design
- Best Handheld Game
- Best Technology
- Best Visual Arts
- Best Writing
- Best New Social/Online Game
- Innovation
- Game of the Year

SPECIAL AWARDS CATEGORIES

- Lifetime Achievement
- Pioneer
- Ambassador

PRESENTED BY

Game Developers
Conference

PRODUCED AND HOSTED BY



gamedeveloper





ONLY THREE LITTLE THINGS

MANAGING THE NEEDS OF MULTIPLE PROJECTS

HERE AT OBSIDIAN

Entertainment, we have a relatively small audio department of five people who are tasked with making very large games. Being that we're an independent developer with a centralized audio department, we're also constantly juggling multiple projects and supporting various needs. Handling all this can be a bit overwhelming. In order to manage multiple deadlines, minimize crunch time, and keep our games sounding great, we have had to change the way we scope our projects, task the audio team, and review our work. Since implementing these changes, we've been able to keep a high level of quality while managing our workload and keeping a good work/life balance.

SCOPE: THE BIG PICTURE

» When deadlines are looming and the pressure is on, I find it useful to step back and do a quick gut-check to assess our progress on each project. It all boils down to three little things: music, sound effects, and dialogue. While this may seem like an oversimplification, viewing it in this way can sometimes bring some sanity to the chaos of game development. I know that if these three things are covered, we're in good shape. If not, I can help divert resources to the area that is lacking.

Normally, when one of these areas is in poor shape, it's because it is

out of scope. Obsidian makes very large games with up to a hundred hours of gameplay. Because of this, scope is of constant concern. With the constraints of time, money, and quality being immovable, our only course of action is to manage scope. We do this in two ways: limiting features and variations of sounds, and by using templates.

For *FALLOUT: NEW VEGAS*, we relied on templates to make a first pass layout of music and ambient sound for the game. With so many locations, we had to generalize ambiance into categories like rural house, bunker, vault, and so forth, while for music, we categorized by emotions, such as creepy, peaceful, rural, and mystical. Doing this allowed us to first cover the scope of the game, and then allowed us to go back and refine those areas that were truly important with a unique music and ambient pass.

TASKING AND DEVELOPMENT: GETTING IT DONE

» Let me first say that I am not a fan of bug tracking software for tasking on a project. Having to deal with strict formatting while managing priorities seems to hinder productivity more than help. We have found that working off of simple lists, generated through group playthroughs, are much more effective. When in a creative mindset, it is important to keep focused



Obsidian's *FALLOUT: NEW VEGAS*

and minimize clutter. By keeping these lists simple and deleting the items once they have been reviewed, we keep a clear and concise list of what needs to be worked on, which allows a certain level of creative freedom. However, since these lists are not tracked, it is especially important to have a strong follow-through to make sure everything is getting done.

When it comes to scheduling, we've also found that simple is good. I tend to only schedule the project we're going to be working on, and what milestone goals we're trying to achieve. We have found that strict schedules that attempt to manage specific workloads and assets on a day to day basis are almost always out of date as soon as they're written. Game development is such a fluid process that I have found it much more useful to assign areas of ownership to each

of the team members and allow them the autonomy to make their own schedules. This level of ownership allows everyone to feel that their contribution is unique and important, and it shows in the quality of their work.

QUALITY CONTROL: DOES IT SOUND GOOD?

» It all comes down to that simple question: Does it sound good? I believe the only way to say "yes" to this question is to constantly iterate on a project—to play it, and refine it again and again. At Obsidian, the audio team gets together and constantly plays through our games and makes refinements. This process is fairly time intensive, but doing so is the only way to keep everyone on the same page. It's also good for the morale of the team because it allows us to see the progress that's being made on a day-to-day basis. This format also allows for a lot of

peer feedback, and we are all open and honest about our criticisms and responses.

