



POSTMORTEM: SONY ONLINE ENTERTAINMENT'S FREE REALMS

VOL17NO4APRIL2010

# gamedeveloper

THE LEADING GAME INDUSTRY MAGAZINE

## 9TH ANNUAL \$ALARY SURVEY



# **Scaleform**®

#1 Video Game User Interface Solution



## Smokin' Fast UI

Create highly optimized Flash® UI with Scaleform GfX 3.1

Use the Scaleform AMP profiler to tune runtime memory and performance

Improve HUD performance by ~10x with direct access to Flash objects

Download the new SDK and HUD Starter Kit to learn more



[www.scaleform.com/ui](http://www.scaleform.com/ui)





## POSTMORTEM

- 20 SONY ONLINE ENTERTAINMENT'S FREE REALMS**  
Despite years of experience building MMOs, SOE found itself in uncharted territory when it decided to create a kid-friendly, casual virtual world. Old assumptions had to be cast aside and new methodologies adopted to create a game that young players would take to heart.  
*By Laralyn McWilliams*

## FEATURES

- 7 9TH ANNUAL GAME DEVELOPER SALARY SURVEY**  
How much is your job worth? For our annual survey we crunched the numbers across discipline, experience, gender, and region, as well as new stats for the independent developer sector.  
*By Brandon Sheffield and Jeffrey Fleming*
- 14 THE THREADS THAT BIND US**  
Current languages in use by game programmers were not created with multithreaded concurrency in mind. Erlang, a language built from the ground up for concurrency by the telecommunications industry, may offer a solution.  
*By Nicholas Vining*
- 28 INTERVIEW: DAVID CRANE**  
As one of the original Gang of Four who left Atari to form Activision, David Crane helped lay the foundation for the game industry that we know today. We caught up with the die-hard coder and found him at work on iPhone development.  
*By Brandon Sheffield*
- 33 LESSONS FROM DOOM**  
DOOM has an enduring legacy in the game industry. Here, BioShock 2's Jean-Paul LeBreton looks at the venerable title to see what lessons it can teach modern game designers.  
*By Jean-Paul LeBreton*

## DEPARTMENTS

- 2 GAME PLAN** *By Brandon Sheffield* [EDITORIAL]  
Ethical Choices
- 4 HEADS UP DISPLAY** [NEWS]  
Indie Fund details, IGDA board elections, and 2600 MAGIC.
- 36 TOOL BOX** *By Michael Greenhut* [REVIEW]  
FlashDevelop 3.0.6
- 38 THE INNER PRODUCT** *By Jari Komppa* [PROGRAMMING]  
Porting From DOS To Windows
- 42 PIXEL PUSHER** *By Steve Theodore* [ART]  
Going Solo
- 44 DESIGN OF THE TIMES** *By Damion Schubert* [DESIGN]  
The Truth of Consequences
- 46 AURAL FIXATION** *By Vincent Diamante* [SOUND]  
Subversive Audio Design
- 48 GOOD JOB!** [CAREER]  
Richard Garriott interview and new studios.
- 52 EDUCATED PLAY** [EDUCATION]  
Focus on UCF's GALACTIC ARMS RACE
- 56 ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]  
The Genius Game Designer



# ETHICAL CHOICES

MAKING DECISIONS MATTER IN "MORALITY"-ORIENTED GAMES

## SOMETHING SOREN JOHNSON

wrote in his March design column re-ignited a spark that's been smoldering in my brain for some time now. Most ethical choices in games are not much of a choice at all.

He was discussing BIOSHOCK, and how you're given the option of harvesting the innocent(ish) Little Sisters for their Adam, which is used as currency in the game, or saving them, for which you get much less Adam. But after saving a few Little Sisters, you get a huge package of Adam, worth about what you would've gotten if you'd harvested them. Thus, your reward is simply deferred, and the choice ultimately isn't an ethical one, it's "do I want to be a jerk."

Many games that pose different ethical choices have this problem. One of the worst offenders for me was INFAMOUS. The story simply did not support ethical choices being made. The world of INFAMOUS takes place in a quarantined city. The first choice you're presented comes when some food is airdropped in—you can choose to harm the other citizens and take all the food for yourself, or only take what you need, and share with them, risking going hungry later.

The trouble is, food is not a currency, it does not determine health, and it has no bearing on the actual world of play. In this case, there is nothing to the question but "do you want to be mean to these people." There's no upside for you, no tradeoff. The next ethics scenario boils down to "I can help these people get away from the guards [who are bullying them senselessly] or let them die." In INFAMOUS, these "moral" choices are simply a means to an end—they break up the skill tree by only allowing "good" players certain skills, and "evil" players other skills. It also changes the attitude of the anonymous populace toward you, and which NPCs you can

get missions from. It only affects gameplay to the extent of choosing one of two paths. Ultimately the choices are hollow.

## I CHOO-CHOO-CHOOSE YOU

» I truly believe that if one is going to present choices or issues in games as ethical, those choices have to matter in the game world. But I get antsy when games present me with choices that clearly open one door while closing another, as I want to see all of the game's content, since I'm unlikely to go through it multiple times [Damion Schubert talks about this in his column on page 44].

As an example, in DRAGON AGE you have the option of siding with the golems or a rogue blacksmith in a particular scenario. Depending on who you go with, you will either get a squad of golems to command in your final battle, or the one golem in your party will leave forever (which is a ballsy move, incidentally, considering that character is DLC). This kind of choice makes me very uncomfortable, because I want to do what's going to be best for my one playthrough, and weighing those odds is very difficult, given that there are unknown variables (not having fought the final battle yet, I didn't know if I needed golems).

## METHODS OF CHOICE

» DRAGON AGE and MASS EFFECT have similar but subtly different ways of representing your moral choices, but both do it directly with numbers, which is controversial, as there are no "renegade points" in real life. In MASS EFFECT, you get points in one column or the other for your actions being good or evil, so to speak. In DRAGON AGE, members of your party will approve or disapprove of your actions depending on how the characters are designed. They will also react to your decision with dialog.

I far prefer the latter method. DRAGON AGE poses your choices as

affecting your teammates' opinions of you, and if their opinion is low enough, they will leave the party. In MASS EFFECT, while characters may approve or disapprove, it has much less in-game relevance. Furthermore, DRAGON AGE's choices are one-to-one. Your character speaks exactly what you select. In MASS EFFECT, you choose a summary of what your character will say in dialog, choosing the top option for "good," middle for "neutral," and bottom for "evil." The simplistic approach is far less interesting than the larger case of party approval.

A third solution can be found in FALLOUT 3. There, the choices you make in dialog certainly change how characters react to you, and it does fall into the trap of "say something nice" versus "be a jerk just because," but the more important choices are in what you do, not what you say. You have the option of blowing up the first town you come to, Megaton, by detonating a bomb there. A suspicious fellow urges you to, offering promises of riches and the key to the elite Tenpenny Towers if you do. The choice here is clear, but not intimidating. If you choose to blow up the town, most of the quests and shops are still available to you in a different form. If you choose not to, you still get to visit Tenpenny Towers later, albeit in an antagonistic way. Shoot a resident of a town and expect the rest to turn on you. Steal from them, and expect the same. It's simple, but it works as immediate feedback for a clearly moral choice.

I think moral choices are incredibly interesting territory for games, but they really do need to be integrated into the gameplay and story both. You can't just tell a player these choices affect the world, or that they're important. You have to show that as true, and you have to make them believe it.

—Brandon Sheffield

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

## SUBSCRIPTION SERVICES

FOR INFORMATION, ORDER QUESTIONS, AND ADDRESS CHANGES

t: 800.250.2429 f: 847.763.9606  
e: [gamedeveloper@halldata.com](mailto:gamedeveloper@halldata.com)

## EDITORIAL

### PUBLISHER

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

### EDITOR-IN-CHIEF

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

### PRODUCTION EDITOR

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

### ART DIRECTOR

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

### CONTRIBUTING EDITORS

Jesse Harlin  
Steve Theodore  
Daniel Nelson  
Soren Johnson  
Damion Schubert

### ADVISORY BOARD

Hal Barwood Designer-at-Large  
Mick West Independent  
Brad Bulkeley Neversoft  
Clinton Keith Independent  
Bijan Forutanpour Sony Online Entertainment  
Mark DeLaura Independent  
Carey Chico Pandemic Studios

## ADVERTISING SALES

### GLOBAL SALES DIRECTOR

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

### MEDIA ACCOUNT MANAGER

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

### GLOBAL ACCOUNT MANAGER, RECRUITMENT

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

### GLOBAL ACCOUNT MANAGER, EDUCATION

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

## ADVERTISING PRODUCTION

### PRODUCTION MANAGER

Pete C. Scibilia | [peter.scibilia@ubm.com](mailto:peter.scibilia@ubm.com)  
t: 516-562-5134

## REPRINTS

### WRIGHT'S REPRINTS

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

## THINK SERVICES

CEO UBM THINK SERVICES Philip Chapnick  
GROUP DIRECTOR Kathy Schoback  
CREATIVE DIRECTOR Cliff Scorso  
CHIEF INFORMATION OFFICER Anthony Adams

## AUDIENCE DEVELOPMENT

### TYSON ASSOCIATES Elaine Tyson

e: [tysonassoc@aol.com](mailto:tysonassoc@aol.com)  
LIST RENTAL Merit Direct LLC t: 914.368.1000

## MARKETING

### MARKETING SPECIALIST Mellisa Andrade

e: [mandrade@think-services.com](mailto:mandrade@think-services.com)

## UBM TECHNOLOGY MANAGEMENT

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





# I WANT YOU



# TO CREATE GAMES



WWW.WARIOWAREDIY.COM

## YOU CAN WIN THE WARIO™ AWARD!

The Wario™ Award celebrates your creativity. Create your own games, compete with others and have the chance to win a trip to the exclusive Nintendo E3 Media Briefing in L.A. You can enter by creating and submitting a microgame with the new WarioWare™: D.I.Y. game, or by submitting a microgame design concept via the website. Challenge your creativity and learn more at [www.WarioWareDIY.com](http://www.WarioWareDIY.com)



## MAKE IT YOURSELF! PLAY IT YOURSELF! DO IT YOURSELF!

NINTENDO DS™

PLAY BIGGER!  
Nintendo DSi XL™



Comic Mischief  
Mild Cartoon Violence

NO PURCHASE NECESSARY. PURCHASE WILL NOT INCREASE YOUR CHANCES OF WINNING. Open to legal residents of the 50 U.S., D.C. and Canada (except Quebec). Must be legally able to travel to Los Angeles, CA. VOID WHERE PROHIBITED. Entry deadline: 5/16/10. Grand Prize includes airfare, two-night hotel stay and admission to 2010 Nintendo Media Briefing at E3 on June 15, 2010, for two persons (AFV: U.S. \$2,500). Odds of winning depend on the number and quality of entries received. Only one entry per person is permitted. Restrictions apply. For complete details (including judging criteria and how to submit a microgame design instead of a microgame to enter), see Official Rules at [www.WarioWareDIY.com/TheWarios](http://www.WarioWareDIY.com/TheWarios). Sponsor: Nintendo of America Inc., 4820 150th Ave NE, Redmond, WA 98052.





WORLD OF GOO by Indie Fund members Ron Carmel and Kyle Gabler.



## INDIE FUND ANNOUNCED

**INDEPENDENT GAME STARS LIKE** the WORLD OF GOO creators, BRAID's Jonathan Blow, and FLOWER's Kellee Santiago have revealed Indie Fund, an "angel"-style funding source for indie game makers.

According to the Fund's official website ([www.indie-fund.com](http://www.indie-fund.com)), "Indie Fund is a brand new funding source for independent developers, created by a group of successful indies looking to encourage the next generation of game developers."

The Fund was established "as a serious alternative to the traditional publisher funding model," and its aim is to support the growth of games as a medium by helping indie developers get financially independent and stay financially independent.

The current list of investors backing Indie Fund includes some of the most successful independent game creators of the last few years, including Ron Carmel and Kyle Gabler (WORLD OF GOO), Jonathan Blow (BRAID), Kellee Santiago (FLOWER), Nathan Vella (CRITTER CRUNCH), Matthew Wegner (OFF-ROAD VELOCIRAPTOR SAFARI), and Aaron Isaksen (ARMADILLO GOLD RUSH).

"Indie Fund is managed and fully funded by the seven of us. We put in enough money to fund a few games a year for two or three years. If things go well and it looks like the indie scene can take in a larger investment and put it to good use, we will raise another round, probably bringing in external investors as well," Ron Carmel said.

The Indie Fund has already backed a number of independent game projects, and will be announcing the name of them soon. "Actual funding has already begun, and we're also at various stages of discussing funding with several indie teams. This all happened through word of mouth within the indie community, but we will soon have a more open process for developers to apply for funding," Carmel added.

Speaking at the 2010 Game Developers Conference Independent Gaming Summit, Ron Carmel elaborated further on the Indie Fund's mission. Calling the relationships between indies and publishers "a system that never worked" he examined why independent game developers need an alternative to traditional publishing models. "How

do we do for funding what Valve did for digital distribution? The answer, we hope, is Indie Fund," Carmel said.

The Indie Fund aims to create a transparent distribution process. Stories of indies being pushed around by publishers results from no transparency in the process, said Carmel. "With the process that we're planning, it's going to be a lot shorter than the regular approval cycle for publishers."

The fund's investors also feel that publicly available deal terms are important so that developers can comparison-shop, he said. Indie Fund isn't revealing its terms yet until it is sure the model works, but transparency will be a must.

The fund will also offer developers a single point of contact. A personal relationship helps avoid conflict of interest and simplifies the process.

A focus will be on flexible development. "Anybody who's been in the game industry for more than a year or two realizes that when you start working on a game, you don't necessarily know how it's going to end up," he said. The process requires experimentation and iteration, so

the old methodology of coming up with big design documents ahead of time, with milestones attached that risk employee pay, ends up causing developers stress and ultimately risks game quality.

Under the Indie Fund model, the developer submits periodic builds to the fund along with a change list, so the game can be evaluated based on where it was last time it was evaluated, "not on where we think it should be," said Carmel. This approach "respects the game design process as it should happen," he added.

Importantly, Indie Fund will not seek IP ownership. "We want the developer to own the IP and for the developer to be master of their own destiny," Carmel said. This means no IP control either. "We don't want to tell you how to make your game," he stated. "If we provide funding for a game, then that's a vote of confidence in the team that they have a vision and that they can execute it," he said. "If I know better than you what's right for your game, then we probably shouldn't be funding your game."

— Leigh Alexander and Simon Carless



# 2600 MAGIC

**DAVID CRANE'S** RECENTLY released iPhone application 2600 MAGIC is the first in his TECHNICAL WIZARDRY series of documentary-style apps that will take players behind the scenes in the creation of iconic Atari 2600 titles. Detailing long-hidden programming techniques, his fascinating series offers an interactive glimpse into a bygone era from one of the originators. We spoke with Crane at the D.I.C.E. Summit 2010 where he was on hand to accept the Academy of Interactive Arts & Sciences first Pioneer Award.

**BRANDON SHEFFIELD:** *Why did you decide to create 2600 MAGIC as an app? Why not simply write a book?*

**DAVID CRANE:** You know, it was very simple. This Pioneer Award got me thinking, "Oh, I'm the

old guy now. Am I going to die soon and all this lore will be lost?" And I was just thinking, "What things were so magical about the technology and the tricks that we used that people today might find appropriate?"

There have been books written. A lot of the information on 2600 MAGIC, for example; four hours of research on Google will save you 99 cents. I mean, all that information is there. But I thought that I wanted to put some of these very esoteric techniques into print somewhere.

So DRAGSTER MAGIC, which is the second title, was really supposed to be the first. I said, "I can't put that out without at least providing some sort of background, you know, a little primer." And so, 2600 MAGIC. There's a free version. If you don't know

anything about the 2600, you can find out if these techniques are going to be interesting to you before you pay for the full version. But really, the series starts at DRAGSTER MAGIC. So, the question is "Would a book be more appropriate?" I thought it would be nice to be able to do animated figures, because it's an animated medium.

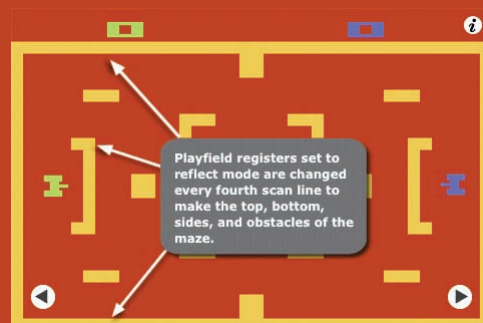
Also, I could go to the trouble of trying to find a publisher that's interested in publishing this—but then I said, "Wait a minute, I am a publisher. I'm an iPhone publisher!" So I figured I'd publish it on the iPhone. And it's not going to be a million-seller. It's just kind of quirky stuff that people who think this way will find to be really interesting.

**BS:** *I definitely found it very interesting, and I don't have that kind of*

*brain really, but it's neat what someone whose brain work this way can do with this kind of constrained hardware.*

**DC:** And the stuff we had

because it's a fabulous way of looking at it. What those guys were doing is they were looking at how the console that a game is designed for



to go through just to put a single object on the screen.

**BS:** *Right. I've always been really interested in people pushing hardware to the limits.*

**DC:** Well, I pimped the book *Racing the Beam* in, I think, DRAGSTER MAGIC

affects the game that is being designed. And that book is book number one in what they call their console series about games being designed for a particular console and its constraints, and how they affect the game design.

—Brandon Sheffield

# IGDA ELECTIONS

**FOUR NEW MEMBERS JOINED** the board of directors of the International Game Developers Association during the group's most recent election, and one incumbent retained his seat, organizers announced.

Incoming board members Wendy Despain, David Ederly, Darius Kazemi, and Jane Pinckard will now serve as directors for three-year terms. Coray Seifert successfully defended his seat for another three years.

Seven other existing board members are still partially through their own three-year terms that began in 2008 and 2009, while



another three—Erin Hoffman, Rodney Gibbs, and Brian Robbins—received their seats through special appointments, and will serve through September 2011.

Of the new board members, Despain is a writer and designer with the consultancy firm International Hobo, and was

recently credited on the MMO CARTOON NETWORK UNIVERSE: FUSIONFALL. Ederly formerly served as worldwide games portfolio manager for Xbox Live Arcade, and currently runs the design consultancy Fuzbi.

In addition, Kazemi is a former metrics analyst for developer Turbine, and now operates metrics firm Orbus Gameworks. Pinckard works in business development for Foundation 9 Entertainment, and is also known for her former career in game journalism.

Finally, Seifert is a game designer at THQ subsidiary Kaos Studios, developer of FRONTLINES:

FUEL OF WAR, and has previously held numerous other design roles.

According to the IGDA, this latest election drew a membership voter turnout of more than 10 percent, and the slate of candidates was the largest in the organization's history.

This year's elections drew an uncommon amount of attention, in part due to highly publicized affairs involving the resignation of board member Tim Langdell, who was embroiled in litigation-fueled controversy, and debate over the role of the IGDA in industry quality of life concerns.

—Chris Remo

June 15-17, 2010 \* Los Angeles Convention Center \* E3Expo.com



# CREATIVITY UNLEASHED

© 2010 Entertainment Software Association

# E3 Expo 2010

E3 Expo is the world's most important annual gathering for interactive entertainment. It's where business gets done, connections are made and the future of the industry is revealed.

International Media Sponsor

**MCV**

Official Show Daily Sponsor

**GAMEPRO MEDIA**

**REGISTER NOW AT E3EXPO.COM**

Produced By

**IDG**  
WORLD EXPO

E3 Expo is a trade event and only qualified industry professionals may attend. No one under 17 will be admitted, including infants. Visit [www.E3Expo.com](http://www.E3Expo.com) for registration guidelines.



# NINTH ANNUAL SALARY SURVEY GAME DEVELOPER

**WELCOME TO THE 9TH ANNUAL SALARY SURVEY FROM *GAME DEVELOPER*! WE'VE SHAKEN IT UP** a bit this year, splitting out indies and independent contractors into their own separate listings (see page 12), allowing for an even greater focus on a burgeoning area of the industry, while making our salary results even more streamlined.

The big number for the year, the average game developer salary across all disciplines and all levels of experience, is \$75,573. That's down almost \$4k from last year's number, and probably gets closer to reaching the actual average than we've gotten before. Making accurate predictions from these numbers is difficult, as they are voluntarily submitted, and who wouldn't want to pad their salary in a survey so they could point to what they should be making? That said, there are some interesting demographics here, from the disparity in salary for those who can afford homes, to the relative unimportance of education to pay rates.

2009 was a tough year, with record layoffs, but also record sales, which sends mixed messages. It also saw the rise of the Facebook and social media game, and the proven viability of the Apple's App Store. We asked developers several questions to gauge their thoughts about the current state of the industry, and learned some interesting things.

For example, only 13 percent of developers feel that there were more jobs in 2009 compared to 2008. A slightly higher group, but still not a majority, felt that the game industry

was improving (33 percent). And larger still was the group that felt there were more opportunities for game developers than ever before (38 percent). It should be noted that in this category 36 percent were neutral.

In spite of all the ups and downs of the year, only 9 percent felt that the game industry was no longer a good place to work, with the grand majority still feeling like this is the industry for them. Interested in hearing more specifics? See page 13 of this survey for some choice words from the developers surveyed.

Use this information wisely—now is probably not the best time to thunk down the survey in front of your boss and demand a raise. By all accounts, that intern you've been training is ready to take your spot at half the cost! Here's to a bright 2010!

—Brandon Sheffield and Jeffrey Fleming



## programmers

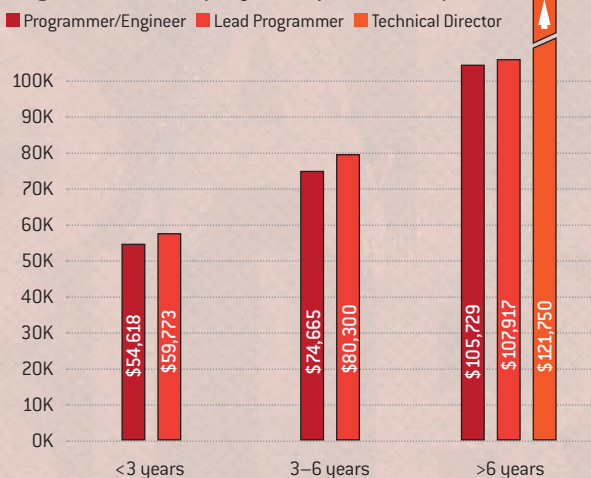
AVERAGE SALARY  
**\$80,320**

PROGRAMMING TALENT DRIVES THE GAME INDUSTRY, AND AS A GROUP coders generally enjoy the highest salary among the major disciplines (excepting Business). However, this year saw a decline in the average compensation, down from last year's \$85,024.

Still, the overall pay prospects are good. Junior programmers with three or less years of experience earned an average of \$54,975 in 2009 and experienced coders with more than six years on the job were paid an average salary of \$109,567.

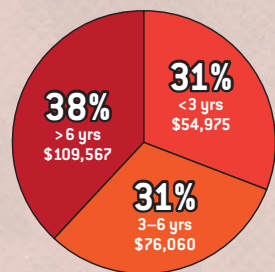
Programmers working in Canada earned an average of \$67,937 (USD) in 2009, which is slightly up from last year, while European coders were somewhat down from the previous year, earning \$46,198 (USD) on average.

Programmer salaries per years experience and position



### ALL PROGRAMMERS AND ENGINEERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **78%**

Average additional income: **\$15,937**

#### Type of additional compensation received

Annual bonus	49%
Pension/Employer contribution to Retirement plan	47%
Profit sharing	19%
Project/title bonus	29%
Royalties	16%
Stock options/equity	32%

#### GENDER STATS FOR PROGRAMMERS

Percent receiving benefits: **92%**

Gender	Percent Represented	Average Salary	Type of benefits received	Percent
Male	95%	\$80,128	Medical	98%
Female	5%	\$84,062	Dental	93%
			401K/Retirement	80%

## artists and animators

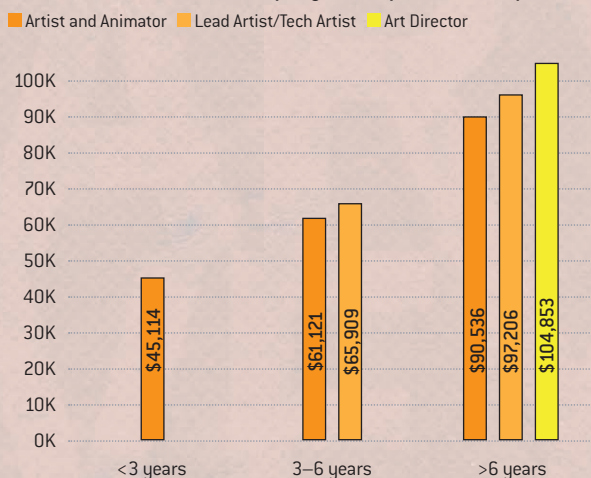
AVERAGE SALARY  
**\$71,071**

SALARIES FOR VISUAL ARTISTS WERE UP \$1,539 OVER LAST YEAR'S average. Although in general compensation was stable, 14% of the artists surveyed reported a pay increase over the past year, which is the highest percentage out of the creative disciplines.

The bulk of our artists surveyed have been in the business for three years or more, and at the top end their salaries were almost double that of entry level game artists (this could also mean that many low-level artists are on contract). The industry values talented art directors and lead/technical artists—across all experience levels these advanced disciplines earned a yearly average of \$94,000 and \$83,000 respectively.

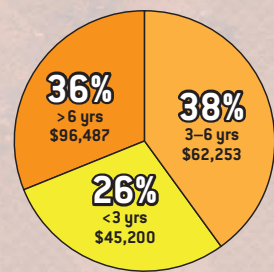
\$59,400 (USD) was the mean for Canadian artists, a salary that was also up slightly from last year. European artists brought home \$38,152 (USD), down more than \$5,000 from the previous year.

Artist and Animator salaries per years experience and position



### ALL ARTISTS AND ANIMATORS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **77%**

Average additional income: **\$12,217**

#### Type of additional compensation received

Annual bonus	46%
Pension/Employer contribution to Retirement plan	43%
Profit sharing	13%
Project/title bonus	32%
Royalties	17%
Stock options/equity	28%

#### GENDER STATS FOR ARTISTS

Percent receiving benefits: **94%**

Gender	Percent Represented	Average Salary	Type of benefits received	Percent
Male	92%	\$72,500	Medical	99%
Female	8%	\$51,071	Dental	94%
			401K/Retirement	81%



# SALARY SURVEY

## game designers

AVERAGE SALARY  
**\$69,266**

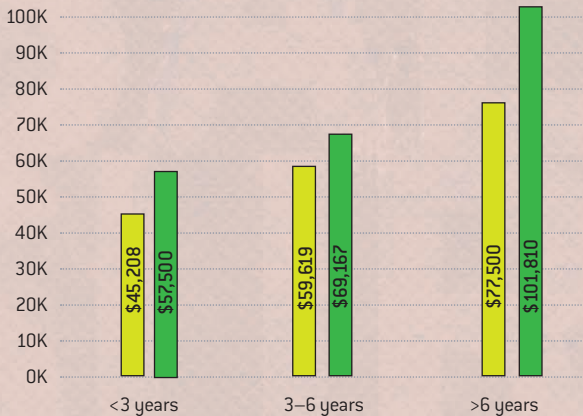
IN OUR SURVEY, GAME DESIGN ENCOMPASSES A RANGE OF JOBS including creative directors, designers, and writers. The field as a whole enjoyed a boost of almost \$2,000 over last year's average salary.

Predictably, creative directors/lead designers were well compensated, earning an average of \$90,640 per year, while game designers working in the trenches brought in \$61,859 for the year. Game writers reported making almost as much as non-lead game designers (\$61,786 on average), although our sample size for writers was low.

Canadian game designers in general earned \$61,520 (USD) this past year. Designers working in Europe reported earning \$42,423 (USD), which is up almost \$2,000 over last year's survey.

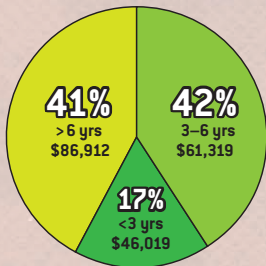
### Game Designer salaries per years experience and position

■ Game Designer ■ Creative Director/Lead Designer



### ALL GAME DESIGNERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **72%**

Average additional income: **\$12,485**

#### Type of additional compensation received

Annual bonus	40%
Pension/Employer contribution to Retirement plan	47%
Profit sharing	15%
Project/title bonus	34%
Royalties	12%
Stock options/equity	35%

Percent receiving benefits: **96%**

#### GENDER STATS FOR DESIGNERS

Gender	Percent Represented	Average Salary	Type of benefits received	Percent
Male	92%	\$69,790	Medical	99%
Female	8%	\$62,500	Dental	97%
			401K/Retirement	80%

## producers

AVERAGE SALARY  
**\$75,082**

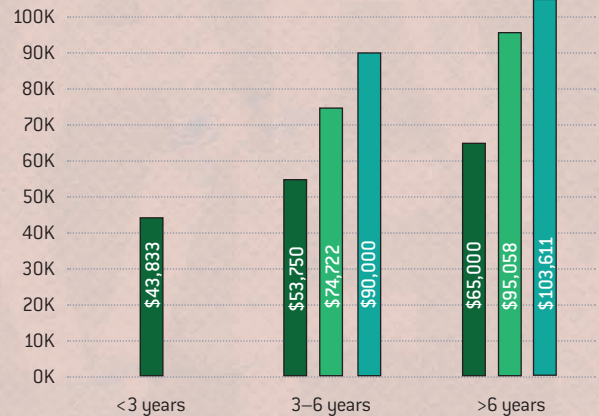
THE OVERALL SALARY FOR THE PRODUCTION DISCIPLINE IS DOWN \$7,823 from last year's average. The majority of our production respondents had three or more years of experience and 48.8% had been working in the industry for six or more years—the highest percentage out of all of the disciplines.

Of the creative disciplines production had the highest percentage of female respondents. According to the data, women were paid an average of \$4,814 less than their male coworkers in the production field.

In Canada, producers brought in an average of \$87,130 (USD) for 2009 and European producers made an average of \$52,125 (USD) for the year.

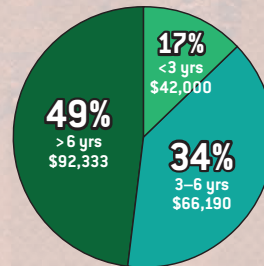
### Producer salaries per years experience and position

■ Associate Producer ■ Producer/Project Lead ■ Executive Producer



### ALL PRODUCERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **77%**

Average additional income: **\$14,565**

#### Type of additional compensation received

Annual bonus	53%
Pension/Employer contribution to Retirement plan	42%
Profit sharing	12%
Project/title bonus	26%
Royalties	11%
Stock options/equity	34%

Percent receiving benefits: **96%**

#### GENDER STATS FOR PRODUCERS

Gender	Percent Represented	Average Salary	Type of benefits received	Percent
Male	82%	\$75,950	Medical	97%
Female	18%	\$71,136	Dental	93%
			401K/Retirement	81%



## audio developers

AVERAGE SALARY  
**\$82,045**

### IF THE GAME INDUSTRY EVER MOVES TOWARD A HOLLYWOOD

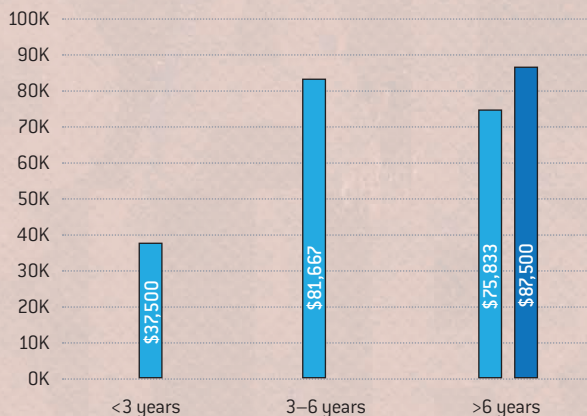
production model (see Steve Theodore's March 2010 Pixel Pusher column) we may have an early glimpse of what that could mean for salaries and benefits by looking at the largely independent Audio Developers field.

Those working with audio, which include directors, composers, and designers, reported earning \$3,878 more per year on average than they did last survey. Interestingly, while audio developers were among the least likely to receive additional compensation (excepting QA) for their work, the average additional income for those that did was among the highest (\$15,875) in the creative disciplines, particularly with regard to royalties.

Canadian audio developers earned on average \$61,250 (USD) up from last year's \$58,929 (USD). Audio developers working in Europe reported a yearly average of \$40,833 (USD), down a bit from the previous year's \$42,955 (USD).

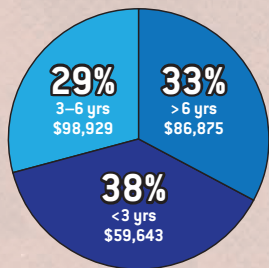
### Audio Developer salaries per years experience and position

■ Sound/Audio Designer/Engineer ■ Sound/Audio Director



### ALL AUDIO DEVELOPERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **71%**

Average additional income: **\$15,875**

#### Type of additional compensation received

Annual bonus	53%
Pension/Employer contribution to Retirement plan	47%
Profit sharing	20%
Project/title bonus	40%
Royalties	27%
Stock options/equity	27%

#### GENDER STATS FOR AUDIO DEVELOPERS

Percent receiving benefits: **86%**

Gender	Percent Represented	Average Salary	Type of benefits received	Percent
Male	88%	\$81,184	Medical	100%
Female	12%	\$87,500	Dental	89%
			401K/Retirement	83%

## qa testers

AVERAGE SALARY  
**\$37,905**

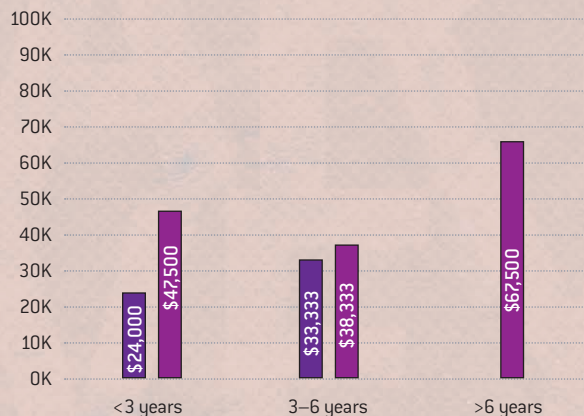
QUALITY ASSURANCE HAS THE LOWEST BARRIER TO ENTRY OF ALL THE disciplines and accordingly has the lowest pay and least benefits. As a result, turnover is high and the position is often seen as a stepping stone to other areas of game development rather than a career itself.

The QA discipline, which includes entry-level testers as well as more experienced leads, averaged a salary of \$37,905 overall, down \$1,666 from last year. Testers with less than three years experience on the job reported earning an average of \$24,000 annually, while experienced leads that had been in the business for six or more years reported earning \$67,500 on average.

QA workers based in Canada earned an average of \$39,375 (USD) per year and those working in Europe brought in \$29,500 (USD).

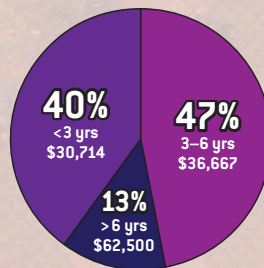
### QA Tester salaries per years experience and position

■ Tester ■ QA Lead



### ALL QA TESTERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **59%**

Average additional income: **\$5,841**

#### Type of additional compensation received

Annual bonus	45%
Pension/Employer contribution to Retirement plan	55%
Profit sharing	10%
Project/title bonus	20%
Royalties	0%
Stock options/equity	20%

#### GENDER STATS FOR QA TESTERS

Percent receiving benefits: **79%**

Gender	Percent Represented	Average Salary	Type of benefits received	Percent
Male	89%	\$37,803	Medical	100%
Female	11%	\$38,750	Dental	90%
			401K/Retirement	77%



# SALARY SURVEY

## business and legal people

AVERAGE SALARY  
**\$96,408**

OUR SURVEY OF THE BUSINESS AND LEGAL DISCIPLINE LOOKED AT a wide range of positions including chief executives and executive managers, community managers, marketing, legal, human resources, IT, content acquisition and licensing, and general administration staff.

Overall, the average salary for the business sector was \$96,408, down \$5,735 from last year. Still, those working on the business side are the most likely (89%) to receive additional compensation and enjoy the highest average amount of additional income (\$17,807). They are also some of the more experienced workers in the game industry with only 16.9% having been on the job less than three years.

Examining the various job titles revealed considerable variation in yearly salaries. Executive managers earned quite a bit more than the mean, bringing in an average of \$129,167 annually. Marketing and PR positions averaged \$83,804 and human resources earned \$71,136.

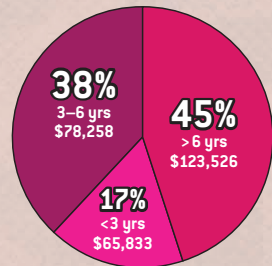
Looking at the salary for business-oriented personnel in general for Canada revealed an average of \$58,929 (USD) while Europe averaged \$59,231 (USD).

### ALL BUSINESS AND LEGAL PEOPLE

#### YEARS EXPERIENCE IN THE INDUSTRY

Percent receiving additional income: **89%**

Average additional income: **\$17,807**



#### Type of additional compensation received

Annual bonus	64%
Pension/Employer contribution to Retirement plan	30%
Profit sharing	29%
Project/title bonus	18%
Royalties	11%
Stock options/equity	41%

#### GENDER STATS FOR BUSINESSPEOPLE

Percent receiving benefits: **91%**

Gender	Percent Represented	Average Salary	Type of benefits received	Percent
Male	75%	\$100,192	Medical	97%
Female	25%	\$85,227	Dental	91%
			401K/Retirement	78%

## LAYOFFS

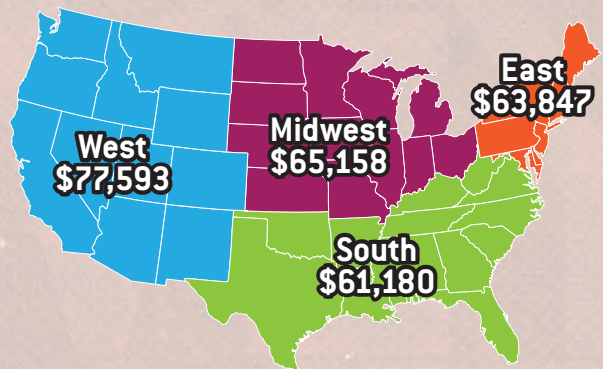
FOR THE SECOND YEAR, WE ASKED OUR SURVEYED DEVELOPERS whether they lost their jobs at any time during the last year. The number is up from the previous period. In 2008, 12% of those surveyed lost their jobs. In 2009, that number climbed to 19%.

Of those who were laid off, 46% found new employment at a game studio or publisher, 17% moved into contract or consulting work, 10% founded a new company, and 16% went into indie development. Unfortunately, 25% were unable to find new work in the game industry.

One particularly interesting statistic is the rather large number of people moving to independent development. Last year, consulting and indie moves combined comprised about 24% of what laid-off developers did with their time. This year, that combined number climbs to 33%, indicating the increased viability of independent game development or working on a game-by-game basis.

## AVERAGE SALARY BY U.S. REGION

(across all levels of experience and disciplines)



## TOP 5 STATES WITH HIGHEST AVERAGE SALARIES

(across all levels of experience, excluding states with low sample size)

	AVERAGE SALARY	PERCENT WHO OWN HOMES	AVG. SALARY OF HOMEOWNERS
1 California	\$80,557	31%	\$107,370
2 Washington	\$73,981	41%	\$93,728
3 New York	\$64,167	20%	\$77,500
4 Florida	\$60,357	38%	\$70,395
5 Texas	\$60,326	43%	\$70,245

## AVERAGE SALARY BY U.S. REGION BY DISCIPLINE

	EAST	MIDWEST	SOUTH	WEST
Programmer	\$63,198	\$71,154	\$64,542	\$91,805
Art and Animation	\$62,500	\$54,265	\$60,521	\$80,196
Game Design	\$62,262	\$75,682	\$56,842	\$73,840
Production	\$68,250	\$41,389	\$71,167	\$81,474
Audio	\$70,833	\$87,500	\$50,000	\$98,000
QA	\$36,250	—	\$28,214	\$41,591
Business	\$87,885	\$122,500	\$89,868	\$99,265

## AVERAGE SALARY FOR HOMEOWNERS VS. NON-HOMEOWNERS BY U.S. REGION

	EAST	MIDWEST	SOUTH	WEST
Homeowners	\$82,500	\$82,244	\$73,776	\$99,108
Non-Homeowners	\$52,266	\$47,361	\$49,493	\$63,717

## AVERAGE SALARIES IN THE U.S., CANADA, AND EUROPE

(across all levels of experience, by discipline, given in USD)

	U.S.	CANADA	EUROPE
Art and Animation	\$71,071	\$59,400	\$38,152
Programmer	\$80,320	\$67,937	\$46,198
Game Design	\$69,266	\$61,520	\$42,423
Audio	\$82,045	\$61,250	\$40,833
Production	\$75,082	\$87,130	\$52,125
QA	\$37,905	\$39,375	\$29,500
Business	\$96,408	\$58,929	\$59,231



## AVERAGE SALARY BY EDUCATION LEVEL AND DISCIPLINE

(across all levels of experience)

	ART	PROGRAMMING	DESIGN	AUDIO	PRODUCTION	QA	BUSINESS
High School Diploma or GED	\$74,167	\$79,375	\$56,136	\$114,167	-	\$35,000	\$72,500
Some College	\$101,310	\$98,500	\$79,929	\$140,000	\$78,833	\$28,056	\$75,833
Technical Certification	\$52,500	\$79,167	\$80,000	\$87,500	\$52,500	-	\$82,500
Associates Degree	\$69,000	\$68,409	\$67,500	\$60,000	\$60,833	\$40,000	\$65,000
Bachelors Degree	\$66,492	\$76,383	\$67,767	\$72,045	\$70,643	\$40,119	\$96,429
Some Graduate	\$93,500	\$91,923	\$68,611	\$117,500	\$91,875	\$30,000	\$110,500
Masters Degree	\$68,816	\$78,013	\$63,750	\$32,500	\$88,125	\$97,500	\$102,976
Some Doctoral	-	\$111,786	\$57,500	-	\$127,500	-	-
Doctoral Degree	-	\$77,500	-	-	-	-	\$110,833
Some Post-Doctoral	-	-	-	-	\$107,500	-	-
Post-Doctoral Degree	-	-	-	-	-	-	\$167,500

## METHODOLOGY

**NOW IN ITS NINTH YEAR,** the *Game Developer Salary Survey* was conducted in February 2010 for the fiscal year January 1, 2009 through December 31, 2009, with the assistance of Audience Insights. Email invitations were sent to *Game Developer* subscribers, Game Developers Conference attendees, and Gamasutra.com members asking them to participate in the annual survey.

We gathered 4,050 responses from developers worldwide but not all who participated in the survey provided enough compensation information to be included in the final report. We also excluded salaries less than \$10,000 and the salaries of students and educators. The small number of reported salaries greater than \$202,500 were excluded to prevent their high numbers from unnaturally skewing the averages. We also excluded records that were missing key demographic and classification numbers.

The survey primarily includes U.S. compensation but consolidated figures from Canada and Europe were included separately. The usable sample reflected among salaried employees in the U.S. was 1,014, for Canada 275, and for Europe 378; and 605 for indies and independent contractors who provided compensation information worldwide.

The sample represented in our salary survey can be projected to the U.S. game developer community with a margin of error of plus or minus 3.06% at a 95% confidence level. The margin of error for salaried employees in Canada is plus or minus 5.9%, and is 5.0% for Europe.

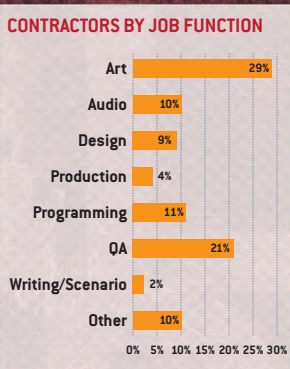
## THE INDIE REPORT

**AS A NEW PART OF OUR SURVEY,** we have included new data on independent developers and independent contractors (meaning those who are not part of a contract development team, such as an art outsourcer).

Of those who match the above description, 40% identified as an independent contractor, 30.4% were members of an indie team, and 29.6% said they were an individual indie developer.

So how much did these people make? Independent contractors fared the best, with an average compensation of \$45,137. Among full-on team-based indie developers, the average was less than half, at \$20,248. Individual indie developers fared the worst, making a scant \$11,638.

Not all indie developers make their money exclusively through game development though. To that end, we also inquired about revenue from related merchandise, such as t-shirts, comics, toys, and the like. Less than 10% of our indie respondents made extra money through those means, and of those, 23% made less than \$100 on their extra promotions. 21% made



between \$100 and \$1,000, 21% made between \$1,000 and \$5,000. 18% made between \$5,000 and \$20,000, and a rather surprising 16% of indies involved with non-game merchandising made over \$20,000 on their extra ventures.

We also asked the entire group, both indies and contractors, whether they had ever worked at a traditional salary-based game developer, and a very large number had not—68%, in fact. This is significant especially when you consider the fact that almost 27% of our respondents identified as a non-salaried game developer. That 68% of that larger

number have not worked at a traditional game developer before indicates that the roads to game development are clearing again, and that the traditional story of having to work up from QA perhaps applies less than it did in the past.

### JOB FUNCTIONS

Indies are often necessarily a one-stop shop for all aspects of game development, so rather than ask them what one element of game development they were responsible for, we asked them to check all boxes that apply. So read the indie chart as “75% of indies worked on at least design.” In the case of contractors, we asked them to choose only one area in which they made the majority of their money.

### INDIES BY JOB FUNCTION

Art	48%
Audio	26%
Design	75%
Production	53%
Programming	66%
QA	50%
Sound	26%



### AN EXTENDED VERSION OF THE 9TH ANNUAL GAME DEVELOPER

Salary Survey, including detailed data for year-over-year results since 2004, will be made available for purchase through Game Developer Research, a division of United Business Media, LLC. Visit [www.gdmag.com/research](http://www.gdmag.com/research). This detailed report, The Game Developer Salary Report: 2004—2009 will be available in April.



# SALARY SURVEY

In order to hear what developers are truly saying about the economy right now, we allowed space at the end of our survey for direct comments. We include some more notable responses here.

## THE BAD

“2009 was a mixed bag for the industry. Even though software sales remained relatively strong, development studios were being shuttered at an alarming rate. I feel this is due to the industry reaching a saturation point prior to the recent economic recession. Now I feel there are way more applicants than there are jobs to fill, and larger studios are taking advantage by making low-ball offers to new hires and neglecting the wellbeing of their development employees to improve their bottom lines. I'm sure this occurs in many other industries, but unless things turn around soon, the high cost of game development and the high retail prices of games will make for fewer and fewer studios taking chances on “riskier” projects, thus stunting the growth of the industry.”

“2009 has been a good year for smaller, independent developers and very bad one for publishers and publisher-owned developers.”

“There's a lot of movement in the future toward social/broad reach games with new business models, and they're coming faster than most anticipate. A large percentage of the industry is still ignoring the appeal of these types of games and there needs to be a better awareness of why they work. Not everyone needs to work on these types of games but their systems could be brought more into the mainstream titles.”

“As a recent graduate of DigiPen Institute of technology in 2009 I am becoming more jaded by the game industry. As many friends of mine have gone to work for many game companies in the Seattle area, I have witnessed the systematic exploitation of young, extremely talented, extremely driven artists. Of course,

there are some game companies that do treat their employees well, but the majority of what I have seen is disgraceful. Increasingly, “permalancers” are becoming all the rage. Companies abuse the contract system in order to get cheap labor from talented artists while at the same time denying them full-time



benefits such as health and dental insurance plans. It hurts me to see my friends always worrying that their contract won't be renewed at the time of review and to see their spirits slowly breaking under the pressure that type of employment puts on people. This treatment of artists is a race to the bottom. I no longer feel that the people at the top in these game companies care for the wellbeing of the artists under their roof. The priorities seem to be aligning toward the bottom line and profit margins rather than the greater good. In the end you will have a generation of alienated and hurt artists, and let's not forget, human beings. Would I buy the next great game if I know that it was developed with the blood and sweat of overworked and exploited artists? Certainly not.”

“The game industry has become highly volatile for established industry veterans. Companies are laying off full time employees as soon as the game is ready to ship, and they often chop their higher paid senior staff first instead of last. This practice used to be applied to contractors first,

temp employees second, and then the senior staff when there were none left to cut.

Now, once your project is completed, all bets are off. And those kids fresh out of school that you spent the last few years training? They will either be laid off and you will have to train another new batch of students from scratch, or one of them will be given your job. Seniority? Forget about it.

This destabilizing practice of dumping seasoned talent is leading many to become cynical about working in the industry beyond their 30s. Additionally, with the constant reduction of long-term gigs, artists, game designers, and engineers frequently have to move to other states or even other countries to continue to eke out a gig-to-gig living. Game designers are becoming nomads or guns for hire.

If you are 24 years old, single, don't own a house and don't have a family, then this rolling stone lifestyle is probably very doable and may even sound fun. But if you are married, your spouse has a local job, your kids are in school, and you have local friends and family, this lifestyle can be very disruptive and is not sustainable.”