When working on an individual system in a game, it can be easy to get lost in the specifics of what you are doing and lose perspective of the project as a whole. Maybe that footstep sound you are working on in your DAW sounds amazing, but in the game, it doesn't quite fit into the soundscape. By stepping back and viewing the project as a whole, you regain some of that perspective. That is where all the individual elements of a project come together as a whole. The music, the sound effects, and the dialogue all come together to form the complete picture. As long as these three little things are alright, you're all good! 🎧

SCOTT LAWLOR is the audio director at Obsidian Entertainment.



Vancouver Film School

Game Design at Vancouver Film School is an intense one-year program that covers everything you need to join the game industry as a designer or producer, from theory to hands-on practice to the production of a professional-quality portfolio. There's a reason why the *L.A. Times* called VFS one of the top 10 schools "favored by video game industry recruiters."

VFS Game Design students learn more than just one side of game design - they experience the full scope of this varied and rewarding career through an in-depth curriculum that includes:

- » Interactive Narrative
- » Analog Games
- » Interface Design
- » Scripting
- » Level Design
- » Pre-Production
- » Project Management
- » Flash
- » Mobile & Handheld Design
- » Game Audio
- » The Business of Games

// LED BY INDUSTRY

In VFS Game Design, you're mentored by a faculty of respected industry pros – your first crucial connections to the professional world. At the helm is veteran Dave Warfield, who, as a Senior Producer for EA, helped produce and design the *NHL*

franchise for 10 years. His many other credits include titles like EA's *NBA Live* and Konami's *Teenage Mutant Ninja Turtles*. An Advisory Board of industry leaders, including luminaries from Irrational Games (2K Boston), Microsoft, Nokia, and Ubisoft, keeps the curriculum on the cutting edge.

// A STUDIO ENVIRONMENT

In a process that closely mirrors a real-world studio environment and production pipeline, you work in teams to take games from concept to completion. Toward the end of your year at VFS, you get the chance to present your final playable games to an audience of industry representatives and recruiters: a unique chance to prove yourself and make valuable professional contacts.

// LIVING & CREATING IN VANCOUVER

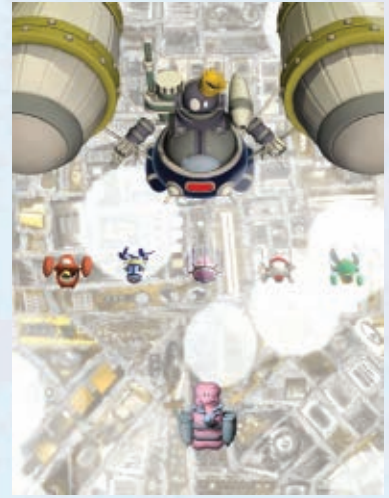
In VFS Game Design, you have the advantage of learning in Vancouver, B.C., Canada. Along

with strong its film, TV, and animation industries, Vancouver is a world center of game development, meaning that VFS is always industry-current, hosts many guest speakers, and provides you with vital mentorship and feedback opportunities throughout your year. It's the perfect place to get your career started.

// THE RESULTS

Our graduates have gone on to earn key design and production roles at top studios around the world. A small selection of their recent and upcoming titles includes: *Warhammer 40,000: Dawn of War II*, *Marvel Ultimate Alliance 2*, *Prototype*, *Dragon Age: Origins*, *Punch-Out!!*, *Mass Effect 2*, *FIFA 10*, *Skate 3*, *True Crime*, *Dead Space 2*, *Star Wars: The Old Republic*, *Dead Rising 2*, *Pirates of the Caribbean: Armada of the Damned*, and *ModNation Racers*.

Find out about VFS Game Design and begin your career at vfs.com/gamecareer.



"It was a really mind-opening experience for me, in terms of what is possible creatively. VFS was really instrumental in me being successful today."

— Armando Troisi, VFS Graduate
Lead Cinematic Designer
Mass Effect franchise



VFS VANCOUVER FILM SCHOOL
Results Matter

CONTACT INFO

Vancouver Film School
200-198 West Hastings St
Vancouver, BC V6B 1H2
Canada
604.685.5808 or 800.661.4104
inquiries@vfs.com

www.vfs.com/gamecareer



Game face. Market focus.

Careers in Applications Development

Crunching vast amounts of data in lightening speed. Designing flashy graphic user interfaces. Exploring experimental hardware that's never seen the light of day. Developing algorithms that will automate millions of decisions. Jumping in and building solutions from the ground up. Test it. Support it. Beat the competition. Change the game...Sound familiar?

This is a career in Applications Development at Citi. Here, you'll apply skills you already have and learn things you never imagined, all in a collaborative team environment that supports you to go even further.

Apply to Citi's Applications Development opportunities today at: careers.citigroup.com

The career you want - the benefits you expect!
An Equal Opportunity Employer M/F/D/V.



Ahead of the Game.

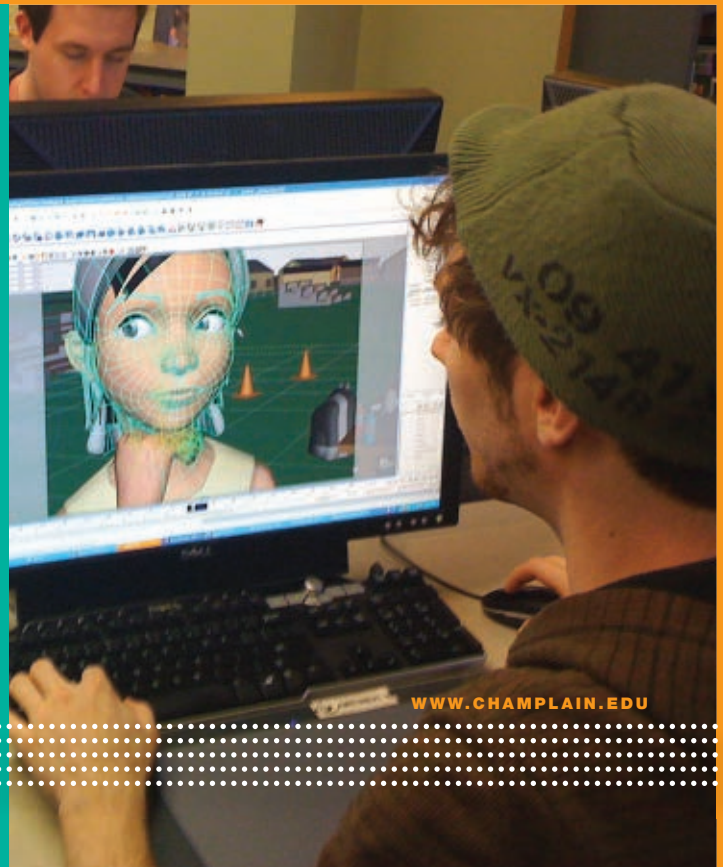
The video game industry is booming, and gaming companies are struggling to employ highly skilled developers and designers to keep up with this explosive demand. Led by an expert team of seasoned faculty, Champlain College students acquire a complete set of technical skills while refining their creative abilities.

Champlain College's unique team-based structure spans multiple programs to merge video game entertainment with emergent immersive technologies, 2D and 3D art, animation, storytelling, and interface design—providing their students with real-world experience—reflective of the best practices the most successful game and interactive media companies use today.



CHAMPLAIN
COLLEGE

BURLINGTON, VERMONT | (800) 570-5858



WWW.CHAMPLAIN.EDU



Screen shot from 2011 Senior Capstone Project, *The Warden of Ro'al*

He'd like to talk to you about majoring in **Game Design.**

Columbia College Chicago's Interactive Arts and Media Department offers majors in:

**Interactive Arts & Media
Game Design**

with concentrations in:

**Game Art
Game Development
Programming
Sound Design**

game.colum.edu

Columbia

COLLEGE CHICAGO

INNOVATION IN THE VISUAL, PERFORMING,
MEDIA, AND COMMUNICATION ARTS

create...
change

INVEST IN YOURSELF. BECOME A MASTER OF DIGITAL MEDIA.