## THE GOOD

“2009 was a great year for some absolutely monumental titles such as UNCHARTED 2, MODERN WARFARE 2, plus refreshing indie games such as PIXELJUNK SHOOTER and MACHINARIUM, which really pushed the boundaries of playability and visual style, raising the bar for the future of gaming. Some wonderful titles on the iPhone too, such as CANABALT, MINISQUADRON, and GEODEFENSE. All in all, 2009 was a pinnacle year, with some eagerly anticipated titles, and previews for the coming year.”

“The return to more small casual and social games means it's a great time to be a

designer ... lots of opportunities to exercise innovation and creativity! Reminds me of my early days in the arcade industry.”

“2009 was a year of major transition. The last time I can remember seeing this much change was in the late '80s to early '90s. The direct-to-consumer digital revolution is definitely upon us—how we choose our next steps will be critical to long term success”

“I think the trend to Indie success stories has been inspiring for many young designers, artists, programmers, and sound designers alike. This has been uplifting and I hope it continues. People are showing that new markets can be created out of ingenuity and a good product without the aid of big money and major ad campaigns.

I am happy that the online market has embraced the 'free to play, fee to upgrade' mantra with gusto. It is in the hands of the developers now to make sure not to squander their trust. It is vital that this give and take isn't abused. Customers will run to other markets the second they feel cheated. It would then take a concerted effort to bring those clients back into the fold. 'Premium content' has to mean just that.”

“While there were a lot of jobs lost during 2009, I am confident that the industry will move forward and that the market will grow. I also believe with all the emerging technologies that there are now many more opportunities for game developers within and around the games industry that extend to fields like animation, computer graphics, interactive web applications, and so forth. Just because people are losing jobs in the industry does not mean that they have nowhere else to turn, by any means.”



# THE THREADS THAT BIND US

Adapting Lessons from Erlang  
for Multithreaded C++



## EVERYBODY AGREES: MULTITHREADED DEVELOPMENT IS A MISERABLE

experience. Anybody who has tried it will have at least one horror story about the race condition that they just couldn't find, or the crash bug that stopped the game from shipping on time. We have all gotten comfortable thinking about our programs as a set of sequential single-core operations, and as soon as we try to deviate from this model, our brains—not used to the rigors of concurrency—start hurting. Everybody also agrees: multithreaded development is the way of the future. The free lunch is over, and with multiprocessor machines in the marketplace and modern consoles requiring us to embrace this paradigm, we must adapt if we want to fully exploit the possibilities of the next generation of hardware.

Trying to successfully write concurrent code with our feeble intellects requires us to either accept race conditions, or to think outside the box. In many ways, the threads-and-mutexes model of programming is wrong for concurrent development; in his lecture at the Symposium on Principles of Programming Languages in 2006, Tim Sweeney noted that “manual synchronization (shared state concurrency) is hopelessly intractable.” Other models for concurrent programming exist, and we need to start looking at them.

The programming language Erlang was originally developed by telecom giant Ericsson during the late '80s, and was released as open source in 1998. In spite of its age, Erlang offers a few creative solutions to the concurrency conundrum. Telecoms have used it for years to handle the problem of massively concurrent programming for systems with cores that number in the thousands. More recently, social networking sites such as Facebook and del.icio.us have used Erlang to help build scalable systems for delivering web content. Erlang works very, very well across thousands of cores. How well does it work for us, and what can it teach us about how to program for multiple cores without losing our minds?

## GETTING ON MESSAGE

» To begin, let's look at Erlang directly, as it's a language designed for concurrency. Erlang's creator, Joe Armstrong, designed the language around four fundamental tenets based on life in the real world: the world is concurrent, things in the world don't share data, things in the world communicate with messages, and things in the world will often fail. Every design decision in Erlang stems from these four tenets; the language is fully designed from the ground up to embrace concurrency rather than simply supporting it.

The first thing that you must understand about Erlang is that it's a functional programming language. As such, it has no concept of state, and it has no concept of mutable data. This stems from tenet number two of Armstrong's design philosophy: with no mutable data, and with no concept of state, we are encouraged to work in a way that promotes concurrency. To understand the difference between a functional programming language and a procedural one, consider the following Erlang routine to calculate a factorial:

```
factorial(0)->1;
factorial(N)->N*factorial(N-1).
```

The idea of using recursion instead of a for loop or other procedural data structure is at the core of functional programming, and it may make your head hurt until you get used to it. Even more mind-boggling is the idea that Erlang variables are write-once. For instance, we can write:

```
Foo = 2.
```

or even:

```
Foo = Bar * Baz.
```

but because the variable `Foo` is already assigned, we cannot write:

```
Foo = 2, Foo = Foo * Foo.
```

This has some relevance to our concurrency issues: by not having any state,

it becomes impossible to share it. When you can create a variable, Erlang lets you assign a value to it once, and only once. At that point, it becomes fixed. Erlang is not a pure functional programming language, in the sense that it is possible to produce what is known as a side effect; my experience has shown that spending 80 percent of your time doing functional programming and 20 percent of your time pretending that you're writing 1980s-era BASIC code seems to work well.

Spawning a process in Erlang is as easy as one function call:

```
Pid = spawn(Module, Function, Arguments).
```

People create Erlang processes willy-nilly; it's a regular occurrence. Compare this to our standard programming practices in C++, in which creating a thread requires planning, a specific reason to do so, and at least two meetings with your project's lead programmer. To an Erlang developer, the C++ notion of threading can be compared to an object-oriented programming language in which we can create objects, but only about five of them. The problem isn't object-oriented programming; the problem is that our language's implementation of objects is terrible.

Erlang's main strength, as far as we are concerned, is in how the language itself helps us program concurrently. Rather than having threads communicate with each other via shared data, Erlang processes communicate with each other via an asynchronous message-passing architecture. The process ID, returned by the `spawn()` routine, is used as a reference point for such communication. This model encourages the creation of many smaller, isolated processes. Each process then communicates via message-passing with other processes that it needs to talk to, and all concurrency issues are resolved by the message-passing architecture. Erlang processes are so-called “green” threads, in that they are scheduled internally by the Erlang virtual machine; however, Erlang itself will distribute your program across multiple cores—in fact, it will distribute your program across multiple computers and multiple instances of the Erlang interpreter. This little miracle is achieved thanks to a quirk of the Erlang message passing system, which internally uses sockets (yes, sockets) to communicate between instances of the Erlang VM on separate threads.

Note that Erlang threads cannot share data between themselves. The language simply does not allow two threads to have access to the same chunk of memory. How well does this work in practice? Surprisingly well. When you spawn a new process, you simply pass it whatever information it needs to take care of business in the list of arguments. This is not always suitable for some tasks—in particular, those tasks where you really want shared data (for instance, compressing a texture)—but programming without shared data is surprisingly palatable once you complete the mental gymnastics necessary to do so. If you need to send further data around, you simply send it over as a message to the thread that needs it; it then makes a local copy of the data. Memory requirements do not balloon as much as you might expect. This is partly because functional programming languages discourage the acquisition of data and state anyway, so we don't have much of it to spread around in the first place.

## ERLANG IN GAMES

» Is Erlang suitable for your application? If you are writing an MMORPG or some other application that requires scaling across a large server-based architecture, Erlang is ideal. It comes with Mnesia, one of the most powerful databases in the world, and it scales like crazy. (To the best of this author's knowledge, the only shipping MMORPG to actually use Erlang for its backend was VENDETTA ONLINE.) It also comes with a generational copying garbage collector, which is an essential item for any garbage collected scripting language designed for real-time operation.

For writing an actual video game, and not just a server, Erlang may not be quite the right tool. Erlang's socket-based interprocess communication, which does allow it to scale easily across multiple machines, may not be the most optimal tool for a single machine with less than ten cores. Shared memory access—albeit shared memory access handled by the Erlang VM—might be more promising, but it's not part of Erlang's mission statement. Provided that you are willing to learn it, Erlang may also be suitable for conditions where



# THE THREADS THAT BIND US

your project requires multi-core support, but where you do not have the programming manpower to spend time and energy debugging a traditional shared-data style language. In this case, it may be a good idea to write anything that is highly procedural in a C++ extension to Erlang, shove that in its own thread, and then use Erlang for all the tasks that do need to be efficiently parallelized. The trade-off of reduced efficiency on a project that ships versus 100 percent efficiency on a product that never ships is one that I am completely willing to accept.

The main problem with mixing Erlang and C is that the Erlang-to-C binding library is fairly immature. It certainly can be done, but it is not the most pleasant prospect. The C++ part of your program wants to think in terms of objects and function calls, whereas the Erlang part of your program simply wants to throw messages into a void and be done with it. Writing physics or rendering code that gets controlled by a single event passing mechanism—a rather large one, at that—is not my idea of a good time. Still, it's worth a thought.

## PAGING DOCTOR GREENTHREAD

» For those of us who want to stick to C++ and not work with Erlang at all, the question remains: How can we use the ideas behind Erlang in C++? Can we produce a massively-threaded application in C++? Well, yes and no. We are stuck with OS-level threads, which are generally considered very “heavy” by Erlang programmers, and we have no functional programming and no language support. Language support, in my mind, is the big issue; without language support, a sufficiently determined programmer can always screw something up and make hash of a beautifully designed multithreaded application. We have to roll our own process-based message-passing system, which C++ people, for some reason, usually refer to as an actor-based system to distinguish OS-level processes from green processes.

Let's consider what an actor in this system needs to do. It needs to have a message-passing queue, meaning that other actors (in other threads) can write to it and we can read from it. It needs to live in its own thread, and we need a way of making sure that every actor in a given thread gets its chance to parse messages. We'll need to develop a thread-safe queuing operation, and we'll need to use whatever C++ features we can in order to make sure we don't accidentally get too clever for our own good. Since we don't know what a given actor might look like, we will separate the code to handle the actor mechanisms from the actual functionality of a given actor. My preferred approach is to use a template, but your mileage may vary. Using a template allows us to add some degree of security to our actor. By making the instance of any given actor class private, we can ensure nothing can access the class except through the methods that the templates allow. What we cannot stop is the actor class performing operations outside of itself that are not thread-safe. The best we can do is try to be self-disciplined and ensure that we never, ever, put something in a global space that will be accessed by multiple processes. A determined programmer who decides to make a global variable in C++ without a mutex and then has multiple green threads accessing it simultaneously can still make trouble. This is where the safety of Erlang's internal language support for concurrency comes in handy.

To support messaging in a “green thread,” each actor needs to have a queue for incoming messages. Because this is the main structure where threads may interact with other threads, we need this queue to be thread-safe. Fortunately, thread-safe queues are well understood. What we would really like is for our queue to be lock-free, but the absence of atomic operations in C++ makes this a somewhat difficult prospect. The new C++0x standard proposes an atomic keyword which would be useful, but unfortunately we don't have this yet. Instead, we must resort to using the Win32-specific function `InterlockedCompareExchangePointer()` to do our dirty work for us, or, alternately, override the compiler's judgment and write some assembly code. [Brrrr.]

To give you an idea of the difficulties of lock-free programming, consider what happened in 2008 in programming publication *Dr. Dobbs's Journal*: a respected developer produced an implementation of a lock-free queue in a magazine article. Herb Sutter spent the next three issues analyzing why it didn't work and how to fix it. Accordingly, I have lifted Sutter's implementation more or less verbatim; his was written to the C++0x standard, so I have simply removed his use of the atomic keyword and replaced it with the appropriate Win32-specific calls. Note, however, that `InterLockedCompareExchangePointer()` and `InterLockedExchangePointer()` just wrap a couple of

## a note on performance

How fast is Erlang compared to other languages? If you look at the Great Language Shootout (see Resources) you will see that Erlang's speed is a mixed bag, to put it mildly. For instance, if you compare Erlang's performance to Python, the former outperforms the latter on a number of tasks (most of the math-flavored tests) and does not handle other things nearly as well (various tasks to do with regexes, for instance). But it certainly appears to be more than fast enough for games—provided that you are prepared to accept the fact that any interpreted language will be slower than its compiled equivalent.

It is worth noting that these tests do not tell the whole story; they are not always designed well, and do not always emphasize a programming language's true strengths when it comes to speed and efficiency. Read the results for yourself, and be sure to look at the Language Shootout maintainer's notes on why, exactly, you shouldn't believe any of the tests that they wrote.

assembly functions; if you don't like using them, write the assembly code yourself. In this implementation, writing and reading are lock-free operations, but because we may have multiple threads sending messages to the same actor, we need to use a traditional mutex operation to handle the possibility of multiple threads writing to the queue. Under our Erlang-esque assumptions, messaging is asynchronous; we don't care what order stuff gets written to the queue in, as long as it gets there.

To actually transmit messages, there are a number of options, again. In the interest of simplicity, I have used the Boost: `any<>` template class from the superb Boost library. (I have also used their mutexes and threading library, but this is more a matter of convenience than anything else.) Feel free to use whatever message-passing scheme you feel works best for you; I envision a design in my head where different sorts of actors send each other different sorts of messages, and the `any<>` template is as good a tool to accomplish this as any. The last thing that we need to do is to determine how to handle the processing of messages. If we were running interpreted code on a virtual machine, we would be able to simply trade off between processes as our whims demand it. Unfortunately, we can't do that with C++, and we have





## Introducing Vision Engine 8

- Now optimized for performance on browsers, PS3, Xbox 360, PC and Wii
- Superior graphics power across genres and platforms
- Unrestrictive workflow, including many 3rd party integrations and built-in editors
- Flexible licensing to meet every project's needs. Free from royalties and back-end fees

*Experience the freedom*



[www.trinigy.net](http://www.trinigy.net)  
[www.sneakpeek8.com](http://www.sneakpeek8.com)



# THE THREADS THAT BIND US

to resort to another queue. Specifically, we create one thread per processor that we wish to distribute tasks to, and these threads then pull actors off of a giant queue. This queue needs a read mutex and a write mutex. While our queue will allow for simultaneous reading and writing, it really just means that we have to have two mutexes instead of one. Still, this means that adding things to the queue of processes to be parsed will not lock threads trying to figure out what to do next—we will simply have to take what we can. (Again, proper support for atomic types or a different lock-free queue implementation, might help matters.)

## IN SEARCH OF CONCURRENCY

» The sample code accompanying this article includes my implementation of an actor-based system for multithreaded processes. It is designed to provide support for an Erlang-type green-threaded system where processes (animation or gameplay logic, for instance) can be dispatched across multiple independent processes. Note that the code is thread-safe only if you obey the Erlang-style convention that no data is shared between processes. Every process must be passed a copy of the data that it requires through its creation process. As it stands, the code makes life very difficult for you to do otherwise—you would have to add new global variables outside of the classes, and then use them to directly bypass class security. You will also need to have Boost installed in order to run it.

The sample program can be found online at [www.gdmag.com/resources/code.htm](http://www.gdmag.com/resources/code.htm). It contains a sample program that shows how we might use this approach to concurrency in a video game. Here, two types of tasks are being performed: some characters move around in a map, and some particle systems are animated. Each particle system, and each character, is its own actor. They communicate with each other, with the renderer, and with the map, by message passing. This is a trivial sample, but it demonstrates that it is possible for us to use a message-passing, actor based system with no shared data in a real world example.

In the end, how close does our system come to achieving Erlang's simplicity? All things considered, I'm pleased with the result. We can send anything we like as a message, provided that we know how to successfully `any_cast<>` it to something that we can accept. (If we get a bad message, we can catch the exception and then process it.) We can create new actors and add them to the actor queue any time we please. Creating a new green thread is as simple as creating a new instance of the actor class we wish to use, wrapped by the `ActorTemplate<>` class, and throwing it at the `ActorManager` singleton to be added to the queuing mechanism. We have some guarantee that deadlock will be handled semi-gracefully—if two actors are waiting on messages from each other, the system will continue to operate without their involvement. Better yet, the code to accomplish all this is fairly small, weighing in at two reasonably small header files. This style of concurrent programming is elegant in its simplicity, and this is reflected in the simplicity of the code.

Another disadvantage of C++ here is figuring out which process owns which data, and as such, which process is responsible for its own cleanup. Ideally, any data that is used by an actor should be deleted by that actor, but enforcing that is not easy. Just keep track of who owns what, obey all sensible and sane memory-handling precautions, and life will be fine. (Easier said than done!)

It's worth taking a minute to examine this actor-based approach to "microthreading," as implemented by libraries such as Intel's TBB library and John Ratcliff's Jobswarm. Using a process-based architecture versus a job-based architecture is partly a matter of perspective. In a library such as Jobswarm, sequential tasks are parallelized by splitting them into "jobs." In this approach, everything is considered to be a process. The disadvantage of a job-based model is that it encourages laziness. We break tasks down into jobs when it is easy to do so, but we are still thinking as though we are operating with only one thread. When things do not parallelize well with a jobs model, we simply don't parallelize them—instead, we shove all these difficult things into our main thread and do them in a way that makes us feel comfortable. In the Erlang model, concurrency is something that is treated as a first-class citizen. Everything is concurrent, because being concurrent is always easier than not being concurrent. The cost of producing another actor is seen as free, so we produce as many

actors as we need. That said, there are still areas where a job-based approach will win. For instance, any task where the existence of a common block of shared memory used in a read-only-fashion is a good idea. It may also be easier to schedule job priorities than it is to schedule actor priorities; in fact, most Erlang concurrency experts advise you not to mess with the priority of a process, for fear of gumming up the whole works.

Our implementation of the Erlang concurrency model in C++ fails to capture many of the attractive advantages of the Erlang language, but it does show that we can exploit some of the paradigms and ideas that Erlang uses. We lack language mechanisms to encourage defensive programming, and we lack the advantages of functional programming in reducing state that must be shared between processes. Maybe one day we will all be using Erlang, or a language similar to Erlang, and will laugh when we consider how we once used C++. It's not that far-fetched—I can still remember when we wrote everything in assembly language. We shifted to C (and eventually C++) when processing power became high enough that we could take advantage of it; now, we have multiple processors. It may be time for the paradigm to shift again. @

*NICHOLAS VINING is the technical director at Gaslamp Games, a small independent game developer that specializes in oddball entertainment and boutique technology development. He has not slept in ten years; other people with similar sleeping habits are encouraged to email him at [mordred@icculus.org](mailto:mordred@icculus.org).*

## resources

### Erlang Home Page

<http://ftp.sUNET.se/pub/lang/erlang>

**Erlang Programming: A Concurrent Approach to Software Development** by Francesco Cesarini and Simon Thompson (O'Reilly Press)

### Great Language Shootout

<http://shootout.alioth.debian.org>

### "Writing Lock-Free Code: A Corrected Queue" by Herb Sutter

[www.drdobbs.com/cpp/210604448](http://www.drdobbs.com/cpp/210604448) on 2010/03/01

### Intel's TBB library

[www.threadingbuildingblocks.org](http://www.threadingbuildingblocks.org)

### John Ratcliff's Jobswarm

<http://code.google.com/p/jobswarm>  
[www.codesuppository.blogspot.com](http://www.codesuppository.blogspot.com)

### Tim Sweeney's Talk

[www.cs.princeton.edu/~Edpw/popl/06/Tim-POPL.ppt](http://www.cs.princeton.edu/~Edpw/popl/06/Tim-POPL.ppt)



# GDC Canada

Learn. Network. Inspire.

Game Developers Conference™ Canada

**May 6-7, 2010**

Vancouver Convention Centre | Vancouver, BC

Visit [www.GDC-Canada.com](http://www.GDC-Canada.com) for more information





# FREE REALMS

## postmortem

L A R A L Y N M C W I L L I A M S



**FREE REALMS ISN'T JUST A NEW VIRTUAL WORLD FOR KIDS. FOR US, IT WAS A BIG EXPERIMENT—** we took a genre, a team, and a company in new directions. From the start, it was the open road. We had an original IP, a new engine, and a largely untapped target audience. In a relatively short development cycle, we created a virtual world that raised the quality bar for the “free-to-play” space. We devised a combination of microtransaction and subscription model, along with the combination of browser-based services with streaming assets in a stand-alone 3D client.

Developing FREE REALMS was consistently challenging and demanding for everyone associated with the project. At its peak, we had over 150 people on the development team, and at least that many folks working in support departments. Effectively guiding and managing that many people is a challenge in itself ... but FREE REALMS also launched with 15 job classes and more than 10 different mechanics, all of which had unique minigames, reward sets, quest lines, and progression paths. Almost all the systems and content were developed in the last 1.5 years of development.

This wasn't a small game SOE developed on the side while we focused on our other “real” games. FREE REALMS was a big bet for the company and for the team.

### WHAT WENT RIGHT

**1) EXPERIENCED TEAM LEADERSHIP.** Although development of FREE REALMS started in 2005 with a white paper by SOE President John Smedley, it struggled to get traction. In 2006, SOE added senior leadership staff to the project, and it immediately began to move forward. Every director on the team that launched FREE REALMS had 12+ years of game development experience. As the team grew, we brought in leads to help manage the tasks and the team.



Each lead had 6–10 years of experience shipping MMOs, RPGs, and console titles.

This track record was vital both to making big decisions at the foundation of the project, and to small, quick decisions as we developed the game. For example, there was an initial debate over player run speed and potential speed boosts from potions or wearables. Our technical director (Jamey Ryan) drew on his years of work on EVERQUEST to point out that the maximum player speed would increase over the years, because a boost in movement speed is one of the most valued benefits you can give players in an MMO. If we set the speed too high

early on, it would become unmanageably quick over the years—and more importantly, players would move faster than our streaming technology could hand them assets.

Balance of experience was important. While the senior producer (Andy Sites) and art director (Rosie Rappaport) worked on the original EVERQUEST, the lead designer (Stephen Bokkes) and I (creative director) came from a console/PC development background, and had made several successful games for kids. The mix of mileage and the cross-section of skills created the right kind of environment for production-oriented decision making and selective, careful innovation. For an online game of its quality and size, FREE REALMS was developed quickly and it went to beta within three months of the original target date.

**2) EMBEDDED RESOURCES FROM OTHER DEPARTMENTS.** During the development of FREE REALMS, we took a unique approach to creating our new IP, game, and service. The team absorbed key members from other departments into the development team as dedicated resources. At one point, the FREE REALMS team included members of SOE's marketing, customer service, community manager, web presence, and platform groups. They sat with the development team, attended our meetings, and were instrumental in FREE REALMS' smooth and successful launch.





# REALMS





# FREE REALMS

postmortem

The integration of marketing is a great example of this approach. Kids are savvy consumers, and they're inundated with high-quality advertising and marketing for toys, games, movies, and television. Not only is it a crowded marketplace for free-to-play virtual worlds—it's a crowded marketplace inside of every kid's head. We needed to make sure FREE REALMS was memorable, that it stood out among similar products, and that its first impression was on par with top advertising for toys and movies.

The marketing and development teams worked hard to coordinate our efforts very early into the project. While there was always tension over elements like branding and logos, we recognized those debates as a part of the creative process that would make us stronger as a team. Every piece of marketing material—from trade show booth art, to television ads, to viral marketing Flash games—passed through the development team for approval. In addition, the FREE REALMS art director worked closely with marketing to make sure everything "felt like FREE REALMS."

When you look at the marketing and advertising for FREE REALMS and then at the actual game, you'll see a consistent look and feel that's rare in game development. That carried through to other products like the FREE REALMS Trading Card Game and the Station Cash cards sold at retail.

**3) INTEGRATED SCRUM PARTWAY THROUGH DEVELOPMENT.** Traditional wisdom states that you should never change horses mid-stream. But what if your horse can't swim?

The FREE REALMS team used a traditional waterfall approach for over half the game's development. It was working about as well as it usually works in a complex scope, large team environment. With only about a year left in development we decided to move to agile development and start using scrum with a team of 80 people. We sent all the leads and key personnel to scrum training, asked those team members to train everyone else, and then made the leap.

From the start, there was a marked improvement in team morale and communication. Although



there were (and still are) a few holdouts who gripe about the daily meetings, everyone recognized that the game was making more progress in less time than it had before we started agile development. Over time, we modified some of the more traditional scrum elements as we added team members and features, but we maintained the heart of scrum: daily meetings, user stories going into a backlog maintained and championed by a product owner, and increased scrum group responsibility and ownership.

Sense of ownership is key to understanding the impact scrum can have on a large team. When you have so many different people working in parallel, it's easy for each individual to lose his sense of purpose—he starts to feel like a cog in a machine. You could definitely see that "just doing my job" feeling in parts of the FREE REALMS team before we used scrum. Many people on the team didn't understand the big picture, had no idea what other parts of the team were developing, or had lost some of the entrepreneurial spirit that started FREE REALMS off so strongly.

Scrum brought with it daily meetings and bi-weekly sprint reviews where we saw what everyone else had developed. Those meetings

were inspirational when we saw fantastic work from other teams, and embarrassing when one group's work wasn't quite up to par. The team morale woke up and the quality of the game improved along with productivity.

**4) STRONG COMPANY-WIDE BACKING.** FREE REALMS didn't just come from its development and marketing teams—it came from all of SOE. Although SOE has studios in six different locations (San Diego, Austin, Seattle, Denver, Tucson, and Taipei, Taiwan), we interacted with and obtained assistance from almost every studio and every department. Since the first time we showed "the new FREE REALMS" at SOE's internal town hall 2006 meeting (a gathering of employees from all SOE offices), we felt like we had a tremendous set of teammates helping us get to the finish line.