We offer a 20-month Master's program in entertainment technology and digital media that combines industry-facing curriculum, real-world projects and a four-month internship. The MDM degree is jointly awarded by four leading Canadian post-secondary institutions: The University of British Columbia, Simon Fraser University, Emily Carr University of Art + Design, and the British Columbia Institute of Technology.

Our students come from around the globe, and from diverse undergraduate and professional backgrounds – including media and fine arts, computing science, natural and social sciences, business, and engineering.

Come work and play with us in Vancouver BC: Canada's video game capital.

For more information about the Masters of Digital Media (MDM) Program, go to mdm.gnwc.ca

VISIT US AT GDC IN SAN FRANCISCO | FEB 28 – MAR 4 2011

**CENTRE FOR
DIGITAL MEDIA**

Great Northern Way Campus



JOIN OUR NUMBERS



14 AWARD-WINNING
STUDENT GAMES

191 ALUMNI WORKING AT
42 COMPANIES AROUND THE WORLD

\$50,852 AVERAGE STARTING SALARY
FOR GRADS

3 TRACKS TO CHOOSE FROM
PRODUCTION, ART & PROGRAMMING

16 MONTHS TO GET YOUR ACCREDITED
MASTER'S DEGREE IN GAMING

Art from student game 9 Lives 'Til Midnight

Florida Interactive Entertainment Academy

At the University of Central Florida
Graduate Game Development Program
Orlando, Florida



407-823-2121 | www.fiea.ucf.edu |  

Game Art & Animation

Associate's Degree

Careers Include:

- Animator
- Modeler
- Technical Animator
- Level Designer

Program Highlights:

- Motion Capture facilities
- Utilizing: Unreal III Engine, Maya, Motion Builder, Mudbox, Body Paint, & more

Additional Emphasis:

- Story development
- Performance
- Cinematography
- Traditional art
- Color theory

Launch your career today!

Madison: 800.236.4997 Minneapolis: 877.416.2783



MADISON: 2702 Agriculture Drive | Madison, WI | madisonmedia.edu MINNEAPOLIS: 4100 West 76th St. | Edina, MN | minneapolismediainstitute.com

TURN YOUR PASSION FOR GAMING INTO A CAREER

Game Art

Bachelor's Degree Program
Campus & Online

Game Design

Master's Degree Program
Campus

Game Development

Bachelor's Degree Program
Campus

Game Design

Bachelor's Degree Program
Online



Campus Degrees

Master's

- Entertainment Business
- ▶ Game Design

Bachelor's

- Computer Animation
- Digital Arts & Design
- Entertainment Business
- Film
- ▶ Game Art
- ▶ Game Development
- Music Business
- Recording Arts
- Show Production
- Web Design & Development

Associate's

- Graphic Design
- Recording Engineering

Online Degrees

Master's

- Creative Writing
- Education Media Design & Technology
- Entertainment Business
- Internet Marketing
- Media Design
- New Media Journalism

Bachelor's

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



FULL SAIL UNIVERSITY

fullsail.edu

Winter Park, FL

800.226.7625 • 3300 University Boulevard

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

NO PANTS? NO PROBLEM

When you take online classes from **FUTUREPOLY**, you'll learn how to create professional video game art from the leading artists in the industry, all from the comfort of your own home.

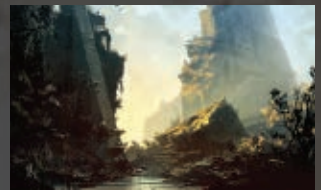
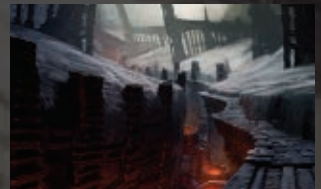
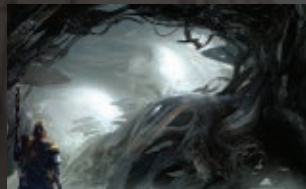
Although, truthfully, the instructors would probably feel more comfortable if you at least pretended to have pants on.