Part of that support came from departments being jointly responsible for running FREE REALMS as a live service, like operations, platform, customer service, and community. Additional support came from unexpected sources just when it was needed. Other teams offered seasoned employees to help relieve our team members during the final months' push to launch.





Many SOE San Diego folks brought their kids in afternoons and weekends for usability tests and feedback. Company-wide playtests were instituted, and employees in different time zones found ways to participate and post to the internal forums during our beta testing. Busy studio heads and creative directors of all the studios took time to play and send detailed feedback and offers of help as well.

This sounds like a lot of inner-circle cheerleading, but the point is that FREE REALMS would not have shipped on time or at such high quality if we hadn't had the support of the entire company behind it. Most game development today happens in studios with more than one title and more than one team. Those teams usually run separately and have "healthy" competition over resources, marketing spend, and attention from company decision-makers. Now that I've experienced what happens when you work as a team instead of competing, I can say without hesitation that more companies should find ways to bring all their experience and resources to bear for every project, and foster an atmosphere where everyone works hard to make sure every team succeeds. It's a night-and-day difference, both in terms of company atmosphere and the end result for the game itself.

**5) CONSTANT USABILITY TESTING.** The only thing worse than building a complex machine in the dark is when that machine is designed for tiny little hands and your big grown-up hands can't even work or fit the controls. We desperately needed to watch kids play our game! For the first half of development, we used traditional focus testing every 4–6 months and asked employees to bring their kids in to test when they could. We got decent feedback, and we were really grateful to have it.

Then we built the SOE Usability Lab. From the start, it had a massive effect on the quality and playability of the game. The Usability Lab has several play stations, each with two cameras (one on the player's face and one on the mouse/keyboard) and a capture feed from the screen. A moderator takes notes and the kids answer surveys. You can also view the session live from a viewing room or review the captures later across the company intranet.

The first few sessions proved the value of direct information from our target audience. After that, we booked out the usability lab with sessions 3–5 days a week. Having the facilities on site meant the kids could test in our internal development environment. It also meant we could test a feature on day 1, and incorporate changes based on the feedback for subsequent testing on day 2.

I can't overstate the importance of watching your target demographic play the game, especially if the development team isn't a part of that demographic. We encouraged team members to attend usability sessions, even to the point of responding to off-target designs by saying, "Go to tomorrow's usability test and you'll change your mind." Having friends and family test your game from the start is valuable. If you have the opportunity to create even a single station usability lab, take that step and you'll reap the benefits almost immediately.

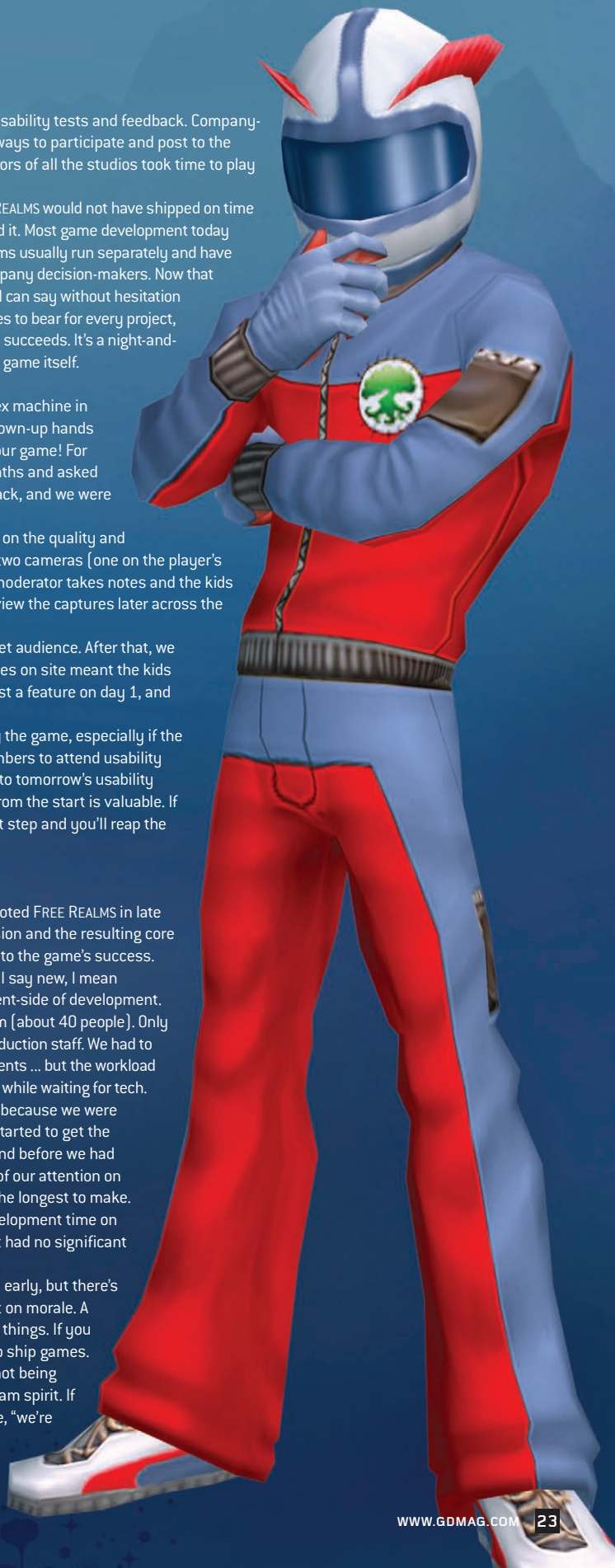
## WHAT WENT WRONG

**1) STAFFED THE TEAM BEFORE THE ENGINE AND TOOLS WERE READY.** We rebooted FREE REALMS in late 2006, and although it was controversial and extremely challenging, that decision and the resulting core architecture from our technical director and his team were a large contributor to the game's success. The reboot included shifting to a brand new engine and streaming tech. When I say new, I mean completely new—at least from the perspective of folks who worked on the client-side of development.

While the reboot was essential, FREE REALMS already had a decent-sized team (about 40 people). Only about 10 of those were programmers—the rest were artists, designers, and production staff. We had to plan the look and feel of the game, and write the systems and world/lore documents ... but the workload wasn't large enough to keep that many people productive for that many months while waiting for tech.

On top of that, engine and tools development took longer than anticipated because we were tackling so many new technologies (like content streaming). That meant we started to get the planned influx of staffing when the tools were still rudimentary and shaky—and before we had done any real iteration on key features and concepts. We focused 75 percent of our attention on art tools because the art department was the largest, and their content took the longest to make. It was the right decision, but the result was a very short amount of actual development time on the minigames and content with largely untried design tools and features that had no significant previous iteration.

All of that is what you would expect to happen when you staff a project too early, but there's a significant secondary effect that shouldn't get lost in the shuffle: the impact on morale. A game development team thrives on building things, not talking about building things. If you have experienced staff who are successful in their roles, these are people who ship games. It's what they do and who they are. Spending that long in pre-production and not being able to make anything they could consider "final" took a tremendous toll on team spirit. If you have the "luxury" of long pre-production cycles, schedule regular, playable, "we're up a creek if we miss this milestone" deliverables or you can watch team momentum spiral down the drain.





# FREE REALMS

postmortem

## 2) TOO MANY KEY FEATURES WERE FINISHED TOO LATE.

The mismatch of tool readiness and team size meant that many features and elements of the game came in later than we anticipated. For example, the back-end for microtransactions (both from the development team and from the support team) came online within a month of the start of internal beta ... and we weren't able to test it in any environment other than our live alpha server. Also, we'd been making mock-ups for the monetization UI for months, but once we had the technology, we scrapped all that work and re-designed it to suit what the system actually needed. We also discovered that the tools for item creation on the Station Cash Marketplace were almost unusable in a production environment, with no option for mass item creation in Excel or directly to the database.

These kinds of last-minute development surprises had two main negative effects. First, it meant the team's attention was critically divided when it needed the most focus. Elements like avatar customization with the final assets, itemization, and the marketplace are vital to the success of the product, yet we were not able to complete them until very close to the start of alpha. That meant some of the features that needed the most time to bake in QA came online when QA was swamped with checking a hundred NPCs, a thousand quests, and dozens of minigames. Itemization alone could have consumed the efforts of our entire QA team for a full month ... but it was first testable at a time when we couldn't afford to dedicate that many testing resources to a single feature.

Second, it meant we had the least usability, iteration, and bug-fixing time on some of our most important player-facing features. Just like the QA team's attention was divided, the FREE REALMS team's attention was spread across the game as a

whole. Although we dedicated entire scrum teams to particular features, those teams struggled to get the support they needed from team management and support groups. Firefighters can only put out so many fires a day; and our whole world was on fire at the same time.

You can see the result of this when you compare FREE REALMS now to the original launch product. Check out the marketplace now versus the marketplace at launch in Figure 1.

These improvements and additions weren't the result of some big light bulb turning on in our heads; they were the result of having user feedback and iteration time. Ninety percent of the information that led to these changes came from testing in our own usability lab and not from the live player base. If those features had been available for proper focus and usability testing before launch, we would have improved them earlier, and without a doubt we would have seen a significant increase in the game's revenue during the first year.

## 3) UNDERESTIMATED THE NEED FOR ITERATION ON BACK-END FEATURES.

The effect a lack of iteration time has on elements like user interface or items is clear—but what about character login protocols? Chat filters? Web profile updating? Change management and migration across development, test, and live environments? Localization and live string updating? Item stack counts?

The trickle-down effect of the engine reboot meant that we were always robbing Peter to pay

Paul. It seemed like the right choice to get back-end features out of the way ASAP and focus on player-facing features as much as possible. So although we spent the time and effort to develop the new engine, and then to develop FREE REALMS' core features on top of that, we didn't schedule enough time for those systems to go through test and iteration time before we started using them.

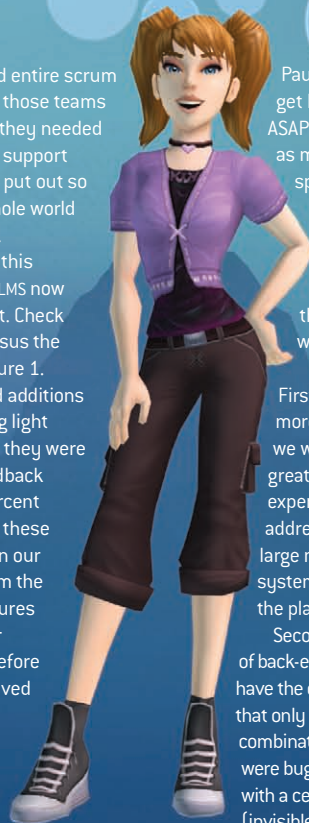
That had two effects on the game. First, it meant we shipped our beta with more back-end and systems bugs than we would have liked. FREE REALMS had a great, stable launch, but that came at the expense of a large team of people actively addressing new bugs 24-7 for months. A large number of those bugs were in core systems and tech, which distracted us from the player-facing bugs.

Second and more importantly, the lack of back-end iteration time meant we didn't have the opportunity to find the kinds of bugs that only occur with accumulated data and a combination of attributes. For example, there were bugs that only occurred when a character with a certain number of items crossed [invisible] zone lines with a pet following him. If we'd had the time to iterate on back-end systems, we could have discovered and addressed that bug earlier in the process (in addition to addressing some related data storage size issues that ended up affecting other areas of the game).

**4) LACK OF DATA REPORTING AT LAUNCH.** For a variety of reasons, our plans for robust player data logging and reporting didn't work out. The result: we went the first 3-4 months after launch without the ability to easily consolidate and consider data about what FREE REALMS players were doing. We had reporting for specific parts of the process (like creating an account on the web) and data from logs we could manually sort and aggregate into reports (a very time-consuming process), but nothing simple, clear, and quick.

If you work in the online space, you know how important it is to have accurate and timely player data. If you don't, imagine trying to give docking instructions to astronauts when all your control panels and monitors have gone completely dark and you're doing the trajectory calculations with a notepad and calculator. You may get the right data, but it will be critically (and in some cases, fatally) delayed.

We tried to take advantage of opportunities to outsource the data reporting, but the game team should have always felt—and taken—responsibility for data logging and reporting. It's our game—we know the development structure and the goals better than anyone. In the heat of the last six months of development, we were



GAME DATA

## FreeRealms

<b>PUBLISHER</b>	Flash, Illustrator, Photoshop, DreamCoder, Active Perl
Sony Online Entertainment	
<b>DEVELOPER</b>	60,000 and growing
Sony Online Entertainment	
<b>NUMBER OF DEVELOPERS</b>	8,000,000 and growing
Approximately 150+ at peak	
<b>LENGTH OF DEVELOPMENT</b>	<b>TOP FIVE MOST POPULAR ITEMS IN MARKETPLACE LIFE TO DATE</b>
4 years total, including 1 year of pre-production	1 FREE REALMS Digital Booster Pack
<b>RELEASE DATE</b> April 28, 2009	2 Pipsqueak (Penguin Pet)
<b>TECHNOLOGY</b>	3 Skeletal Hoodie
Kynapse, Scaleform, Miles Sound System, Enterprise DB	4 Munchy (Dino Pet)
<b>SOFTWARE</b>	5 Humongous Health Potion
Microsoft Developer's Studio, C++, Maya,	<b>PLATFORM</b> PC, Online







relieved that we might be able to set a part of that burden aside to focus on other tasks. Instead, we should have taken the time to ensure we'd have the logging-and-reporting structure we needed.

One of the biggest pieces of advice I can give to teams who are just starting to create their virtual world, MMO, or online service is to build in data reporting and logging now. Build it before you even have a client. You can get great and useful information from your reporting system before you have a single player. Even beyond the fact that you'll get useful data, consider this: When you're six months from launch and you're looking at an unfinished housing system, combat that needs a re-design, items that need a tinting system, tools for CS that haven't been started yet, and an E3 demo that has to be hands-on for the floor in eight weeks, do you really think you're going to stop working on any of those to dedicate your most experienced coders to writing robust server logging code and automated Excel reports?

**5) STRUGGLED TO MAKE THE SHIFT FROM TRADITIONAL MMO DEVELOPMENT TO CASUAL VIRTUAL WORLD DEVELOPMENT.** SOE is a flagship studio for MMO development. EVERQUEST is going into its 11th year as a live service. We have an unprecedented depth of experience in online world design and development.

That's also a lot of history and habit to overcome when you try to make something new. Even with a huge amount of team enthusiasm for the concept, phenomenal support from the entire company, our seasoned leads and directors, we struggled as a company to overcome all our ideas and preconceptions about the way an online game "has to work."

You get a mixed message from the launch version of FREE REALMS. At its heart, FREE REALMS is about doing what you want when you want to do it, in the way you want to do it. Don't like playing a Medic? Try a Ninja! Not into combat? Level up as a Miner by playing a match 3 game! The core of FREE REALMS works, and focus/usability tests along with player data show us that when our target audience gets to that core, they have a great time.

Unfortunately that tasty "do anything you want" core is surrounded by elements that, while normal in other games, in FREE REALMS become MMO detritus. Want to buy a specific item? Wander around the 3D world until you happen upon the only guy who sells it. Feel like leveling up? Then go find some quests and finish them because that's where the XP is (not in the minigames



FIGURE 1 shows the evolution of the FREE REALMS marketplace from launch (top) to its current iteration (bottom).



# FREE REALMS

postmortem



FIGURE 2 shows the welcome screen for FREE REALMS that announces new features and new items for sale.

themselves). Can't find the location of a battle? Well, hover over every icon on the atlas until the tooltips reveal it, and then put on your walking shoes.

Still, FREE REALMS got rid of a lot of the work and tedium that comes with playing many MMOs. When you defeat an enemy, everyone in your group gets the reward. You can teleport to any city in the world or directly to a friend just by clicking on the atlas, and you don't have to use a calculator to figure out what pair of pants to wear.

We didn't go far enough down that path though, and similar to important features not having enough focus, the ongoing attempts to overcome our strong MMO background are obvious in the changes we've made to FREE REALMS after launch. We took baby steps for sure. First, the Take Me There button would automatically run your character to a destination in between major cities. A couple months later, we let you teleport directly to any activity on the atlas. Most recently, we added the Game Guide which lets you start an activity without having to move to its location, along with the Coin Shop which lets you buy items


for coins without having to visit a vendor. We'll get there, and we're learning along the way, but we'd have a stronger, more cohesive game (and higher revenue for the first year) if we'd stayed closer to our conceptual goal.

## WHAT DOESN'T KILL YOU MAKES YOU STRONGER

» It's easy to look back at the development of this ambitious, experimental, and challenging project to see how our choices worked out. As I write this, FREE REALMS just celebrated the milestone of nine million registered players, and our uniques, retention, and revenue have been trending steadily upward.

It's also easy to see how the lessons learned on FREE REALMS changed our development processes. The FREE REALMS engine and technology is being used for two unannounced games in development here at SOE, and team members from FREE REALMS seeded both those teams. Those new teams have started not just with knowledge and experience, but also with a solid engine and tools. The improvements and additions they make to the code base and systems will be added into the core for the benefit of other teams. We saw what happened when you have a team waiting for technology, so we're committed to giving every team room to iterate on gameplay as quickly as possible.

You can also see the influence of FREE REALMS in our other games. For example, both EVERQUEST and EVERQUEST II added a welcome screen to advertise new features and items for sale in the marketplace (see Figure 2). Our streaming technology is also being used very successfully to stream content for a small download trial version of EVERQUEST II.

We still do usability testing on FREE REALMS three days a week. We make a tweak, look at the player data, then tweak it again. We share that data with other teams and work together to broaden the market for online games. We know nine million players is just scratching the surface! By far, that's the biggest "what went right" we could have imagined. We've learned how much we still have to learn. 

**LARALYN MCWILLIAMS** was creative director for FREE REALMS, and previously worked on FULL SPECTRUM WARRIOR, as well as two film-licensed games for boys. She is currently working as the senior producer on an unannounced project for SOE.







# Unreal Technology News

by Mark Rein, Epic Games, Inc.

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

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

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

## Upcoming Epic Attended Events:

**Triangle Game Conference**  
Raleigh, NC  
April 7-8, 2010

**E3 2010**  
Los Angeles, CA  
June 15-17, 2010

Please email:  
mrein@epicgames.com  
for appointments.

## AUTODESK'S FBX MAKES IMPORTING 3D CONTENT INTO UNREAL ENGINE 3 A ONE-CLICK PROCESS

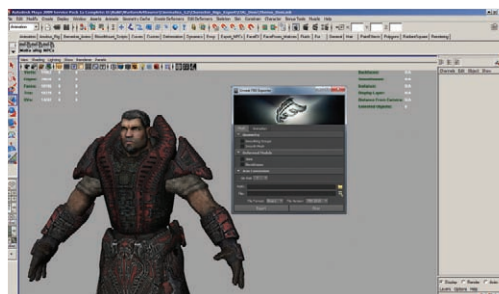
Importing 3D content is now just one click away. Autodesk and Epic are now providing greater connectivity between Autodesk's art creation and animation tools and Epic's Unreal Engine 3, using Autodesk FBX data interchange technology. Autodesk FBX 2011 offers a faster, more streamlined workflow for transferring content created in Autodesk Maya 2011 and Autodesk 3ds Max 2011 software into Unreal Engine 3 – boosting production efficiency and preserving creative intent.

The streamlined interchange of 3D assets between Autodesk art creation tools and Unreal Engine 3 is the result of a long-standing, productive relationship between Autodesk and Epic.

The Autodesk FBX file format is a robust standard for rich 3D data exchange within the games community. With this streamlined workflow, FBX should be the first choice for developers using our powerful Unreal Engine 3 when it comes to transferring art from Autodesk software.

New for Unreal Engine 3 licensees and Unreal Development Kit (UDK) users is an FBX importer that enables game developers to import FBX files created in Maya or 3ds Max directly into the Unreal Editor. The importer automatically breaks files down into assets in the Unreal Editor, such as level of detail (LOD) information, animations, character meshes, character rigs and models. In addition, if a 3D model is updated in 3ds Max or Maya, a new FBX file can simply be imported into the Unreal Editor where the assets will be automatically refreshed.

Marc Stevens, vice president of Autodesk Games, said, "We're really excited about our collaboration with Epic Games. We've teamed up to provide greater connectivity between art creation tools, animation tools and middleware from Autodesk, and Epic's Unreal Engine. This is made possible through advances in Autodesk FBX data interchange technology and better integration of our middleware into the Unreal Engine. In short, our combined efforts are aimed at giving game developers a faster, easier, more streamlined workflow that boosts production efficiency and preserves their creative intent."



Maya's Unreal FBX Exporter

## STEAMWORKS INTEGRATION NOW AVAILABLE TO UNREAL ENGINE 3 LICENSEES

### DID WE MENTION IT'S FREE?

Epic and Valve are teaming up to deliver Valve's Steamworks suite of services to anyone who's licensed Unreal Engine 3 for use in its products, free of any additional fees.

Everybody knows what an awesome distribution network Valve has created with Steam. The Steamworks suite provides developers with the tools to make their games closely integrated with the cool features offered on Steam. We're excited to be able to offer Steamworks integration to our customers as a standard part of Unreal Engine

3, free of charge. We're big fans of Steam and our games have been very successful on the platform so it was a no-brainer to bring Steamworks and Unreal together.

"Unreal is one of the most widely used engines in the industry, period, and it's been behind the scenes on some of the very best games created over the past 10 years, on all kinds of platforms," said Gabe Newell, co-founder and president of Valve. "It's an honor to have Steamworks included in the technology offered to all Unreal Engine 3 licensees. It's hard to think of any community of developers who could get more from all the services that come with Steamworks."

Steamworks is a complete suite of publishing and development tools that offers PC game developers and publishers access to the game features and services available through Steam. These include product key authentication, copy protection, auto-updating, social networking, matchmaking, anti-cheat technology and more. The features and services available in Steamworks are offered free of charge and may be used for both electronic and tangible versions of games. For more information on Steam, please visit [www.steamgames.com](http://www.steamgames.com).



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

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

WWW.EPICGAMES.COM



# DAVID CRANE the one-man show

**DAVID CRANE IS ONE OF THE STEALTHIER LEGENDS OF THE GAME INDUSTRY.** He doesn't make a lot of noise, or push big blockbuster releases. What Crane has done over the years is quietly innovate, bringing new concepts to bear decade after decade. His biggest successes were in the golden age, with *FREEWAY*, *PITFALL*, and *PITFALL II* moving big numbers in the early '80s—some of the first successful products from Activision, which he helped form in 1979.

*PITFALL* is considered by many to be the first action platformer, and its influence can be felt as far as the *TOMB RAIDER* and *UNCHARTED* franchises. In 1985, Crane, along with designer Rich Gold, created *LITTLE COMPUTER PEOPLE*, a life simulator which had no small amount of influence on *THE SIMS* series. Crane left Activision the next year to form Absolute Entertainment, where he created the venerable *BOY AND HIS BLOB* for the NES. In the '90s, he helped code the infamous *NIGHT TRAP* interactive movie for Digital Pictures, one of the more lawsuit-afflicted games of the era.

In the late '90s, Crane and his partner Garry Kitchen formed Skyworks Interactive, creating the first major advergaming portal in Candystand. Now, he's moved on to the iPhone platform with his new company AppStar Games, creating small, bite-sized games, which has been his passion since the early days. Indeed, though he has not been at the forefront of the HD era like some of his contemporaries, Crane has never stopped questioning the nature of games, or the industry. And that is what we explore today.

**BRANDON SHEFFIELD:** I would like to start by talking in general about the mindset during the 2600 and Intellivision era. Obviously, everyone was in a one-man show. Where were these people coming from? There wasn't really training necessarily, right?

**DAVID CRANE:** Yeah, in what I call the good old days, one man, one project. It was kind of interesting that the people who could do that came from a number of different walks of life. At Atari, for example, just if you look at the four of us who ended up starting Activision, two were trained as electronic engineers, which helped us understand the hardware, and we came at it from that direction. Two were trained as computer science majors. And Larry Kaplan went to Berkeley, you know, which was not a hotbed of engineering. He learned computer science there. So, everybody came at it from a different direction, but it didn't take too long for you to see whether the person had both the technical skills and the creative skills to be able to do all this. It is very much a left brain, right brain thing. There are a lot of computer nerds who can be very, very technical, and there are a lot of very creative people. These days, they would simply gravitate toward their specialty and be part of a team, but back then, it was kind of rare to have people who had both capabilities. If we set out to try to find one, it's not an easy thing



to do. You would have someone who's very much interested in this kind of thing and throw them in the fire, and find out if they could do it.

**BS:** Where did the innovations in the 2600 hardware come from, considering that it was designed to do very little? How were people figuring out that there was more you could do with it?