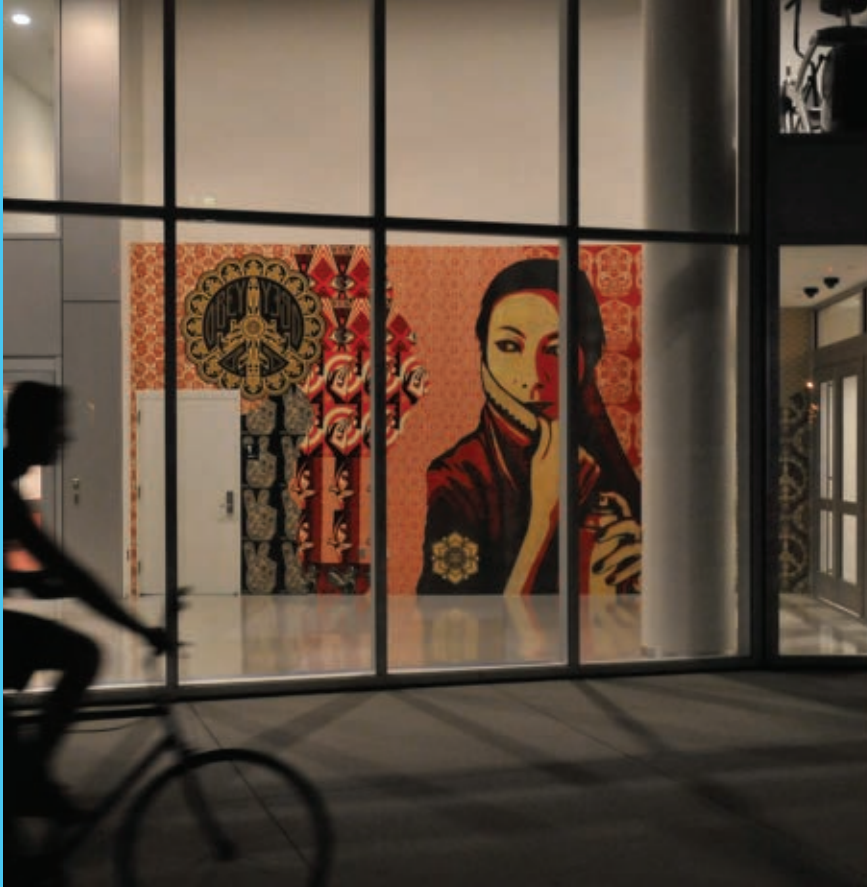
FUTUREPOLY | RELEVANT EDUCATION

FEATURED INSTRUCTORS

DANIEL DOCIU | RICHARD ANDERSON | LEVI HOPKINS | MATTHEW BARRETT
KEKAI KOTAKI | THOMAS SCHOLES | JAMES KEI | HORIA DOCIU



Visit www.futurepoly.com to register.



Game Design and Interactive Media in the heart of Boston

Our Creative Industries program strives to create, foster, and implement digital media innovation through research, education, and ground-breaking team-based interdisciplinary projects.

Collaborate and create with one of seven combined majors, or the Creative Industries minor.

For more information, go to:
www.ci.neu.edu



Northeastern

CREATIVE INDUSTRIES
 Northeastern University
 360 Huntington Avenue
 Boston, MA 02115-5000

le cnam enjmin

The graduate school of games and interactive media

The graduate school of game and interactive media welcomes you in France !

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



March the 22th
 Subject available online
www.enjmin.fr

- ENJMIN
- 121, rue de Bordeaux
- 16 000 ANGOULEME Cedex FRANCE
- Phone : +33(0)5 45 38 65 68
- Fax : +33(0)5 45 35 65 66
- contact@enjmin.fr

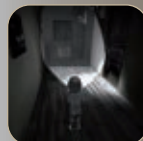
Get THE Game Master Degree in Europe !