**DC:** Well, what was really interesting about the 2600 is, yes, it was designed to play *TANK* and *PONG*. And to do that, there were two players and two missiles for *TANK* and a ball object for *PONG*, [in which case] the player objects became the paddles in *PONG*. And they made it cartridge programmable, where you could remove the cartridge, just so they could sell two cartridges, *TANK* and *PONG*. And even at that, the

hardware was too expensive. There was too much stuff going on in the chip. So, they went through a value-engineering period where they removed as much as they could from the circuit design and put the onus on the software and the microprocessor to do those things that otherwise they would have put in the hardware. Because they had to have a microprocessor and they had to have a video chip, but the more they put in the video chip, the more expensive the hardware cost, so they stripped it out.

The beauty of that is if the microprocessor and its associated program have more and more control over what happens on the TV screen, and it's the program that is removed when the cartridge is exchanged, you're basically having much more power over the video because the video chip was stripped down. So, the serendipity there was just a wonderful thing. The reason the 2600 could do so many different kinds of games is because it couldn't do anything very well at all, but the microprocessor program could control video. If you know how the thing works, there's no bitmap, there's no objects ... Really, there are no objects! There are 8 bits for each project, and if you change them every scanline, you can make them look different. That was a premise going in. You knew that you were going to have to change every scanline

going in to make a game display, so it wasn't too difficult to say, "If I have to change it every time, let's change it this way, let's change it that way, let's try this, and let's try that." A lot of experimentation gave you all those different kinds of game displays.

**BS:** What language were you using?

**DC:** It was all written in Assembly language with a microprocessor.

**BS:** I always like to ask people who have been around their thoughts on the crash and why.

**DC:** The video game crash of 1980... Was it 3, 4? I don't even know.

**BS:** 1983, I believe.

**DC:** It was very interesting because we should have seen it coming. Basically, it was a very simple combination of events. Activision was extremely successful as a third-party developer of video game software. And then there was Imagic, which was another group of people that we had trained at Atari—they were the younger guys that we had trained—and they as a group looked at Activision's success and said, "We're going to do this." And with two companies clearly showing success, the venture capitalists came in, and they said, "Well, we want a piece of that." In one six-month period between two CESes—there used to be two a year—in the one six-month period, 30 new game companies showed up.



**BS: Wow.**

**DC:** They hadn't been there in the previous one. When we saw those, you know, we had gravitated to the top of Atari in game design, and the second group, the Imagic group, was right up there as well, but all of these new companies had no game design talent. Where do you get it?

And it takes a while to develop that, to create a new group of guys, and instead they were just hiring programmers off the street. So, we saw that, and we looked at the games, and we said, "You know, none of these companies are going to be around a year from now. They're all going to lose their shirts." And we didn't take the intuitive leap one step further and say, "And when they do, there's going to be a crash in the video game business." It was very simple. What happened was those venture capitalists put a couple million dollars into each of those 30 companies, they developed three or four games that nobody wanted to buy, and they built their first run of cartridges, and they went out to sell them, and they didn't sell.

No money, the company closes. Now there's a warehouse full of games. And some enterprising—we'll call them the bottom-feeders—came rolling in and said, "I'll tell you what, I'll give you \$3 each for those \$30 cartridges." And they took them to retailers and said, "I'm going to sell them to you for \$4, and you can sell them for \$5." There's a lot of good margin there if you really think about it, because they have no cost of goods. In those days, you would walk into a Toys "R" Us or KB Toys, and right in front was this huge barrel full of video games, and they were \$5 each.

That Christmas, dad walked in with his Christmas gift list from his kids. "I want Activision's latest this, and I want all these latest games." And he sees this barrel and says, "I was going to spend \$40 on a new game. I can get eight games

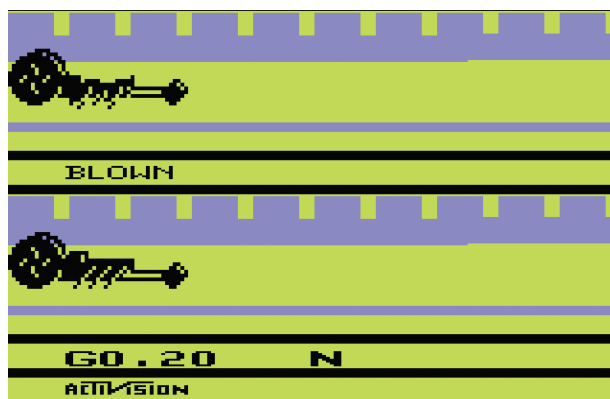
for that same \$40, and I will be the hero come Christmas time." So, in that Christmas, the sale of frontline brand new games went to zero, and there were 20 million of those cartridges, the \$5 cartridges, that flooded the market. And until those all sold through, you weren't going to sell a \$40 game again.

**BS: I suppose not. A lot of guys that were part of that one-man show situation are coming back around to a more social or casual game-oriented platform. Richard Garriott is doing it, and Richard Woita (QUADRUN). And you're doing iPhone. A lot of guys are making that kind of transition.**

**DC:** Well, I think if you actually follow any of those people in the last 25 years since that crash, they haven't been doing nothing. They've been doing pretty much the same thing. I mean, I continue to make games with that small footprint because then I can program every line of code and I can control everything. I've been involved in big projects, and I like the smaller projects. They're just more fun for me with my skills. But now that they're becoming more trendy and more publicized, the social gaming and the casual gaming, now you start to hear about the work that the people have been doing the past 25 years. It's not like they just disappeared and they're, you know, re-emerging.

**BS: Right, I didn't mean to indicate that they were gone and coming back, but perhaps it's more that there's a larger focus on it now. Like Garriott, for instance, is in fact coming back to that because before he was just going bigger and bigger.**

**DC:** Well, those of us who did that really loved it. Again, if you could control every line of code, if you could control every detail, you can make the game that you want to make. On the big projects, the people who could do it in the 2600 days are very valuable because it's... George



PITFALL II: LOST CAVERNS (TOP), DRAGSTER (BOTTOM).

Lucas, Steve Spielberg—I mean, they could sit behind the camera if they had to and make the right shot. They can direct, they can produce, they can do all those things. Now, it's not cost-effective for them to do that themselves, so in a big team, they would be the director or the producer, depending on what their interest is, and they would have specialists in every field—the cameramen, the sound guys, whatever. With big projects, it's exactly the same way. Someone who has proven that they can do it all themselves is really good at directing. I should say in most cases they're really good at directing. If they don't have any people skills in all, then they have that problem.

**BS: And then they're back to a one-man show!**

**DC:** That's right. But, you know, in general, that's the case. So, the people you talked about who have gone into the big projects but now are coming back to the small projects, it's because those are much more fun. They clearly loved it back then. They're going to love it now, you know.

**BS: What language are you programming in now?**

**DC:** I'm working on the iPhone so I program in Objective C. I recently did a tally and came up with 15 computer languages that I've done video games in.

**BS: Wow. What would you say is the best of them? I've heard some people saying that C is no longer appropriate for game development.**

**DC:** You know, I... You use the



# david crane the one-man show



language for the platform you're working on really. Whatever the best tools are that exist for it. And, you know, for those of us who can program, if you can program in three different languages, you can program in 50. So there are people that have their favorites, and they don't like a particular integrated development environment for a particular language or a particular system and would like it to go in a different direction. A lot of that is just grumbling.

**BS: What do you think about the current proliferation of tools from the lower side like Game Maker and ActionScript libraries for Flash, or even larger things like Unreal Engine 3 there that can enable you to do a lot quite simply. Possibly, people could become one-man shows again without as much skill necessarily as in the past.**

**DC:** I think middleware and engines are great. Some of the most creative and brilliant guys gravitate in that direction because they can

really make a difference. You know, there's a little bit of god complex in anything we do. I mean, we basically create the world. [laughs] There is no gravity until we tell it there's gravity. And the same with all the physical laws. Like I said, the best people tend to gravitate to those. There are some fabulous pieces of middleware and systems out there. I'm not sure that really... If your question implies that it's going to make it easier for some people to do that without also kind of understanding those things and almost having those skills, it is almost, like 95 percent of the time, as complicated to use some of those tools as to write them. But the specialists have gone into it and done it so much better and so much cleaner. You still have to know how to use that kind of stuff. I mean, you really have to think in 3D to do a 3D game.

**BS: Yeah. I've been wondering if the kind of removal of some of the technical constraints ... obviously,**

**constraints are different and much more complicated with a PlayStation 3 or even an iPhone than they were with the 2600. But with the 2600, you really had to like smash your head against it to make it work. I wonder how removal of technical constraints changes design.**

**DC:** It actually makes it more difficult.

**BS: Oh yeah?**

**DC:** If you sit down with a blank sheet of paper and say I can do anything in the world, it really makes it tough to say, "Well, where do I start?" The 2600, for me, I would sometimes in between games fiddle for a couple weeks, and I would experiment with the hardware and say, "Oh, here's a cool new thing I can do with it." And as soon as I see that, it leads me in a direction, and I say, "Alright, I'm going to do a video game based on this cool technique I found. It's always been more difficult when you have absolutely no constraints to decide, "Where am I going with the next game?"

Sometimes, there's an inspiration. *FREEWAY* was ... At CES, I'm in the bus going to CES, and some guy is trying to run across ten lines of traffic on Lake Shore Drive. You'll see an inspiration, and if at that point, that was the game you were going to do, the limitations of the machine are not an issue. But when you're sitting there, if you haven't gotten an inspiration yet, it is very difficult.

**BS: I was talking to Don Daglow about something similar—he was a big Intellivision guy—and for him, it was more like, "You know, you can make a game about anything you thought was cool at that time." You'd say, "You know what I like? Jet skis! Time for a video game."**

**DC:** Right, yeah.

**BS: But it's interesting to hear that it could also come from the technical side. When you don't have ideas, where do you try to get them from? Are you ever in a situation where you feel like, "I need to do something right now. I need an idea"?**

**DC:** Well, I think they're coming to me every few days. Even if I'm in the middle of another project, I will see something, and I just kind of store it away, not consciously. And then you reach this point where you're thinking about the next project, it helps if you can brainstorm. I work with my partner Garry Kitchen. We work together really well. We get together, and just a couple weeks ago, we were walking down the street walking his dog and talking about this actually. And we were just saying, "You know, the thing I talked about a few weeks ago? I got to figure out a way to use such and such in a game." So, then we'll chat about it a little bit. "Yeah, that's something that's worth more thought." Then you think about it and you work it out. It's just kind of an ongoing process, but quite often, the sparks



are sitting there from a month ago, two months ago, or six months ago, but you're so busy and embedded in game development until you release, that they just sit there and they don't percolate until later.

**BS:** I wonder if consistently doing, I don't want to say simpler, but like immediate gratification-type games, if that builds up a library of ability in that genre because I see some people trying to make what we would now call a casual game coming from a more hardcore mindset. And they can't help but layer in features and put more stuff in there.

**DC:** More than that, the best casual games have a gameplay mechanic. There's something about them that's interesting—how you're moving something around or how you're interacting with things on the screen. The big story games are more ... They're real world. The person writing the story game could be writing a movie with the exact same skills. When you're trying to find something that's fun to do, it's more like toy design. I mean, who would have thought that putting a ball on a piece of elastic and attaching it to a paddle would be something that was fun. But that became a fun play mechanic, and it became a toy. The slinky, hula-hoop, whatever. It's a way of thinking about, "Here's a fun that you can do electronically."

**BS:** So, you're pretty much talking about mechanics design versus like experiential design. You're not necessarily creating this world so much as creating a game around something that's innately fun to do.

**DC:** That's generally true. I mean, in the old days, there would be both. But right now, a casual game with a story, it's kind of ... It kind of falls in between since you can, on your console or whatever, play such spectacular stories and live them.

**BS:** How often, or in fact can you ever

recycle these mechanics? Because with a 30-some-odd-year career, as you've had ... someone like me may remember all those games, but not everybody will have experienced all those mechanics before. Do you feel like you can revisit them?

**DC:** Occasionally. You know, you will specifically say, "That was just one of the most fun games, and it wasn't a really common play mechanic." And you'll say, "Well, how can I re-capture that same fun in something that's more up to date?" But more often, the experience of 30-some-odd-years comes into play in the middle of a project because there are a thousand decisions that you make that are gameplay-related while making a game. The one thing that I'm personally proud of is I'll be in the middle of a project and I will have something on the screen and I will say, "How do I make this fun?" And five minutes later, I've got a great idea. It's just something that I do really well. You know, I code that, which might take a day or two, and then I'm playing it, and I say, "Yup. I was right. That's fun."

**BS:** In the early days, there wasn't so much playtesting or focus testing really. I wonder if that breeds a different mindset, because nowadays there's like aggressive playtesting against user opinion and "can people even understand this?" I mean, obviously we're talking about simpler mechanics with a casual game, but do you think it's different when you're relying on your own intuition?

**DC:** Yeah. It was always that way. The way we thought about it was that we were our own target market, so we were our own focus group. We had a design group where we would kibitz about each other's games, and so all our combined experience was going into one game. And if it wasn't fun for all of us, it never hit the market. It

never saw the light of day. So, we were focus testing with a very core group, but it just happened, in the Activision days, that we had pretty wide ranges of interests. So, we represented a good portion of the population, which is why millions of people seemed to like the games that we liked.

**BS:** As you advance in years, do you feel that you are still making games for you or that you need to make games for someone else?

**DC:** I'm still making games for me. It's interesting because the target market is aging as well. And so it's pretty easy to hit the market even in my advancing years, as you put it. [laughs]

**BS:** [laughs] Sorry! I mean, in fact, like the biggest market for casual games is women over 40 and all that. I wonder if in fact it might make you better at designing for the market.

**DC:** Well, it's probably a mistake to say that the biggest market for casual games is women over 40. The biggest increase in the market brought about casual games was to bring women in, but you would be amazed at the 18-to 25-year olds who love casual games. They just also play the hardcore games when they have the time. A lot of the work that we've done over the years for games on websites and other things that I've been doing for the last 20 years, whenever we do the analysis, we find out that most of the people that are playing them between 9 and 5 are from the office on the T1. And that age group is, you know, 20 to 50 and a lot of men.

**BS:** That's interesting. I have a friend who works for a casual game company, and they have one of the biggest portals. Their numbers are absolutely skewed toward women over 40. It's a very large percentage of their market, and I mean, who would have predicted the rise of the

**Spot the Difference genre?**

**DC:** Yeah, exactly.

**BS:** That's crazy. I would never think that was a game.

**DC:** And the match 3 games. Just all of those. It brought in a whole new market segment. And there are companies that really focus on those. But I think the best thing is it brought them into saying, "I'm comfortable playing a game on the computer." So, you can give them a casual game. You give them 30 seconds or two minutes of enjoyment, well, they're waiting in line at the bank.

**BS:** I have often felt that many of the best games, from the smallest to the largest, are those that have a specific vision led by a personality. Games that are created by people who make games for themselves, those wind up resonating with me in a different way than something that's designed for everyone, because I can see some auteurship and I can see some kind of intent.

**DC:** What you're seeing is ... It's the way in which all the details work together. You have thousands of decisions that have to be made to make a game, and if those thousands of decisions can all be made in one brain, than they all interoperate perfectly. If they're made by 15 people, and you try to come back together and combine them by committee, you don't get the same vision. That's really why guys who did in the old days, the guys who are able to keep all the details of a single game in their mind at one time, if they are the driving force vision behind a game of any size, it will be much more consistent. I mean, you have to have that world or that level over there dealing with what this one over here implies. It's difficult, but you'll find those games that could theoretically have one guy's name on them even if there's a hundred people. They will be much more self-consistent. 🧐



Supported by



European  
Games Developer  
Federation

gamescom

BIU 

**GDC Europe returns to Cologne in 2010**

# GDC 10 Europe

Game Developers Conference® Europe

**August 16-18, 2010**

Cologne Congress Center East | Cologne, Germany

Visit [www.GDCEurope.com](http://www.GDCEurope.com) for more information





# REVIVING NON-REALISTIC FIRST PERSON SHOOTER DESIGN LESSONS FROM

ARTICLE BY JEAN-PAUL LEBRETON  
ILLUSTRATIONS BY DEREK YU

**A HIGH-MINDED GOAL LIKE EXPANDING** the boundaries of the medium doesn't always mean forging ahead in crazy unknown directions. Sometimes it means examining lost evolutionary lines in game design—picking up ideas that were abandoned long ago and seeing if there's any new life in them. The game I keep coming back to in this regard is DOOM. Not the 2004 reboot, but "Classic DOOM:" DOOM 1 and 2, FINAL DOOM, the MASTER LEVELS and its vast universe of user-made content. What can this series teach us today?

In 1993, the message DOOM sent to the video game world was something like "use cutting edge technology to make something dark, edgy, and violent." The world has changed so much since then that very little of that original impact comes through to players today—the industry has arguably gone on to master the techno-fueled ultra-violence that DOOM put forth. Here's what I've found after many years of enjoying the game and digging ever deeper into its design. »





# DOOM

## DOOM FEELS MORE LIKE FIRST PERSON ROBOTRON THAN A MODERN FPS

When you play DOOM today, it doesn't feel much like you're controlling a human or moving through real spaces. Try this though: press the TAB key, type IDDT twice, and pretend you're playing GEOMETRY WARS. The moving triangles are your enemies (see Figure 1). This is what DOOM's designers were working from in 1993. Back then, the idea of a first-person shooter was barely established. Their closest models for many mechanics were 2D shooters like ROBOTRON, BERSERK, and TEMPEST. This approach echoes throughout DOOM's design. Any semblance of realism in the FPS genre wouldn't appear for another few years, and many of DOOM's decisions were made simply on the basis of being good for abstract shooter gameplay.

Partly thanks to this, many parts of DOOM's "game feel" still compare favorably with modern twitch games. Enemy speeds and patterns are very finely tuned, weapon design is strongly orthogonal, player movement has a nice friction to it, and level design elucidates all of this. QUAKE 3 is still considered the pinnacle of arcade-style FPS movement and feel, and that lineage starts with DOOM—even some of the code is similar.

## DOOM IS ABOUT "MANEUVERABILITY AS DEFENSE"

In almost every modern FPS, the player moves fairly slowly, and a huge proportion of enemies are equipped with instant-hit attacks: pistols, machine guns, sniper rifles, and the like. This usually puts the player in the role of "damage sponge"—they're intended to soak up a certain amount of damage from mostly unavoidable enemy attacks, then seek cover and heal up. HALO's recharging shield makes this mechanic quite explicit—by default, you're exposed to damage and will die, while seeking cover halts that and completes the basic cycle of any combat.

Contrast all this with Doom Guy, who runs at about 50 scale miles per hour—nonsensically fast

by modern standards. Most of DOOM's enemies don't have instant-hit projectile attacks, and most of the ones that do (the lowly trooper and sergeant) are quite weak. Every other enemy projectile takes time to reach its target, and would look comical in a more realistic visual presentation.

Because the player moves so quickly in DOOM, and because most enemy attacks can be avoided, the player can avoid a significant amount of damage simply by moving. A skilled player can often deal with large numbers of enemies and come out with hardly a scratch (see Figure 2). This creates an experience that's quite rare in modern FPS—a feeling that you are powerful because you are agile, not because you're a tank. This frees up DOOM's encounters to feature huge numbers of enemies, to vary scenarios by mixing in different proportions of threats, and to have huge, sprawling, often non-linear spaces that the player can traverse easily. There's nothing quite like it today.

## DOOM HAS A MORE VARIED BESTIARY THAN MOST MODERN FPSES

In many modern FPS titles, the design of every enemy the player faces is sampled from a fairly narrow tactical spectrum—soldier with machine gun, soldier with shotgun, zombie with melee attack. DOOM, on the other hand, has a huge range of monster sizes, speeds, strengths, and movement/attack patterns. Former humans and Imps are slow-moving, ranged fodder. Hell Barons are large, tank-like threats. Flying enemies range from the small, charging Lost Soul to the tough, fireball-belching Cacodemon. Revenants and Mancubi launch homing and spread-fire projectiles respectively, and the three boss-class monsters are each very dangerous in different ways. Some enemies can be stunned by weapon fire more easily than others.

Such diversity creates a large but simple-to-understand toolset that allows for level design to combine with architecture to create a huge

## resources

**DOOM User-Generated Maps**  
[www.doomworld.com/idmaps](http://www.doomworld.com/idmaps)

**Oblige Random DOOM Level Generator**  
<http://oblige.sourceforge.net>

variety of combat setups. One tough guy with a lot of fodder means the player has to do crowd control while focusing on the real threat. Lots of flying enemies make the player seek low cover and choke points. Enemies with strong melee in tight spaces make the player dance and really exploit the stun properties of their weapons. This versatility of the core design makes life easier and more fun for the level designer, and thus the player.

## DOOM WAS ABSTRACT IN WAYS THAT EMPOWERED ITS LEVEL DESIGN

While some of DOOM's levels have a very thin fiction via their title (e.g. "Hangar") and general texturing theme, if you actually explore them, you will find they only resemble real locations in the loosest sense possible. This is precisely what allowed DOOM's level design to present a wide variety of interesting tactical setups. Level designers didn't have to worry about whether a change made something look less like a hangar or a barracks, just whether it was better for gameplay. This was especially critical for a style of game that was just finding its feet in 1993.

As the march of technology has allowed ever-higher graphical fidelity, virtually every FPS since DOOM has attempted greater and greater direct representation with its environments. While games like SYSTEM SHOCK began to show that a real sense of place can be a huge draw in itself, designers of such games will always have to manage the tension between compelling fiction and optimal function, unless they are willing to go all out and have the kind of weird,

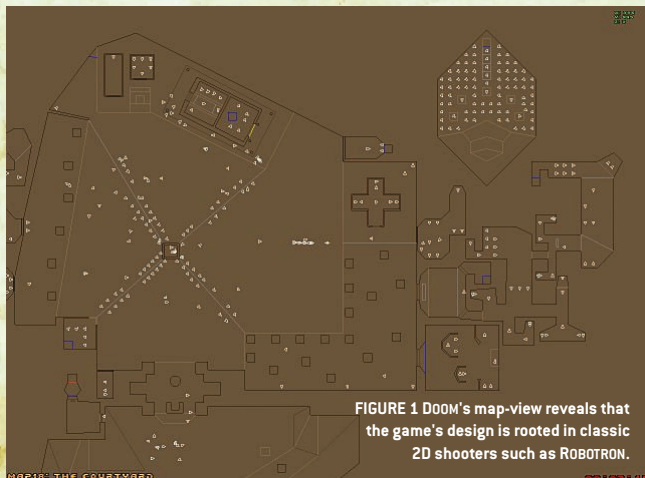


FIGURE 1 DOOM's map-view reveals that the game's design is rooted in classic 2D shooters such as ROBOTRON.



FIGURE 2 Players have the edge in speed and maneuverability over large swarms of enemies.



FIGURE 3 DOOM's environments are often abstract and have very little relation to real-world architecture.



abstract spaces DOOM has [see Figure 3]. I would love to see more modern games break with this conventional wisdom and see where it leads, if only in an indie or experimental context.

### DOOM ENABLED A REVOLUTION IN PLAYER-GENERATED CONTENT

Although advanced for its day, DOOM's technology was still simple enough and its content low-fidelity enough that a huge mod community coalesced around it to produce an unparalleled number of levels, mods, total conversions, and other add-ons. This, combined with the fact that the player base was so focused on a single game, means we'll probably never see something like it again. The lesson for future games might be this: make your technology extremely simple, easy to modify; ship it with a diverse enough pool of content that people can extend it to create a variety of settings and styles, and promote the sharing of this content as a way to add value to your game.


Many PC games have gotten all that right but failed to attract a huge community because of the content fidelity issue. The barriers to entry facing someone who wants to make a mod for UNREAL TOURNAMENT 3 today are vastly higher than those facing a DOOM modder. You can rough out a DOOM map in a few hours and finish it in a few days, while that same amount of time might produce a single texture for a modern game. Again, this is

something we could branch out from if we lost our fixation on technology and high-fidelity visuals.

Another unique side effect of DOOM's simplicity is that its design principles can be synthesized and expressed procedurally. Level generators for more modern games have been attempted and abandoned, while the Oblige random level generator actually creates a decent DOOM level with proper combat and resource balance, key gating, and architectural themes.

DOOM is one of many classics whose less-obvious qualities are seldom revisited.

DOOM's impact has faded, and its precise recipe for success is unlikely to be replicated. Nevertheless, the game industry has become quite adept at mimicking its superficial qualities. We, as creators, owe it to ourselves to look at DOOM and other classics of comparable depth—M.U.L.E., ULTIMA IV, and STAR CONTROL II are a few examples I would offer—to trace less-traveled paths of analysis in search of deeper truths.

Sometimes we must look to the past for guidance. Other times we must strive to forget it entirely. In the balance of both, we will find much to learn about making the games of tomorrow. 

**JP LEBRETON** works at 2K Marin and was lead level designer on BIOSHOCK 2. He thinks too much about games and enjoys making strange things.







# FLASHDEVELOP 3.0.6

REVIEW BY MICHAEL GREENHUT

**IT'S POSSIBLE THAT SOME FLASH** developers long for the days of punch cards. They may think finding a line or two of code in a haystack of ActionScript class files is as much fun as an Easter egg hunt. They might even relish opening one file after the other in Notepad and scanning every line. If you're one of those coders for whom the journey is more important than the destination, read no further. If, on the other hand, you've used swear words in trace statements to vent your frustration at not being able to find what you're looking for, the open source code editor FlashDevelop may be just the tool for you.

To begin with, here are some of the nitty-gritty tech specs of FlashDevelop: it supports a variety of formats and highlighting—AS2, AS3, XML, MXML, XHTML, CSS, and PHP. It allows for automatic JavaDoc creation from methods and it allows multibyte character encoding. To run it, you will need Flash Player 9 ActiveX runtime and, if you're using Flex, Java 1.6 runtime. FlashDevelop is a Windows .Net 2.0 application and is compatible with Mac OS/Linux using virtualization software. One gigabyte of RAM should be enough, although there's some suggestion

that it'll run on 512k without the Flex compiler SDK.

## KEEPING IT CLEAN

» When you look at a class file in FlashDevelop, you'll first notice the colors. Comments are green, variable and function declarations are blue, quoted material is red, and the "meat" of the code is in black. You'll then notice how well-spaced and sectioned off the code is, with

indents and brackets in all the right places. FlashDevelop spaces your code for you—all you have to do is put a function name here, a bracket there, and hit Enter. Finally, you'll notice my favorite part of this visual feast—the humble but visible dotted lines that connect opening brackets to closing brackets, eliminating the frustration of trying to keep track of your eleven nested for loops that span eight hundred lines. Even

a chronic slob like myself would have to be pretty talented to make FlashDevelop code look messy.

For the avid don't-do-it-yourselfer, FlashDevelop is a dream come true. If you're sick of typing "package this, public that" every time you write a new class file, it will do this for you. Just go to the File menu and tell it whether you're making an AS2 or AS3 class, and you'll see a brand new page open up with all the boring

having to go back and forth to the File menu every time you want to open a new file, click on any one of the file names in the Project window and it'll open in the main window.

Your code's .fla file can also be included in this bundle, so when you hit Ctrl-Enter from within FlashDevelop, the Flash IDE will do its thing. Sadly, there are a couple of drawbacks to this Project explorer: you can't seem to work on multiple

If you're a relatively new ActionScript programmer who has yet to try a real coding environment, the sooner you break the shackles of the standard Flash IDE's vanilla ActionScripting environment, the better.

headers in place. Once you start typing the real work, FlashDevelop's code completion features give a new meaning to creature comforts. If you don't like hammering out for loops, while loops, if statements and the like, the Code Snippets menu under the Tools menu will take care of this. Too bad FlashDevelop doesn't feature automated code refactoring. If only it could type your comments for you, too ...

projects at once, and depending on how rambunctious you've been with switching between CS3 and CS4, it will occasionally get confused about which version of Flash to open and execute your .fla with.

For those of you who need your whitespace, code collapsing is another neat feature of FlashDevelop. Those little plus/minus boxes next to the line numbers will fold up the guts of your functions or comments and represent them with thin black lines. Should you decide to do a search for something that happens to be collapsed, the item will automatically make itself visible, though you'll have to collapse the list again when you're done.

## FINDING YOUR WAY

» That "Project" window to the right of your code isn't as silent or deadly as it looks; it's one of your better allies in the war against chaos. If you're a professional ActionScript developer, chances are you'll be navigating through directories of code that are more intricate than the New York subway system. If your company uses FlashDevelop, that's no problem—you'll have a project file in the root directory you can open that will display a list of your relevant files in the aforementioned Project window, complete with a mock-up of the directory structure. Instead of

## NEEDLE IN A HAYSTACK

» Possibly the most useful feature of FlashDevelop is "Find And Replace In Files." Consider this scenario: It's your first day on the job, and you're told to change the physics of an existing pinball game for a very impatient client. The game has a few dozen class files with a thousand

### FlashDevelop 3.0.6

✂ **STATS** [www.flashdevelop.org](http://www.flashdevelop.org)

✂ **PRICE** Free (Open Source under MIT license)

✂ **SYSTEM REQUIREMENTS** Microsoft Windows. Flash Player 9 ActiveX runtime required by the browser control and Java 1.6 runtime required if using the Flex SDK.

#### ✂ **PROS**

- 1 Excellent code completion/generation AI.
- 2 The layout of the Project explorer.
- 3 The ability to find and replace text across multiple files.

#### ✂ **CONS**

- 1 Only being able to see one project at a time.
- 2 Having to refactor code without any extra help.
- 3 A less than perfect .fla testing mechanism.



## product news

### **GDEBUGGER VERSION 5.5 GRAPHIC REMEDY** [www.gremedy.com](http://www.gremedy.com)

Graphic Remedy has released version 5.5 of its gDEDebugger tool for Windows, Linux, Mac OS X and iPhone. gDEDebugger is a debugger, profiler, and memory analyzer for optimizing OpenGL, OpenGL ES, and OpenCL-based applications. This version introduces AMD GPU performance counter integration which displays AMD graphic hardware and driver performance counters in gDEDebugger's Performance Graph and Performance Dashboard views, allowing developers to optimize their application over AMD graphics hardware. AMD Performance counters are available on Windows when using the ATI Radeon HD 2000 series or newer with Catalyst 9.12 or newer.

### **RAKNET JOINS PS3 TOOLS AND MIDDLEWARE PROGRAM** **JENKINS SOFTWARE** [www.raknet.net](http://www.raknet.net)

Online middleware provider RakNet announced that its open source network game engine now offers support for the PlayStation Network

family of services, including online matchmaking, audio/video chat, and title user storage. RakNet is a C++ game networking engine designed for ease of use and performance. Features include object replication, remote procedure calls, patching, secure connections, voice chat, and real-time SQL logging.

### **GAMEBRYO LIGHTSPEED AND SIMPLYGON INTEGRATION** **DONYA LABS** [www.donyalabs.com](http://www.donyalabs.com)

Donya Labs announced the product integration between Emergent Game Technologies' game development engine Gamebryo LightSpeed and Donya's 3D optimization/reduction solution Simplygon.

Donya Labs promises that developers who license both Simplygon and Emergent's LightSpeed will be able to speed up production by automating the creation and setup of all level of detail (LOD) objects in a game, and also achieve more visually optimal LODs for both static and dynamic objects. Select features in Simplygon include polygon-mesh reduction to specified

pixel resolution or absolute distances; preservation of all critical features of the original 3D data for both static and animated objects; generation of decimation history for runtime purposes such as geomorphing; and unique remeshing and repair functions including generation of 2-manifold meshes, topology reduction, and removal of interior objects.

This new integration with LightSpeed allows automatic simplification of meshes for the creation of LODs and the addition of scenegraph nodes for switching between them, the batch optimization of 3D assets, optional settings for control of reduction quality, and AutoLOD methods.

### **XAITCONTROL 3.0** **XAITMENT** [www.xaitment.com](http://www.xaitment.com)

AI game tool provider xaitment has released xaitControl 3.0, a graphical modeling AI game tool that now includes an innovative graphical live debugger for creating and modeling hierarchical finite state machines. The tool allows users to set breakpoints

and make live changes to variables and states during runtime. Its graphical user interface gives designers and non-programmers alike the ability to visually create complex hierarchical probabilistic finite state machines. In addition to AI development, xaitControl can be used to implement game logic and control animation. Xaitment also offers a suite of AI tools called the BrainPack that provides developers with automatic navigation mesh generation, pathfinding, movement, behavior modeling, and advanced decision-making tools.

### **MASSIVE 4.0 RELEASED** **MASSIVE SOFTWARE** [www.massivesoftware.com](http://www.massivesoftware.com)

Massive Software, developer of 3D animation software for AI-driven characters, has released Massive 4.0. Massive incorporates procedural animation and AI to give artists the ability to create and direct CG characters such as humanoids, birds, and animals, as well as moving objects that deliver realistic and emotive virtual performances. Massive's "Agents" are 3D characters that have a fuzzy logic AI

brain and the natural senses of sight, sound, and touch enable them to interpret and react autonomously to the world around them.

V-Ray rendering engine support has been added for scenes requiring highly realistic lighting. Massive generates .vrscene files for the main render files, terrain, and agent archive files. It also automatically generates shader dialog parameters for V-Ray material plug-ins and V-Ray materials, giving users the freedom to assign any available default or custom plug-ins and materials.

Stadium placement has been made easier with the addition of rows and columns to the polygon generator, which allow for locators to be placed in rows and columns in any shape. The polygon generator also offers a stagger parameter which can be used to offset rows laterally.

Massive also features Brain Node improvements that allow for the control of delay time and filter width in the output node with other nodes in the brain, using new options to select between "latch," "delay," and "filter" as destinations for alternative inputs.

lines of code in each one. It's written by someone from a different time zone, and you have until tomorrow to figure it out. If only you could find the exact spot in the code where the ball makes a `hitTest`, you'd be off to a good start. With just the bare Flash IDE at your disposal, you're in for a long night and a few more white hairs.

With FlashDevelop, you need only hit Ctrl+I to bring up a menu that will let you search for specific bits of text across every single file in the game. Type "ball" or


"`hitTestObject`" (or whatever you think is relevant), and at the bottom of your screen you'll see a list of files and line numbers where this text appears with a snapshot of the text itself. Click on any one of them and you'll be whisked right to the point of interest. This feature has saved my life, or at least my job, more times than I can possibly keep track of.

### **ACTIONSCRIPT IN THE AGE OF FLASHDEVELOP**

» It's hard to remember life before

FlashDevelop, and you'll wonder how you ever survived without it. If you're an experienced coder, you might compare it to something like Eclipse, which I find a bit more confusing and less proficient at formatting code. If you're a relatively new ActionScript programmer who has yet to try a real coding environment, the sooner you break the shackles of the standard Flash IDE's vanilla ActionScripting environment, the better.

FlashDevelop is also free and open source, so you can save

your money for that CS4 bundle you've been wanting, and you don't need a whole lot of experience to use it (though experience getting frustrated with the Flash IDE does help). Mika Palmu, Philippe Elsass, and Nick Farina created FlashDevelop. You can find it at [www.flashdevelop.org](http://www.flashdevelop.org). 

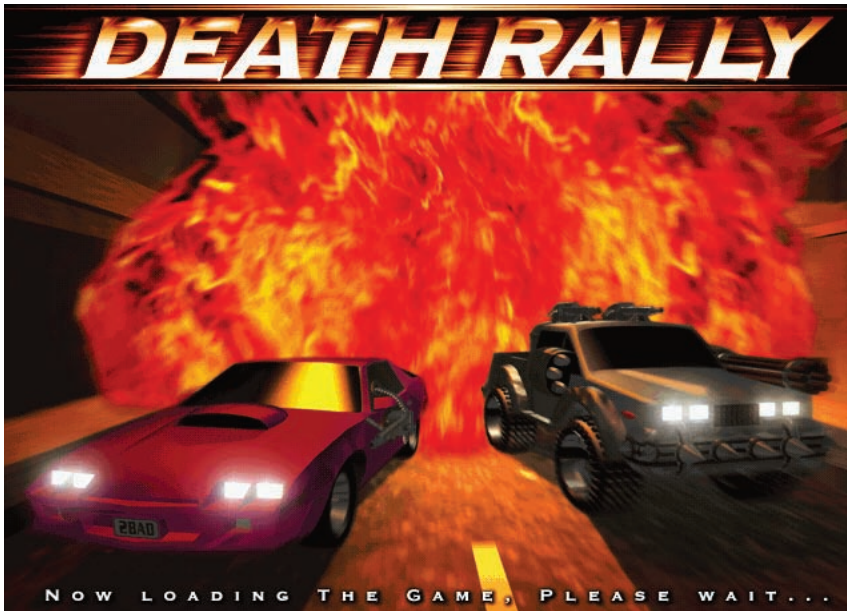
**MICHAEL GREENHUT** lives in Mamaroneck, NY, and is currently a Flash game developer at Arkadium in Manhattan. He likes to write science fiction and fantasy in his spare time.





# PORTING FROM DOS TO WINDOWS

STEP BY STEP THROUGH DEATH RALLY'S JOURNEY TO THE NEW MILLENNIUM



## MAX PAYNE/ALAN WAKE CREATOR REMEDY'S TOP DOWN COMBAT RACING GAME

DEATH RALLY was released for DOS computers in 1996, and although it does run under the open source DOSBox emulator, it doesn't run very well. I felt that DEATH RALLY was still a good game and wanted to get it into a playable form again.

So last May I got an idea, and thought, "What the heck, let's go for it." I sent an email to Remedy Entertainment, volunteering to make DEATH RALLY open source. I didn't expect a reply; at the most, I expected a polite "no." Much to my pleasant surprise, I got a "maybe."

After a couple weeks of legal checking, we agreed that while an open source release would not necessarily be possible, we could probably work something out. And so it came to be that in July, I downloaded the source package for evaluation.

The first task would be to take a cursory glance at the material and see if the project was actually possible. I expect some of you to wonder whether there was any funny code. Sure there was. Take a peek at any large project you've done as a teenager over a decade ago and see if there's any funny code in there. I couldn't find anything truly "daily wtf"-worthy, though, and what I did find wasn't anything a few days' worth of refactoring wouldn't fix.

Instead of refactoring, I took an archaeologist's approach—I made minimal changes and marked my transgressions clearly in the source code.

## STARTING BLOCKS

» The source software platform was DOS, Watcom C, and some Dos4GW-style DOS extender. The extender basically meant you could use more than

640k of memory, and would not need any weird code for data larger than 64k.

The game displayed in VESA 640x480 and MCGA 320x200 graphics modes, all with 8-bit palettes; there was no true color anywhere. There were also some per-frame palette change tricks that emulators have trouble with.

The source code was mostly pure C with a couple dozen inline assembly functions. There were a few missing subsystems, specifically audio and networking, which would have to be replaced completely anyway, as well as one file for which the source code was lost and only a compiled object was available.

## GETTING IT TO COMPILE

» First order of the day: get the game to compile. I started a Visual Studio project, imported all source files, and checked what the compiler would say.

The Visual Studio and Watcom compilers disagree on several points, which is hardly surprising as the Watcom version used was about a decade older than the Visual Studio I used.

One of the obvious things is that Watcom considers chars to be unsigned, while MSVC sees them as signed by default. There's a compile option in MSVC for this, but in order to avoid confusion further down the line, I opted to do some search-and-replace operations to designate all chars unsigned (except for those that were explicitly set to be otherwise).

MSVC is also much pickier about types, so I got lots and lots of warnings, and even errors in some cases. Most of these were relatively simple to fix—some typecasts here, a prototype added there, sprinkle some parentheses around. One rather tricky bit was where Watcom and MSVC disagreed slightly on requesting the address of an array, so I had to manually patch things up in a few hundred places.

After fixing a truckload of small errors and warnings, and stubbing all assembly functions as well as other missing symbols, I ended up with about 90 functions that needed rewriting.

## NO MORE HARDWARE ACCESS

» In DOS, there's not much of an operating system in your way. You could, and in many cases you must, access hardware features directly. For instance, graphical video memory was mapped to the real-mode segment 0xa000. This segment was usually (if not always) mapped to the direct address 0xa0000 in DOS extenders.

Higher-resolution VESA modes could be accessed most commonly in banks through the above segment. If you wanted to access more of the memory, you used some interface to switch memory banks, and then



ACCESS IT IN REAL TIME

# gamedeveloper



DIGITAL EDITION



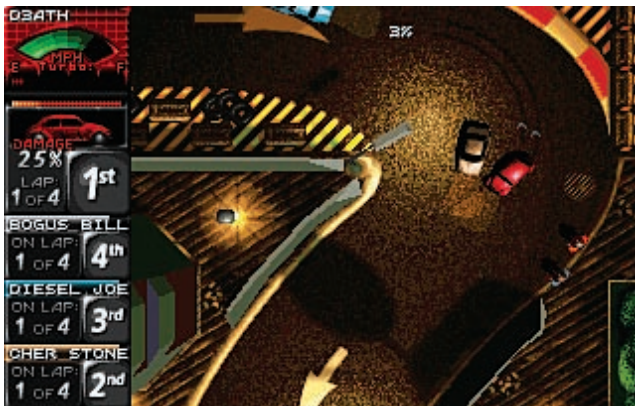
*Game Developer* magazine's 6 month and 1 year Digital Edition subscriptions give you new issues delivered promptly, full searchable access to more than 50 back issues, plus downloadable PDF versions, easy access from any web browser, and more!

Subscribe today!

[HTTP://WWW.GDMAG.COM/DIGITAL](http://www.gdmag.com/digital)







DEATH RALLY in its original DOS version (top) and ported Windows version (bottom).

accessed the same segment again. Thus, the applications set a graphics mode (and possibly segment) and accessed the video memory directly.

I solved this by allocating a frame buffer big enough for 640x480 and creating a global variable called `g0xA0000`, replacing all direct addresses with said pointer. The pointer would be updated to the beginning of the frame buffer on `mode_init` and to different offsets based on the bank switch calls.

Other video features were accessible through hardware I/O ports. The most important were the vertical retrace check and palette access. These I replaced with completely separate functions.

## DATA I/O

» That one object file with no source code happened to house decompression functions for the game data. I disassembled it and wound up with about five hundred lines of assembly, which, from a cursory glance, did not look like the output of a compiler. Not completely inspired to reverse engineer the code at this point, I took a shortcut to more interesting things by using the object file to make a DOS application (using OpenWatcom) which decompressed all the game's data files, and wrote a simple hack to access the decompressed files instead. This was clearly not a final solution, but it allowed me to progress.

There were some small problems with this approach. Audio files were in a differently encrypted format, and some of the game's small animations were handled differently in the decompressor with parts of the data compiled into the executable instead of the data file.

I made a note that while the cutscenes were also compressed, the source code for the decompressor was in C. So if both compression algorithms were written by the same person, the algorithms might also be similar.

A few days (and a dozen rewritten inline assembly functions) later, I

came to the realization that I had to get that decompression function to work. Strange bugs and crashes had started to crop up, most likely caused by bad or completely missing data not produced by my temporary hack.

Trying to find another easy way out, I compared the characteristics of the code with known compression algorithms, discarding most of them due to the requirement of overly large lookup tables or code complexity. The source code to Info-ZIP is invaluable for these kinds of things, as it implements most common compression algorithms, not only the ones found in modern ZIP formats. In the end, it was clear this was a proprietary algorithm, so I really did have to dive in.

I spent a couple days poring over the code and trying to re-implement what it does in C. Once I understood what the assembly code was doing, I took another glance at the section that decompresses the cutscenes and realized it's almost the same—except for some additional encryption. I made a variant of that code and the data problems went away.

With that bit done, I took a look at the audio files which had an additional layer of encryption. At this point, my Remedy contact, Markus Mäki, commented, "Who on Earth has been encrypting all these things and why?" Luckily, the source code to decrypt the audio files was found.

## APPLICATION FRAMEWORK

» The way applications work in DOS is somewhat different from what most people are used to these days. Control was entirely in the application's hands. There wasn't any multiprocessing to worry about, and you could pretty much depend on the characteristics of the de-facto VGA standard. If there were problems, users were expected to manually play around with system configuration text files.

Since the whole game was vertical retrace-synced (at VGA 70Hz), it made sense to place the OS message pump and graphics output into the vertical retrace check function. This worked beautifully, except for places where the game did not bother to wait for retrace (such as simply showing something on-screen and then waiting for a key in a busy loop). No retrace check, no message pump, no keys pressed. Adding the retrace checks to the loops naturally fixed the issue.

```
// Copy image to screen  
memcpy((char*)0xA0000, myImage, 64000);
```

```
//Wait for key press  
getch();
```

Copy data directly to video memory and busy wait for a key—perfectly legal in the DOS era.

The game also utilized a timer interrupt that ran in sync with the video refresh rate. I did not bother trying to make a separate thread to make it run exactly at 70Hz, and simply called the interrupt routine at approximately 70Hz in my message pump code. One positive side effect of this approach was that the per-frame palette-change tricks worked automatically.

I also wrote some placeholder keyboard handling code, which much to my surprise, worked directly. Apparently, the SDL scan codes match whatever DOS had, or came close enough.

## CONNECTING THE DOTS

» Instead of converting one inline assembly format to another, I rewrote all the functions in C. I think the result was actually not slower, as compiler optimization technology has improved a lot and the original assembly was written with original Pentiums (or worse) in mind.

Most of the assembly functions were little things, like rectangle copy or bit mask matching, and did not take too much effort to write. First the menus, then the in-game graphics started to come into view. This part was



pure joy—not so different from eating pistachio nuts: each bite takes a little effort, but has a huge payoff. I always wanted just one more, making it very difficult to call it a day.

One final piece of assembly was the polygon filler. In this area, I opted not to faithfully reproduce the original code, but wrote a software rasterizer from scratch, so if you see the polygon filler glitch, that's probably my fault.

It had been about three weeks and the game was playable. Still no sound and tons of small things to do, but playable.

## AUDIO

» Next up was audio. The game used Scream Tracker 3 S3M modules for music and FastTracker 2 XM modules for sound effects. Why both were not in the more advanced XM format, I do not know. Maybe XM for sound effects was a later addition, or maybe the composer preferred the S3M format. Music was relatively easy to handle, except for a small glitch where the replacement audio system was optimizing things a bit too much.

The game used a trick that was common in those days, where you place several looping songs into one module and instruct the replay routine to switch between the songs by jumping to a certain order number. The songs in question had some empty orders between songs, and these got optimized out, messing up the sub-song order numbers. Luckily, the order list was easy to read from the S3M directly, so I could make simple translation tables from optimized to original and back. The sound effects, however, took a lot more effort.

After several false starts, including writing a complete XM module loader, I took the open source XM player minifmod and made some severe modifications to it in order to use it as the sound effects library. The original sound library had a notion of “pitch” that wasn't in Hertz, but in something related to the notes. I had to invent an algorithm that approximated the conversion from the original “pitch” to a relative note. While the result is probably not exactly the same, it kind of feels right.

## GRAPHICS

» I started off with the idea that since the original game used two graphics modes, I might do the same. Unfortunately, the 320x200 mode did not work out, so I opted to only use one graphics mode—640x480 with a simple scaler for the 320x200 mode. The bad side of this decision was that the aspect ratio for the 320x200 mode was wrong. I could have spent a long time making some kind of weaving algorithm to turn 640x400 into 640x480, but opted to just add black bars at the top and bottom instead.

This got the graphics going quickly. In testing, however, we found that some non-4:3 aspect ratio LCD screens stretched the image, and for dual-screen systems, switching to full-screen 640x480 moved all the other windows in an irritating manner.

Near the end, I figured it would be best to use OpenGL to scale the frame buffer to screen at desktop resolution. This would solve several issues, including the 320x200 screen mode aspect ratio. In high enough target

resolution, the 320x200 looks pretty authentic when using point-sample texture lookup. I was originally wary of this approach because of possible performance concerns on low-end 3D hardware, but testing on some low-end Intel chipset mini-laptops cleared these issues.

Implementing the OpenGL blitter was easy, but it uncovered a nasty issue. While I was running in software, the display update was not synced to the display refresh rate. With OpenGL, it was. The game's internal clock was fixed to the 70Hz VGA refresh rate, which I was faking. The way I implemented the vertical retrace meant that whenever the application even asks about the retrace—not only when it was waiting for retrace—we'd do the message pump and the display would update (up to 70Hz, anyway).

The in-game graphics were requesting information about the retrace about 50 times per frame in the worst case. Different aspects of the game world were asking which frame we were on for animation timing purposes. While doing the message pump, we had no idea whether we would have a new screen to show or not. As a result, we spent a few milliseconds here, a few milliseconds there, and suddenly had to wait for display refresh, wasting a dozen milliseconds and so on, until the game was crawling at about 1Hz.

In order to solve this, I made two changes. First, whenever the code asked about the retrace, I'd increment the “current frame” value by one and return that, instead of jumping to the correct real-world time value I was doing originally. The exception was, if we had already caught up with the real world, in which case, the current value was returned. This solved the slowdown issue everywhere else except in-game.

I added a hack for this: a flag which disables the display update. I set this flag on for all other parts of the game loop except when actually waiting for vertical retrace.

## RALLY CROSSED

» And so the port was done. One major part which was unfortunately left out was the multiplayer networking, as it would have required a non-trivial rewrite. Apart from a few cosmetic changes, the game is the same as its original DOS counterpart: you now exit to the OS instead of DOS—I also added a few additional delay loops where loading times have become insignificant and other little touches like that.

## PITFALLS

» By now, you may have noticed that all the issues I faced with the porting have to do with technologies that have changed, and in most cases, improved with time.

Apart from the speed, memory protection, and compiler issues mentioned earlier, there's one more thing that has also changed with time: quality requirements.

Back then, PCs were not even as standard as they are now. There was no process memory protection, and in general, it was okay for programs to do all sorts of funny things. Sometimes they crashed, and this was considered acceptable within reason. After all, some PCs were more stable than others. Still, these bugs haven't gone away, and you may have to add in some additional bug-hunting time for your modern port.