Come to see us at the GDC 2011, booth #2641, South Hall

Summer language training on the Atlantic coast



- Software design and development
- Sound and Music Design
- Project management
- Game Design
- Ergonomics
- Visual Design



MAKE MORE ENEMIES

Game Design at VFS shows you how to make more enemies, better levels, and tighter industry connections.

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

Our grads' recent credits include *Mass Effect 2*, *ModNation Racers*, and *Dead Rising 2*. The LA Times named VFS a top school "most favored by video game industry recruiters."



VFS student work by Maximilian-Gordon Vogt, completed for the EvolveCG hunter contest.

GET EDUCATED



VFS Find out more.
vfs.com/enemies

"The staff at VFS provided a foot in the door that gave me an opportunity to prove myself."

ARMANDO TROISI | GAME DESIGN GRADUATE
LEAD CINEMATIC DESIGNER, *MASS EFFECT 2*

ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE	COMPANY NAME	PAGE
6 Waves	31	Full Sail Real World Education	76	Rad Game Tools	C4
Allegorithmic	50	FuturePoly	77	Riot Games	40
Blizzard Entertainment	32	Gaikai	18-19	Simpson Strong-Tie	66
Champlain College	73	GeoEye	3	TechExcel	14
Citigroup	73	Government of Newfoundland & Labrador	6	TransGaming	54
Codero	60	Havok	10	Unity Workshop	45
Columbia College	74	Insomniac Games	23	University of Central Florida	75
Course Technology	C3	Invest In Canada Bureau	62	University of Pennsylvania	39
Crytek	46	Madison Media Institute	76	Vancouver Film School	72, 79
DeVry University	27	Masters of Digital Media	74	VSoft Technologies	42
Dolby Laboratories	52	NaturalMotion	36	WMS Gaming	66
ENJMIN	78	Northeastern University	78	Zoo Entertainment	48
Epic Games	13	Perforce Software	C2		

gd Game Developer Magazine (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. 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 *gd Game Developer Magazine*/United Business Media LLC, unless otherwise indicated. Don't steal any of it.



WELCOME TO OUR OPERATION

CAN YOU SPOT THE WARNING SIGNS?

EXCUSE ME, I HAVEN'T SEEN YOU AROUND BEFORE. ARE YOU...? OH, MAN, please tell Johnny I'll have the money real soon now—wait a second, I'm sorry. You're the new art director, aren't you? Ha, sorry about that. I had you confused with ... well, nevermind all that. Welcome to our company! Thanks for showing up; we weren't totally sure you would. No, I'm just kidding.

It's great to have you on board! Getting your artistic abilities onto our team is really going to go a long way toward keeping our investors happy with our progress. I mean, our game is totally awesome in all other respects, but those money guys are just so dumb! They just don't get it unless there's slick art to gawk at, you know?

Anyway, that's where you come in and why I'm thrilled that you're joining our team today. Come on in. Let me give you the tour and get you set up. Now, here's the office—as you can see, lots of room to grow. We're still in the midst of a, uh, a re-shuffling of sorts, just to better deal with changing project requirements and so on. So it's a bit scattered right now. And we still need to fix that window. It broke one day, just like that. Weird, huh?

And here's your desk! Hm, it looks like Bob left some things. Let me just take care of that quickly—Liz, could you put this stuff in a box and leave it outside the door and tell him to come by and grab it? Thanks.

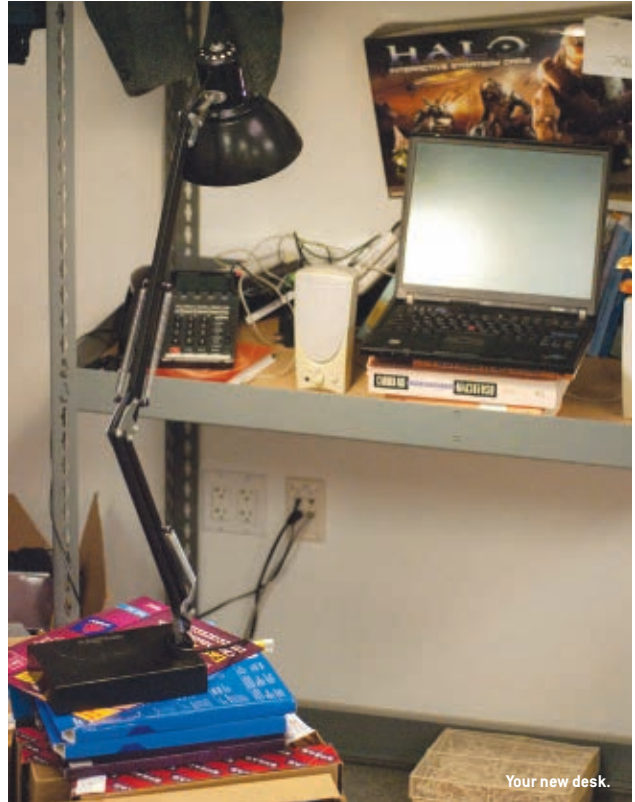
This is your computer: our standard workstations here are top-of-the-line, brand-new netbooks running Linux. Now, I see the look on your face there, but I guarantee you'll be surprised; I know I was! Netbooks are serious computers these days. They can easily run all of your standard high-powered game development applications: Microsoft PowerPoint, the Internet, and so on. Of course, if you need anything extra for your netbook in order to do your job, such as a mouse, you can put in a requisition to the IT department. That trackpad should be good for most everyone, though.

"Photoshop," you say? I've never heard of that. Are you sure you need this? We aren't asking you to take photographs. Is there something that this "Photoshop" does that the software included with the netbook doesn't do? Well, I might be able to swing you a copy ... I'll have to take a look at the balance sheet for this quarter and check with some, uh, sources.

For now, let me get you set up with the project. One thing to note is that we don't use source control here. I don't believe in it. We keep the latest version of everything on a shared folder on my machine. Just type in the password—it's "coolgame2"—and you should see the project directory. It's the folder called "Second Prototype."

We also have a dev kit, just like the big boys! I've been using it to keep my monitor at the right height. I can't leave it with you because that would be unfair to the rest of the guys, you know? But you can e-mail me some instructions on how to use it, and I could test out the art for you. I'll do that thing where I paste the art into that window that runs the game—that's how it works, right? And then I'll tell you, "It's good," or "It's no good," and you can take that information and work on the art some more. I know you won't need to do that too often, though, since you're such an awesome artist.

Alrighty, let's boot up the game here. As you can see, we're still in a pretty early state. We're still trying to figure out if it'll be an FPS, an RTS, an MMO, or what! What we want to do is create a hybrid of all those genres: basically, take everything good about each and put them together and leave out all of the bad parts. Everything you see here is completely temporary. But as you can see, we've got objects loading up and being displayed, you can move the camera around, and so on. We're still working on support for texture mapping, but I'm pretty sure that part is on its way, at least as soon as we can get the frame rate up. I know it looks kind of



primitive right now, but that's what you're here for: you're going to take this and make it look amazing.

Well, that's it for now, since I've gotta run to a meeting with our investors—they said they wanted to have an urgent meeting just yesterday night! No notice or warning or anything ... no idea what that's all about. But whatever. I'll leave you here and check back later.

In the meantime, you can, uh, well, you can browse the project wiki. There's a design doc in there that you should read, but it's really old and totally out of date. There's also some technical documentation that could be useful, but I think most of what we've done has changed around since that was written, since the original author of that document is no longer with us. Um, and there's also a "meet the team" page with photos of some of the team, but I've been kind of bad about keeping that current.