Luckily for me, the DEATH RALLY codebase was pretty stable. Still, I spent some time hunting bugs that occurred rarely, and in some cases, never appeared on my development system. A few crash cases were due to my own misunderstanding of some of what was happening in the source; some were actual bugs in the original code, but they were all more or less simple to fix or work around.

DEATH RALLY was released as freeware for Windows and can be downloaded from [www.death-rally.com](http://www.death-rally.com). I hope you enjoy it as much as I have! ☺

JARI KOMPPA is a demoscene veteran, most known for his work on the X-Forge cross-platform mobile game engine from (now defunct) Fathammer. He's not working in the games industry at the moment, but daydreams about getting back one day.

project at a glance

**DEATH RALLY**

RANKING	
1. SAM SPEED	100
2. JANE HONDA	86
3. NASTY NICK	77
4. MOTOR MARY	69
5. MAD MAC	63
6. MATT BILLES	57
7. CLINT WEST	51
8. LEE WIGG	46
9. DARK RYDER	42
10. CRIC DEER	37
11. SUZY STOCK	33
12. IRON JOHN	28
13. MORI SATO	25
14. CHER STONE	20
15. BISSAL JOE	17
16. MIC PAIR	13
17. LIZ ARDEN	9
18. BOGUS BILL	5
19. FARMER TED	2
20. BATA	0

**MEDIUM RACE RESULTS**

1	IRON JOHN
2	SUZY STOCK
3	LIZ ARDEN
4	MIC PAIR

**ORIGINAL SPECS**  
DOS (w/xtender)  
Watcom C  
MCGA 320x200x8  
VESA 640x480x8  
60-plus MHz CPU  
8 MB RAM

**NEW SPECS**  
Win32  
MSVC7.1  
16 or 24-bit color  
in various resolutions  
1-plus GHz CPU(s)  
1-plus GB RAM

PRESS ANY KEY TO CONTINUE...





# GOING SOLO

## 'CAUSE FREELANCING ISN'T FREE

**ARTISTS ARE NOT ALWAYS THE MOST SOCIABLE CREATURES. A LOT OF US** become artists because we want to say something personal—we're happier taking the show-stopping solos than sawing away anonymously in the back of the orchestra. Setting your own rules and working your way appeals to lots of folks, but artists—with their individualistic approach to work and style—are particularly given to fantasizing that they'd do better as freelancers. Who wouldn't prefer a constant stream of new projects to spending four years noodling on the details of a single IP? Who wouldn't want to earn an elite reputation instead of laboring in anonymity? And who wouldn't want to work when, where, and how they like?

Whether you dream of being a high-priced, globe-trotting gun for hire or of founding the next Massive Black—or, for that matter, if you're just waiting for the job market to thaw and need to pay the bills—freelancing can be a bracing challenge with rewards that are way beyond a paycheck and a sliver of profit sharing every three years. Unfortunately, freelancing is not for everyone. Working solo demands a lot more than simply setting keyframes in your jammies. No matter how good your art chops are, you'll have a tough time as a freelancer if you don't develop the right mindset. So this month we're going to highlight some things you need to know before you break up the band and embark on a solo career.

### STUDIO TRACKS

» Lots of freelancers choose independence because they love the freedom to set their own hours and work from home. It's easy to think working from home is like Nirvana while you're sitting in rush hour traffic or microwaving another crunch-time burrito. It eliminates the frustration and expenses of commuting; it lets you run to the grocery store or pick up the kids from school as needed, and it helps focus your energy on making art instead of meetings, paperwork, and office politics.

Unfortunately, working from home is not for everyone. Even introverts who love to pull on the headphones and hunker down in their cubes can find they need the camaraderie of a team. Sure, solitary confinement might seem like a great thing when you've got a deadline coming up and your cube-mates won't shut up about last night's *Jersey Shore*—but even so, only your fellow artists really know where you're coming from. Roommates and family can't share your frustrations, give art critiques, or help you with technical advice. Whether it's social contact or professional support you need, working alone can be demoralizing and dehumanizing.

Successful freelancers don't work reclusively for both psychological and business reasons. You probably won't lug your quad-core workstation and Wacom tablet down to the local Starbucks every day, but regular lunches with friends and former colleagues can give you the human contact you need to stay sane. Spending time with other folks in games is also the most important way to hear about new business. IGDA meetings, Max/Maya user groups, and industry social events are great places to find business contacts and to enjoy the tribal pleasures of socializing. Freelancers can't survive as hermits—whether the isolation drives you crazy, or you just miss out on good jobs because you're moping at home in your underwear, solitude is a dangerous trap for a soloist.

### GARAGE BANDS

» Nobody likes to commute, but running your real life and your work life in the same physical space can exacerbate our perennial problems with work/life balance. When "the office" is right downstairs in the den, it's way too easy for deadline anxiety or artistic perfectionism to prey on your nerves at any hour of the day or night. Your work, your health, and your personal life will all suffer if you don't disengage the gears once in a while, but this takes a serious effort of will.

There's an emblematic example—a story about Rene Magritte, the surrealist painter. Every weekday, he would get up and dress for work in a suit and tie. He'd leave the house and head off to a local cafe where he took his morning breakfast and read the paper. Then he would walk around the block and report to his studio (a.k.a. his living room) by the back door. At 4:45 pm, he'd change back into his suit, leave by the back door again, and head back around the block to the front door, where he'd enter as punctually as any commuter coming home from the evening train.

You might not want to go that far (it's tough to buy a decent bowler hat these days), but if you can't build an effective firewall between your two lives, working at home can be brutal. If you're sneaking downstairs to tweak UV placements at 3 in the morning, or if you can't meet your deadline because you're trying to babysit and paint textures at the same time, you may need to look at moving your workspace out of the house.

Office co-ops can provide an affordable way to set up shop and they also provide some human contact during the working day. Some of these "rent-a-desk" setups come with serious broadband,



printers, fax machines, and even shared receptionists and meeting rooms (which can be useful when you're first forging a relationship with a new client). For many people, the structure and discipline of having a dedicated workspace is well worth the money.

## ONE-MAN BAND?

» Office space isn't the only expense you'll need to plan for. Freelancing forces you to balance costs and benefits all the time. Going solo makes you responsible for your own logistics, from maintaining the company network to paying for garbage collection. Running your own show involves a lot of small-scale tasks that are hard to enjoy, but impossible to ignore. Freelancers need to think hard about which tasks they take on themselves and which need to be outsourced. Your work time is your most valuable asset: deciding when and where to spend it is the key to success. Don't fall into the trap of seeing only the dollar costs of critical services.

For example, you'll need a webpage to advertise your services and show your portfolio. If you're a professional 3D artist, you're certainly smart enough to learn how to set up a site. But how many hours a week can you spare for learning new web software, applying security patches, and keeping your server up and running? Are you going to be sufficiently thorough about backups? Will your ISP bills go up if one of your animations gets downloaded a thousand times in a month? Each of these questions can be dealt with individually, but time spent on secondary tasks isn't being spent on your real job—building kickass content that keeps your customers coming back.

By all means, do save money by doing simple jobs yourself, but get professionals to help with anything that you're not sure you can do well. The rule of thumb should be that the more critical the mission, the more likely you should be to pay for it. Designing business cards, for example, is something most artists can handle easily. A legal work agreement, on the other hand, is too important to make up as you go along. Here's a quick rundown of some non-artistic but very important services you may need to shop for if you're planning on freelancing.

» **INTERNET SERVICES** Most of your business will come from personal contacts (that's why you need to eat out all the time!), but a good site is critical to making sure that you appear professional in the eyes of potential new clients. Don't point new customers to a personal site with family photos or blog-tastic rants; keep your freelancing site focused on what possible clients want to see: your work and your work history. You'll look like a flake if your site isn't always up and always in a presentable state. If you can't guarantee those things

yourself, you should be ready to pay a hosting outfit to manage the site for you.

» **IT** When freelancing is your "real job," you must be well insured against data loss with good backups and a reliable setup. You don't look like a pro if you're losing vital work to crashes, viruses, or simple disorganization. Frequent backups of your portfolio, work software, and current jobs are a pain in the butt, but they are essential for survival—you need top-notch protection in place before your kid downloads a virus-laden screensaver and nukes your machine! If you have any doubts about your abilities to administer your network, fight viruses, and handle backups, buy some peace of mind from an IT consultant or virtual networking service like Mozy, JungleDisk, or CrashPlan. This isn't an area where gambling makes sense.

» **SOURCE CONTROL** Backup is great for disaster prevention, but it's also handy to have your own source control setup. Knowing you can roll back to an earlier check in frees you up to experiment safely. It also prevents you from filling your hard drive with hundreds of confusingly named files. You'll forget the difference between `Fire_dragon_05_final.mb` and `Fire_dragon_final_final.mb` much faster than you can imagine. Don't get caught overwriting good work or sending the wrong file to the client. Instead, set up a Perforce or Subversion server for yourself. Subversion is freeware, and Perforce is free for one or two users. If you're worried about maintaining the server yourself, there are Subversion hosting services like Beanstalk (<http://beanstalkapp.com>) on the web.

» **TAX AND LEGAL** Being your own boss makes you liable for all sorts of paperwork and taxes you don't need to worry about as an employee. Be sure to do your homework on local business regulations and taxes. If you've never owned a business before, you'll be unpleasantly surprised at the amount of busywork involved. Because the rules vary so much from place to place, it's not a good idea to try to navigate them on your own—at least, not as a first-timer. Local business groups and governments usually have webpages and real-life seminars on how to get started in your area. The Graphic Artists Guild ([www.graphicartistsguild.org](http://www.graphicartistsguild.org)) doesn't deal directly with game art, but does offer good advice for creative freelancers in general. While you aren't legally required to incorporate to do business on your own, you should go over the pros and cons with a lawyer—the differences in your taxes and legal protections can be very significant.

On the plus side, running your own shop lets you claim tax deductions on a lot of your business expenses, including computers,



software, phone, and broadband service. However, the rules for deciding what is deductible are complex and the record keeping involved is significant. It's hard to over-estimate the value of a good accountant, particularly one who specializes in helping self-employed professionals; between finding tax advantages and keeping you out of trouble with Big Brother, a good CPA is the freelancer's best friend. One useful trick that simplifies end-of-year accounting is to get a separate credit card for your business—it's an easy way to keep business and personal spending strictly separate and well documented. But seriously, find a small business accountant!

## SWAN SONG?

» Whew. If you started this article thinking that "firing your boss" would be liberating, this long list of mundane concerns might sound like a downer. Don't be discouraged—these concerns are just the cost of doing business for yourself. They won't kill you—they'll make you stronger. Becoming a freelancer makes you responsible for a lot of things you don't learn in art school, or even as a veteran line artist. Living and working without the safety net of a 9–5 job is a great adventure: the work, the risks, the freedom, and the potential rewards are all amplified. Freelancing isn't for everybody, but paying attention to the foundations makes it a lot easier to survive and prosper. The sooner this stuff is out of the way, the sooner you can step into the spotlight and take your solo. 🎤

**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 a consultant helping game studios perfect their art tools and pipelines.





While death can come quickly in *DEMON'S SOULS*, its long-term consequences can be minimized by careful players, allowing them to fully explore the limits of the game world.

# THE TRUTH OF CONSEQUENCES

## ENCOURAGING PLAYERS TO GO FOR IT

**BILL BELICHICK IS REGARDED BY MANY FOOTBALL FANS AS A BRILLIANT** tactical coach, but in November of last year, he made a decision that is debated to this very day.

His Patriots were up by six against their hated rivals, the Colts, when his team faced fourth and two at their own 28 yard line with two minutes left. Most coaches in this situation would automatically punt. Going for the fourth down and failing would give the Colts' Peyton Manning, one of the top quarterbacks in the game, a short field of 30 yards to score a touchdown and win the game. Punting would make him travel at least 70. The Patriots went for it. They failed, and then lost the game.

After the game, Belichick was defensive. He argued that going for it had high odds of success, and getting the first down would have effectively won the game. On the flip side, the Patriots' defensive line was exhausted, and Manning was cutting through it like butter—in that particular situation, the difference between 30 yards and 70 was relatively insubstantial. He argued that the upside was infinite and the consequences of failure weren't all that different from punting. If he'd succeeded, people would have called him a genius.

### BALANCING FAILURE AND REWARDS

» The relative chance of failure compared to the upside of success is what made Belichick's dilemma the fodder of debate for months after the game. Key decisions in the best games provide well-balanced decisions like this to the player. If the balance is askew, the experience can be weakened. Consider, for example, the myriad Facebook poker games where players have a near-infinite supply of fake cash. Since players face no real risk for their bets, they constantly make monster bets with abandon, making it almost impossible for purists to bet and bluff in a realistic way.

As designers, we create the carrots and the sticks that drive players

through our simulation. Most designers typically think first about the carrots—the rewards and bennies that encourage players to pursue "good behavior," but equally important are the consequences of failure and tough decisions. It's the consequences that give these decisions real weight; they provide the emotional lift for the greatest rewards.

Sometimes, these choices are fairly black and white. In *BIOSHOCK*, the player is given the choice to harvest or save the "Little Sisters" (young girls who provide the psychic energy that unlocks the player's latent power). Harvesting the sisters feels horrific but grants the player more power than saving them. The ratio of sisters saved to harvested determines what game ending you will see. Saving the sisters makes for a harder game experience in the short term, but the player receives gifts that nearly equalize the difference, and provide a much more rewarding ending. In short, the path of evil is the path to quick power, but the path of good has greater long-term gains.

### THE BEHAVIOR YOU INCENTIVIZE

» Economists are fond of saying that you get the behavior you incentivize. One commonly cited fact from real life is that mandatory seat belt laws have resulted in an increase in pedestrian deaths. One consequence of wearing a seat belt is that the driver himself is safer, which allows him to drive faster and more recklessly.

As designers, we must be careful of the behavior we incentivize—it is dangerously easy to penalize good behavior, or reward activities that actively destroy the player's own game experience. If you make a game where jumping is faster than running, players will jump everywhere they go, no matter how silly it looks.

Sometimes, the consequences of a design decision are more insidious than they seem at first glance. It may make sense for quest decisions to



affect the player, but consider that if you immediately slap an alignment penalty on a player as soon as he dares talk to the roguish thief hiding from the city guard, you're punishing him for trying to play your story, and teaching him that some of your content is best bypassed if he wants to take a lawful path. Writers need to be careful of how and where they structure these key choices, and how they ensure that completionist players have an avenue to experience the whole game without trashing their characters.

Facebook games like Zynga's *MAFIA WARS* and *VAMPIRE WARS* also create some incidental bizarre behavior. Success in these games often depends on the player having a large number of friends also playing these games—frustrating for players with few friends, and unnerving for those who are uncertain they want to badger their friends into joining them in their little vampire gaming fetish.

The developers saw spamming friends as an advertising opportunity, but these players saw it as a negative consequence. As a result, many devotees of these games set up alternate Facebook profiles, joining groups of like-minded “fake” friends in order to game the system without polluting their real friends list. Whether Facebook or Zynga sees this as something that negatively impacts them remains to be seen.

## BEING TOO HARSH

» Over the years, most games have become less punitive to failure. Massively multiplayer games are no exception. Back in the days of text MUDs, death frequently meant the loss of a level's worth of experience. By comparison, *EVERQUEST*'s penalty was only one-tenth of that—and of course, it seemed ridiculously benign. Ten years later, penalties have been reduced to the point where death is little more than a minor inconvenience.

Some hardcore players long for the old days, and some armchair designers even push the idea of permanent death as a way to create more tense and dangerous online worlds. What these designers lose track of is how these penalties affect the risks that players are willing to take. Most designers want players that take big risks, try odd and unusual things, players that will test the limits of the simulation to discover emergent gameplay. It's here that the experiential magic of interactive fiction really shines.

Some players crave harsh penalties, of course. They like to play *DIABLO 2* on Hardcore mode or *DEMON'S SOULS*, where the tension of ultimate failure provides an emotional edge, and minimizing the risk of failure is a key strategic decision in the game. Make no mistake, though, this is a hardcore game direction.

However, if the penalties are too harsh players won't take chances. They'll seek out lesser challenges. In MMOs, they'll hunt below their level, avoid grouping with strangers and not show up for player vs. player situations in which they are clearly the underdog. In short, they'll bore themselves to death, and then blame you, the designer. Correctly, I might add.

## HIGH RISKS, HIGH REWARDS

» There exists in baseball a huge subculture of fans that specialize in the statistical examination of the sport. Devotees of the science of sabermetrics attempt to upend any number of common conceptions of baseball. One of these misconceptions is that attempting to steal a base is ever a good idea. Statistically, the consequences of failing to steal a base (losing one of your precious 27 outs) almost always outweighs the potential benefit.

Despite this, managers still try to steal bases. The interesting thing is that, despite the fact that it's usually a terrible idea mathematically, it genuinely makes for a more entertaining baseball game. The apparent risk vs. reward does not match the actual risk vs. reward (which, fortunately, results in managers making decisions that make for better television).

As designers, it's important that the risks a player takes have rewards that correspond to their penalty—or at least feel like it. It's okay to have high-

risk choices for the player, but the rewards have to provide an emotional high that matches.

These high-risk, high-reward choices are excellent ways to provide an additional layer of gameplay for more advanced players. Killing 10 players with your bare hands is, for many players, worth the achievement. Attempting to throw an opponent out of the ring in *SOUL CALIBUR* is hard to pull off, but completely worth your opponent's humiliation if you execute it. Rocket jumping in *QUAKE* to the perfect sniping place is worth occasionally blowing yourself up or shooting yourself into the lava.

The important thing about these high-risk, high-reward choices is in fact the element of choice—they're alternatives. Making these optional gaming avenues leaves a safer, more predictable path for more casual, less-skilled players. Perhaps just as importantly, though, they provide a way for your more dedicated gamers to demonstrate and declare mastery over the game.

## UNCLEAR CONSEQUENCES

» Any first-year psychology student will tell you that the best way for rewards and punishment to work as a modifier of behavior is to ensure that the consequences are swift and directly related to an action. Whacking your puppy on the nose with a newspaper won't work unless you catch her in the act of piddling on the rug. While game players are a bit more self-aware than the average pup, designers can make their games much more powerful by making the consequences of failure explicit.

In the early days of MMO development, designers theorized about a virtual ecology. In this model, the virtual dragons would feast upon electronic sheep. If no sheep existed in their hunting area, the dragon would have no choice but to seek food in the village where players live. The moral of the story is to not destroy your own ecosystem.

The problem is the average player's inability to see the whole system. He doesn't know that killing sheep makes the dragon relocate—he may not even know that there were ever sheep around at all. All he knows is that he was one-shot by a dragon while trying to sell his gear to a vendor in newbie village. The idea that another person he's never met could kill him by making mutton chops half a mile away doesn't feel particularly fair.

This is a difficult problem, but not an insolvable one. MMOs with complicated realm vs. realm combat such as *EVE ONLINE* and *WARHAMMER: AGE OF RECKONING* use political maps to show players' land control, giving them a global view of the actions of the community. *WORLD WAR II ONLINE* goes further—that game's website allows players to see a history of how the front of the war has shifted over time.

## BIG, BOLD CHOICES

» Interactive games are at their best when players learn by doing. For this to work, though, players need a clear cause-and-effect for their actions. Subtlety here is often wasted. Give players a choice between two relatively minor stat penalties, and he will be left wondering if it was the right choice—or whether the choice made any difference at all. In games like *DRAGON AGE: ORIGINS* and *MASS EFFECT*, however, the big choices are black and white. He doesn't wonder if he made the right or wrong choice—he knows, as the consequences of his actions have big, bold effects on the game world.

Too often, though, subtlety is lost within the noise of the simulation. Some may decry this lack of subtlety of consequences, but I think that most players approach games coming from a real world where their choices often have no visible impact, or where the risks are simply too high for them to follow their hearts. Providing an avenue for experimentation and release is one of the things games do exceptionally well. Making the games that best provide this for the player requires a well-designed carrot—and an equally well-designed stick. 🍌

---

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





# SUBVERSIVE AUDIO DESIGN

## USING AUDIO TO INFLUENCE GAME DESIGNERS' CHOICES

**YOU FIND YOURSELF WITH A** game in front of you, tasked with its sound. Suddenly you realize—the game design could be much better!

Mechanics or art, character designs or level layouts, you can see just what the game needs! Immediately, you doff your “audio guy” hat, don a super hero cape and run off from your audio workstation to alert the designers about how they’re doing their jobs wrong!

Or maybe you don’t need to be so dramatic. If you’re a company guy doing in-house audio design, you might already be part of a process that allows you to have input into the game design, and it’s just a matter of exercising that ability. As an artist, you have the ability to influence people with your work in some very subtle ways.

For some, it’s as easy as walking across the office to the lead designer to give them your piece on the latest build. Early in my career, I found that doing that early and often in the game process was at once a great way of fostering the collaborative spirit within the team and a fine individual morale booster.

### THE OVERT APPROACH

» Nowadays, wikis are the preferred method for keeping a central design document accessible to the whole team. Sure, you might still be trying to avoid writing an audio function spec, but it might be worth the time to contribute to the game design wiki with some side notes on the

game components you’ve observed. Or perhaps you just want to quietly assert your opinion on the current state of the game.

Regardless, you do need to strike a balance in terms of how you present yourself. It is far too easy to be received as berating the design lead for his decisions, or passive aggressive, leaving notes in a rarely read wiki for someone to just wander across.

Sometimes a casual remark on an individual game system is enough to get the designers’ brains churning. Similarly, letting someone know you’ve made an update on the wiki can actually draw their interest. (Or maybe they’ll just be surprised that someone is still using the design wiki!)

Sometimes though, saying your piece on the game design isn’t enough to bring around the powers that be to your way of thinking. Or perhaps your working situation isn’t quite as I’ve described above. As audio guys, we can convince them by working our preferred medium.

### THE SUBVERSIVE APPROACH

» For a long time, artists have enjoyed some sway over game design. At this very moment, artists throughout the world are telling their leads, “Look at this awesome weapon I designed!” and subsequently changing the direction of the game. (See GOD OF WAR’s chain sword.) I say audio people have the potential for their work to

exert similar influence.

Let’s say you have a platforming character action that needs a sound. Attack, jump, collect, defend, or the like. If you design an action sound that is cool, and iconic enough, you might find designers scrambling to change their level layouts to feature that action and sound more often.

What if you think this theoretical game is going on the wrong path? Let’s say that right now, this platforming action game makes it easy for players to brute force their way through the stages, mindlessly attacking individual drone after individual drone until there are no more to destroy. But perhaps your gaming background is a bit different, and you’ve an affinity for chaining systems and combos, and you’d like to have the game use some of the techniques found in twitch action games like BANGAI-O.

No, you probably don’t want to suddenly turn this platforming action game into a scrolling shooter, but you do want the player to move in an elegant way, dancing around the enemies, letting them flirt with successfully hitting you before you let loose a single well-placed attack that vanquishes them all.

If you simply say this to a designer, he might look at you incredulously before dismissing the notion as too difficult to deal with. Ideally, you would have a prototype that shows the designer that this could be



BANGAI-O’s combo-heavy mechanics might play well in a platforming game—but how do you convince designers when it’s so visually intimidating?

an interesting battle strategy worth encouraging through the game design. If you can’t do that, though, working that perspective on the game through the audio design can be just as effective. In this scenario, getting the designer to realize that strategy could be as simple as:

» **Tweaking the attack sound to be more negative or less overtly positive sounding (perhaps making it sound more similar to enemy noises than the player’s),**

» **Tweaking the movement sounds (running, jumping, etc.) to be more obviously interesting (perhaps with more musical content, or giving it a subtly shinier mix),**

» **Creating a highly interactive sound for the collection of the enemies’ reward drops so that multiple collection sounds simultaneously are geometrically or even exponentially more rewarding than individual sounds.**

After updating the build with the new sounds, you might find that people in the office are playing the game in a different way. As gamers, we naturally allow all the feedback in the game to tell us how it is we should play the game. Here, you’ve designed the sounds in an attempt to have them dictate to the developers how the game should be played. It can be a subtle effect, and it might take a while for the change to take place, but good developers will eventually listen to what the game is telling them. If you notice a change in their playstyle, this is when you can propose your new idea.

You are the conduit through which the game “speaks.” You can let your voice be heard. 🗣️

VINCENT DIAMANTE is a music composer and educator in the Los Angeles area. He penned the score for ThatGameCompany’s FLOWER and currently teaches at the Interactive Media Division at USC. Email him at [diamante@gmail.com](mailto:diamante@gmail.com).





# BILZARD

ENTERTAINMENT



## 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](http://jobs.blizzard.com)





# From Space to Face

## RICHARD GARRIOTT JOINS THE SOCIAL NETWORK MARKET



**ULTIMA creator Garriott discusses his new social games venture Portalarium now that he's left NCsoft and returned from his space travels.**

**BRANDON SHEFFIELD:** *What made you decide to target the Facebook market?*

**RICHARD GARRIOTT:** The target is not specifically Facebook; the target is really the explosively growing market, what I'll broadly call the casual and social network players. I look at it and go, I feel like I've lived through three major shifts in the game industry. Number one is, of course, the beginning of the game industry! I was lucky enough to be one of the first developers of games, so with that

came great opportunity and revolutions.

The second one was the emergence of online gaming. I would argue that ULTIMA ONLINE was a major stepping stone in convincing people that online gaming was relevant. At the time, I was trying to get it going, no one supported it. It was very hard to get going, but when it finally shipped, it wound up making ten times the revenue of all the previous ULTIMAS combined. That has now, for the last decade, been by far the biggest growth area for gaming overall. I mean, you look at things like WORLD OF WARCRAFT, and it's now 10-to-100 times bigger than ULTIMA ONLINE.

If you look at this new casual and/or social media area of

gaming, a lot of people still either poo-poo it or don't understand it, yet the number of players on these games is dramatically in excess of even things like WORLD OF WARCRAFT. The amount of money flow on this side of the fence is already dramatically in excess of almost anything any game developer has ever developed and still people in this industry have the same mentality they did with online gaming before all the models were proven—"Oh, the quality levels aren't there yet" or "the types of games aren't interesting to me yet." And I say yeah, it's true that it may not be now, but I assure you it will be, and very soon. It's one of these coming juggernauts that you need to learn to understand and participate in the evolution of, or get left behind.

**BS:** *I think there's a lot of reluctance based on the fact that a lot of developers make the kinds of games they want to play, and your FARMVILLES and suchlike are not going to fill that need.*

**RG:** I agree! I don't play FARMVILLE personally. But I can tell you that as an avid gamer, almost 100 percent of my gaming has been done on the iPhone. I've tried every portable platform that's existed and none of them were particularly good. Finally,

we have a platform where lighter mobile games are compelling, even to a hardcore gamer. What's important to the users is not the games themselves, it's their friends. So the first thing you have to realize is their friendship, their networks and their friends, that is the dominant activity. So to find them, you have to go into that community. Then, when you decide that you want to present them with something that's entertaining, you have to be able to let them share that content, maybe send you a link. You have to be able to try it out for yourself with no cost, never having to go to a retail store, never sitting with a long download, never with an instruction manual or a tutorial—you have to be able to just sit down and play it.

In my mind, every online game we've ever made to date would be a better game if I could pick it up for free, if I could download it immediately, if I could launch it without any installs, et cetera. So the things the new market is demanding are actually a benefit to every game we've already made in history. And as you make these games, a lot of them will come along for the ride. A lot of them will become immersed in gaming and want to play more. Either way, it's going to be very market expanding, and the quality is going to go up very quickly.

## new studios

Tecmo veteran Tomonobu Itagaki of DEAD OR ALIVE and NINJA GAIDEN fame has formed a new company, Valhalla Game Studios. According to *Famitsu Magazine*, Itagaki's new house has already reached 50 staff members.

Former Volition technical designer Luke Schneider has started a one-man indie studio in Radiangames, where he plans to release [roughly] one game on the XBL Indie Games platform per month, beginning with twinstick shooter RADIANGAMES JOYJOY.

Former Havok founders Steven Collins and Hugh Reynolds have formed Kore, which develops Lua-compatible virtual machines for developers.

Wii-focused studio Judobaby has just been announced, with a number of industry vets at the helm, including president and CEO Dan Mueller, formerly of Bottlerocket, art director Ben Harrison, previously at Crystal Dynamics, and design lead David Ralston, who has games like PAPERBOY and RAMPART to his credit.

CCP Games, developer of the spacefaring MMORPG EVE ONLINE, opened a new UK studio, which is at work on "current and future" console projects.

Korean MMO company Bluehole Studio has formed a new Seattle subsidiary called En Masse Entertainment that will focus on localizing and marketing online games to Western audiences.

Senior Rockstar Leeds developers Lee Hutchinson and Matt Shepcar have left the company to start their own independent studio, Double 11, to focus on smaller projects, like iPhone games.



FROM THE CREATORS OF "SCRIBBLENAUTS", "DRAWN TO LIFE" & "LOCK'S QUEST"



WORK ON SOMETHING

# TRULY ORIGINAL

Now hiring for Xbox 360:

Lead Environment Artist

Senior Environment Artists

Lead 3D Level Designer

Senior 3D Level Designers

Lead Animator



**5THCELL™**

Visit [5thcell.com/jobs](http://5thcell.com/jobs)



# SCORE YOUR DREAM JOB AT GAMELOFT.

AT GAMELOFT, WE ALWAYS  
KEEP OUR EYES OPEN FOR  
FRESH TALENT.

WE EMPLOY CREATIVE  
AND AMBITIOUS:

- \* 2D & 3D ARTISTS
- \* GAME DESIGNERS
- \* PROGRAMMERS
- \* PRODUCERS
- \* AND MORE!



**[GARGANTIAN GAMERS, DARING DAY DREAMERS, HILARIOUS HIPSTERS,  
SUPER-DUPER SMARTY PANTS, WONDROUS WHIZ KIDS, BRILLIANT  
BRAINIACS AND CLOSET SUPER HEROES ALL NATURALLY, WELCOME]**

For more info about specific job opportunities in Barcelona, Buenos Aires, Montreal, New York City and Shanghai (Redsteam), please visit the career section at [www.gameloft.com](http://www.gameloft.com)

We will contact you if we believe your experience is a good fit for an open position, but if you don't hear from us right away, don't fret. You might be our next hire!

**gameloft**  
[www.gameloft.com](http://www.gameloft.com)





**Join the Team!\***

## —HAPPINESS—

- Be part of a Passionate Team
- Career Not a Job • Make Your Path
- Achieve Your Goals
- Free of Earthquakes and Wildfires!

## —CREATIVITY—

- Independent and Unbound
- Dream Projects & Original Games
- Collaborate with Top Talent
- Make Meaningful Contributions

## —MONEY—

- We Ship Bestsellers & Award Winners
- Over \$14 Million in Profits Paid Directly To Our Talent
- Exceptionally Low Cost of Living
- Get Paid to be Passionate about Work

\*The Awesome are welcome here.

### **Gearbox Original Games**

Brothers in Arms • Borderlands

### **Games from Gearbox<sup>†</sup>**

Halo • Half-Life • Aliens: Colonial Marines  
Tony Hawk • Samba de Amigo  
• 007: Nightfire

**Visit [gearboxsoftware.com/jobs](http://gearboxsoftware.com/jobs)**

<sup>†</sup>All games and franchises listed are the property of their respective owners and may be trademarked in the United States and other countries.

All inquiries regarding such trademarks should be made to their respective owner. All inquiries regarding Skagzilla specimens and Rakk Hive husbandry should be directed to the appropriate zoological authorities.



# Galactic Arms Race

## EVOLVING THE SHOOTER

### THE STUDENT-BUILT GAME

GALACTIC ARMS RACE is the intriguing result of ongoing work from the Evolutionary Complexity Research Group (EPLex) at the University of Central Florida. Starting with the basic idea of an online multiplayer space shooter, GALACTIC ARMS RACE adds a unique wrinkle to the competitive formula by featuring particle-based weapons that evolve into novel configurations during play.

We contacted Kenneth Stanley, the team's faculty advisor, to find out more about the automatic content generation driving GALACTIC ARMS RACE.

**Jeffrey Fleming: Can you tell me a bit about the team that worked on GALACTIC ARMS RACE [GAR]?**

**Kenneth Stanley:** The team reflects the origin of the game inside a research group at the University of Central Florida. I supervised the project as its faculty supervisor and my Ph.D. student Erin Hastings took the lead in software development and technology integration. The project required integrating novel AI technology developed for the project

into the game. The rest of the team was rounded out by volunteers who were mostly undergraduate students interested in gaining experience working on a game. Overall, the project represents a major volunteer and educational effort driven by people's passions, with little financial support.

**JF: What tools did the team use to create GAR?**

**KS:** GAR is made in XNA. It also uses NEAT and something called "NEAT Particles," which is a technology developed before GAR to allow NEAT to evolve particle systems.

**JF: What is the idea behind the NEAT algorithm?**

**KS:** NEAT stands for NeuroEvolution of Augmenting Topologies. I invented NEAT at the University of Texas at Austin when I was a Ph.D. student working with my advisor Risto Miikkulainen. As its name implies, it evolves artificial neural networks, which are kind of like little artificial brains. The innovative aspect of NEAT is that the brains it evolves actually get bigger

as evolution progresses, which is what the word "augmenting" means in its name. In simple terms, the implication is that behaviors can become smarter and more complex over time.

NEAT is the core of the algorithm that evolves the weapons in GAR. Actually, for GAR we introduced a variant of NEAT called cgNEAT, which stands for "content-generating NEAT." Believe it or not, a neural network evolved by cgNEAT drives every particle in every weapon in GAR. So the neural networks are actually controlling the way weapons behave. Because the weapons are evolving through NEAT, their behavior can become more complex and intricate over time.

**JF: How does cgNEAT decide which weapon to evolve in GAR and which are dead-ends?**

**KS:** The way cgNEAT works in GAR is that it tracks which weapons people like by observing which ones are fired the most. Those that are popular become the "parents" of new weapons that are spawned in the galaxy. Thus the question of which weapons evolve is answered by which weapons people like. If people like them, cgNEAT makes new variations of them and spawns them in the galaxy.

**JF: Are the evolved weapons specific to a single instance of the online game or are they part of a persistent world?**

**KS:** In multiplayer mode, the evolved weapons are



stored on the server, so they generally persist as long as the server. In that sense, they are part of a persistent world for each server. So the interesting situation is created in which evolution can continue over months or years.

**JF: Is the neural network very processor intensive?**

**KS:** In GAR, every single particle from every evolved weapon is controlled by a neural network and even when there are ten people on the screen at the same time all firing different weapons at once over a network, GAR players will not experience any slowdown. So from that perspective, today's CPUs are more than capable of handling many simultaneous neural networks being activated at the same time. Neural networks tend to be compact and require only a few floating-point operations, so they can often be less computationally expensive than more traditional control schemes. However, of course, if neural networks are allowed to grow very large, they can start to be more

expensive. Yet that size is well beyond what is needed for the type of control in GAR or many other games.

**JF: How difficult was it to integrate online play into the game?**

**KS:** Integrating online play was a challenge because we had to get the system to perform evolution over the Internet, which means that genomes and fitness information literally have to be sent back and forth through messages over the network. There is not much precedent for a real-time Internet-based evolutionary system like that. For example, if a player flies into your view with a weapon you've never seen before, the neural network for that weapon must be transmitted to your computer right away so that the other player's weapon looks the same to you as it does to the other player. However, once the proper information is set up to transmit to the right places, the overall evolutionary algorithm works seamlessly and is not hard to manage.

## resources

**GALACTIC ARMS RACE**  
<http://gar.eecs.ucf.edu>

**Evolutionary Complexity Research Group at UCF**

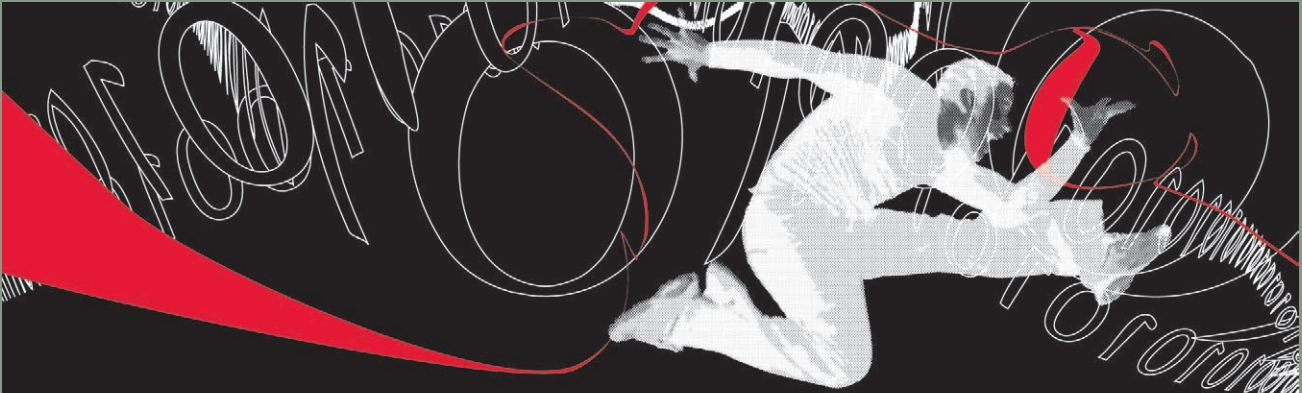
[http://eplex.cs.ucf.edu/publications/2009/hastings\\_ieeetciaig09.html](http://eplex.cs.ucf.edu/publications/2009/hastings_ieeetciaig09.html)

**Google Groups: Procedural Content Generation**  
<http://groups.google.com/group/proceduralcontent>

**Search-Based Procedural Content Generation**  
<http://julian.togelius.com/Togelius2010Searchbased.pdf>







# BECOME A MASTER OF DIGITAL MEDIA

We offer a 20-month master's degree in entertainment technology and digital media. This powerhouse graduate program combines industry-facing curriculum, real world projects, and a 4-month internship in Vancouver, Canada: videogame capital of the world.

Learn more. Contact [alison\\_robb@gnwc.ca](mailto:alison_robb@gnwc.ca) or visit [mdm.gnwc.ca](http://mdm.gnwc.ca)

**CENTRE FOR  
DIGITAL MEDIA**

Masters of Digital Media Program  
@ Great Northern Way Campus



**emily carr**  
university of art + design



*a collaborative university campus environment*

## 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



© 2010 Full Sail, Inc.

### 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
  - Entertainment Business: *with a Sports Management Elective Track*
  - Internet Marketing
  - Media Design
- Bachelor's**
  - Computer Animation
  - Entertainment Business
  - ▶ Game Art
  - ▶ Game Design
  - Graphic Design
  - Internet Marketing
  - Music Business
  - Music Production
  - Web Design & Development



**FULL SAIL  
UNIVERSITY.**

[fullsail.edu](http://fullsail.edu)

Winter Park, FL

800.226.7625 • 3300 University Boulevard

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



# 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 *Prototype*, *Mass Effect 2*, and *Dawn of War II*. The LA Times named VFS a top school "most favored by video game industry recruiters."



VFS student work by Julianna Kolakis

**VFS** Find out more.  
[vfs.com/enemies](http://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



THINK SERVICES  
GAME GROUP

MAY 6-7, 2010

**GAME DEVELOPERS  
CONFERENCE™  
CANADA**

Vancouver Convention Centre,  
Vancouver, BC

[www.gdc-canada.com](http://www.gdc-canada.com)



OCTOBER 5-8, 2010

**GAME DEVELOPERS  
CONFERENCE®  
ONLINE**

Austin Convention Center,  
Austin, TX

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



DECEMBER 5-7, 2010

**GAME DEVELOPERS  
CONFERENCE™  
CHINA**

Shanghai International  
Convention Center,  
Shanghai, China

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



AUGUST 16-18, 2010

**GAME DEVELOPERS  
CONFERENCE™  
EUROPE**

Cologne Congress Center East,  
Cologne, Germany

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



FEBRUARY 28-MARCH 4, 2011

**GAME DEVELOPERS  
CONFERENCE® 2011**

Moscone Center,  
San Francisco, CA

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



For updates and more information on our events visit [www.tsgamegroup.com](http://www.tsgamegroup.com)

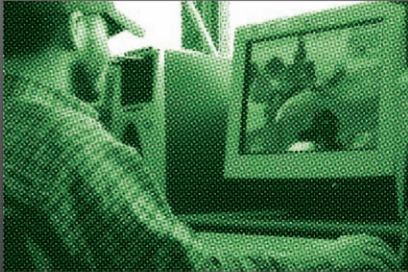






# GAMASUTRA

The Art & Business of Making Games



The Leading Professional Game Source for Job Seekers and Employers

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

Take Control of Your Future Today!  
Log onto [www.gamasutra.com/jobs](http://www.gamasutra.com/jobs)



GET EDUCATED

[ GEEKED AT BIRTH ]

06

You can talk the talk.  
Can you walk the walk?  
Here's a chance to prove it.  
Please geek responsibly.

[ IT'S IN YOUR GENETICS ]

GAME ART & ANIMATION  
GAME DESIGN  
GAME PRODUCTION AND MANAGEMENT  
GAME PROGRAMMING  
SERIOUS GAME & SIMULATION

ADVANCING COMPUTER SCIENCE > ARTIFICIAL LIFE >  
PROGRAMMING > DIGITAL MEDIA > DIGITAL VIDEO >  
ENTERPRISE SOFTWARE DEVELOPMENT > NETWORK  
ENGINEERING > NETWORK SECURITY > OPEN SOURCE  
TECHNOLOGIES > ROBOTICS & EMBEDDED SYSTEMS >  
TECHNOLOGY FORENSICS > VIRTUAL MODELING &  
DESIGN > WEB & SOCIAL MEDIA TECHNOLOGIES

[www.uat.edu](http://www.uat.edu) > 877.UAT.GEEK

DNA Genetics (DC)

## ADVERTISER INDEX

COMPANY NAME	PAGE
5th Cell .....	49
Blizzard Entertainment .....	47
E3 Expo .....	6
Epic Games .....	27
Full Sail Real World Education .....	53
Gameloft .....	50
Gearbox Software .....	51
Havok .....	C3
Masters of Digital Media Program .....	53
Nintendo of America .....	3
Rad Game Tools .....	C4
Scaleform .....	C2
Trinigy .....	17
University of Advancing Technology .....	55
Vancouver Film School .....	54

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





# THE GENIUS GAME DESIGNER

CAUTION!! GENIUS AT WORK!!!!

**Dear Design Log,**

One of our testers came up to me today and said, "Remind me why we have the fireballs in this game?"

This guy obviously has a lot to learn, especially if he wants to be a designer like me. It's obvious that fireballs not only look cool, they're an important part of any game that takes itself seriously. What awesome game can you think of that doesn't have some kind of fire-based attack in it? Correct. There isn't any.

After further argument, he said there was no "use" for it—fireballs damage enemies the same as the sword. So I decided I would add doors that only open when you throw fireballs at them. That'll be a good reason to have a fireball spell.

—*The Genius Game Designer*

I was just playing WORLD OF WARCRAFT when I was struck by a flash of inspiration: our game should have EPIC BATTLES. Working on the e-mail to send to the rest of the team now.

—*The Genius Game Designer*

Tomorrow will be the first playtest session where we put this creation in front of the unwashed masses. I don't understand what the point of this is, but I guess the publishers wouldn't know brilliance unless other people told them about it. It sucks that I'm supposed to take it seriously, though—they just invite people in off the street! How will rabble like that be able to recognize my art, let alone appreciate it? It's ridiculous.

I asked the studio head why they couldn't recruit playtesters who actually understood the kind of game I'm making here. If we got some of the people who post a lot on the forums on my official website, for example, I think we'd get much, much better results.

It isn't easy being a genius game designer.

**Dear Design Log,**

Today I upped the damage on the Sword of Cutting. But then it made the enemies too easy, so I raised their HP, too.

—*The Genius Game Designer*

Big meeting today to discuss what we should do if the player doesn't time his or her Spell Ring correctly. In order to really encourage them to get it right, I proposed having the spell damage the player instead of the enemy, take control away for 30 seconds while they stagger around, and add the failure to a permanent "number of times failed" statistic that gets uploaded to an online leaderboard ("loserboard," as I like to call it).

Some of the other guys were wondering if that was too much, but if there aren't consequences, it doesn't matter, right? Plus, it's hilarious.

—*The Genius Game Designer*

**Dear Design Log,**

Just dealt with a bunch of playtest feedback.

Some people said it was unclear how to actually enter the Temple of Sorrows, so I put in a request for the sound guys to record a line of your sidekick saying, "Come on, we have to get in!" which will repeat every 10 seconds. The urgency will prompt players to figure it out faster.

There's a lot of misunderstanding about the use of fireballs, even though the loading screen with all of the controller mappings CLEARLY states that fireballs are used to open doors, not harm enemies. Players are so stupid! Anyway, I've now got some help text set to pop up any time the user hasn't cast a fireball for more

than 30 seconds that says "Press X to shoot fireball."

I was upset with the player's comments about being "unengaged" with the story, so I put in more detail as to why the Chogoth Empire blockaded the Council of Riversong in 523 Q.F. and the resulting outcry from the Keepers in some unskippable scrolling text at the beginning of the campaign. That will really inspire players to rally to the Baldoonian cause!

—*The Genius Game Designer*

Time's running out. I've got to put the finishing touches on this gameplay masterpiece I'm creating. The EPIC BATTLES that I wanted never really came through even though, I wrote an e-mail about it a long time ago. I think I'll quadruple the number of enemies throughout the game and increase the rate at which they cast spells at the player to once every couple of ticks.

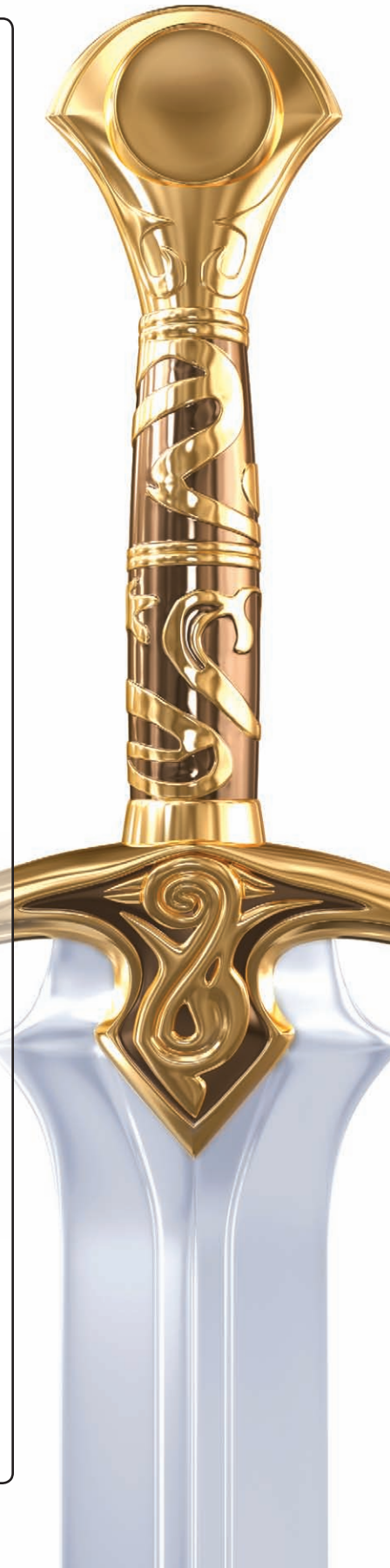
Yeah, so, the lead programmer yelled at me for trying to make our game better. Whatever. It's his fault the lame engine can't handle the awesomeness. Going to go complain to the studio head tomorrow.

**Dear Design Log,**

Well, the reviews are in. I'm pretty proud of what I did, but the game didn't score well because of the bad tech we have and the fact that the sound guys didn't make the spell effects loud enough. I need to find a place that appreciates my talent!

—*The Genius Game Designer*

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





# Reliable Pathfinding Technology. Havok AI.



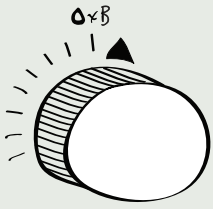
**havok**<sup>™</sup>

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

*Turning Creative Aspirations  
into Technical Realities<sup>™</sup>*



TURN UP THE



# WICKED

AUDIO IN YOUR GAME WITH

# MILES 8

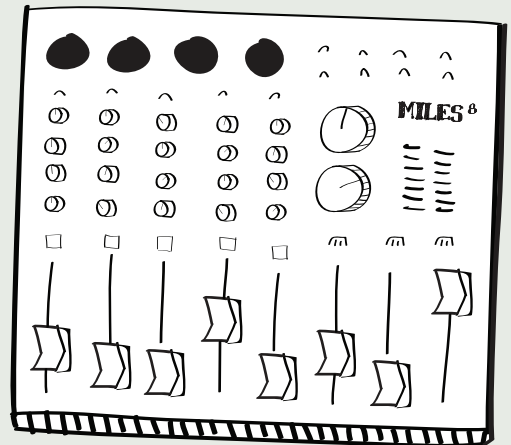
BRAND NEW!



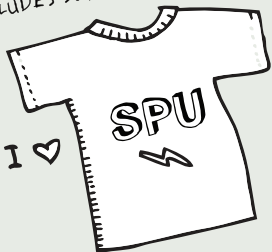
Our multichannel sound system features cool, new

high-level audio tools

+ the world's FASTEST



INCLUDES SUPPORT FOR THE PS3



MP3 and Ogg DECODERS.

USING MILES 8 NOT ONLY MAKES YOU

WANT TO TURN IT UP, IT MAKES YOU rad!



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

(425) 893-4300