With that, I'll leave you to—what's that? A chair? You mean you don't want to work standing up? Ha, that was a joke. Actually, some people do work standing up. It's more ergonomic and better for your body. I've seen a lot of standing desks at some of the bigger game studios. But yeah, we'll get you a chair. I mean, if that's really how you want to work. Give me like, 10, 15 minutes to have this quick meeting with the investors, and we'll get that chair thing solved just as soon as I'm done, okay? Welcome to the team! 🍷

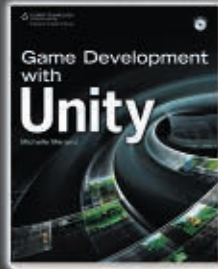
MATTHEW WASTELAND writes about games and game development at his blog, *Magical Wasteland* (www.magicalwasteland.com).

Attending GDC 2011? Visit the GDC bookstore to Purchase These Titles and Others!

YOUR ULTIMATE RESOURCE



**Beginning Java SE 6
Game Programming, Third Edition**
1-4354-5808-7 • \$34.99



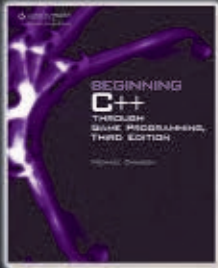
Game Development with Unity
1-4354-5658-0 • \$39.99



**Visual Basic Game Programming
for Teens, Third Edition**
1-4354-5810-9 • \$34.99



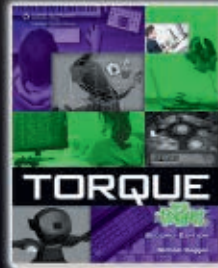
PSP Game Creation for Teens
1-4354-5784-6 • \$34.99



**Beginning C++ Through
Game Programming, Third Edition**
1-4354-5742-0 • \$34.99



**Multi-Threaded
Game Engine Design**
1-4354-5417-0 • \$59.99



Torque for Teens, Second Edition
1-4354-5642-4 • \$34.99



**iPhone 3D Game Programming
All in One**
1-4354-5478-2 • \$39.99



Introduction to Game AI
1-59863-998-6 • \$39.99



**C# Game Programming:
For Serious Game Creation**
1-4354-5556-8 • \$49.99



Video Game Optimization
1-59863-435-6 • \$39.99



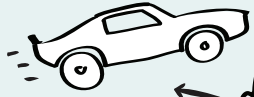
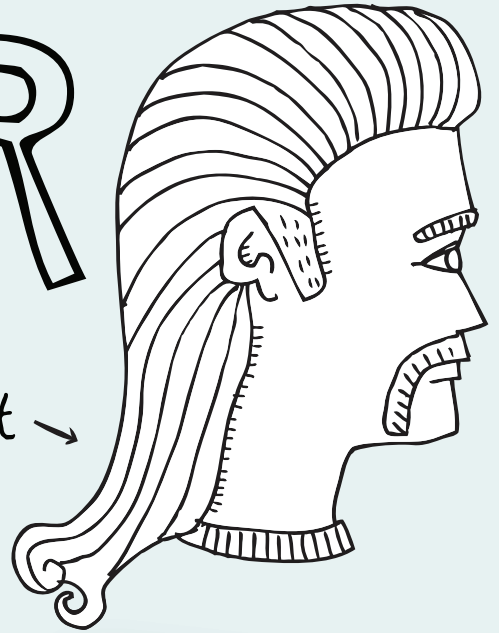
Game Programming Gems 8
1-58450-702-0 • \$69.99

GDC 2011 attendees, be sure to visit the GDC bookstore to browse and purchase our comprehensive list of game development titles.

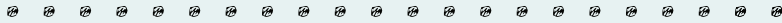
Available from Amazon, Barnes & Noble, Borders, and other retailers nationwide and online. Visit us online at courseptr.com or call 1.800.354.9706.

THERE ARE LOTS OF WAYS TO BE AN EVEN

COOLER GAME DEVELOPER



like grow a mullet →
← drive a sweet old Camaro
or you can use **BINK VIDEO**.



You get an amazing, super **FAST**
video and audio codec - all in a
simple, clean API. And Bink Video
runs on **EVERY PLATFORM!**



USING BINK VIDEO NOT ONLY MAKES YOU
cooler, IT MAKES YOU **rad!**



www.radgametools.com
(425) 893-4300