

# GAME DEVELOPER MAGAZINE SUPERMEATBOY

THE LEADING GAME INDUSTRY MAGAZINE VOL 10 NO 4  
APRIL 2011 INSIDE: 10TH ANNUAL SALARY SURVEY

gd



GAME DEVELOPER MAGAZINE





# RED 5 STUDIOS

USES MORPHEME

“In Firefall, we are creating a futuristic and fantastical world, but believability remains a priority. If you cut corners when it comes to animation, you are betraying the vision of your game. Morpheme lets us meet our high standards of quality while maintaining a very efficient pipeline.”

Mark Kern, Founder and CEO of Red 5 Studios



morpheme  
advanced animation system



# gd

CONTENTS: 0411  
VOLUME 18 NUMBER 4

GAME DEVELOPER MAGAZINE

## DEPARTMENTS

- 2 **GAME PLAN** *By Brandon Sheffield* [EDITORIAL]  
Living Luminaries
- 4 **HEADS UP DISPLAY** [NEWS]  
G.A.N.G. Award winners, Vietnam curfews online games, and IGF, Choice Award winners.
- 29 **TOOL BOX** *By Tom Curtis* [REVIEW]  
Report From The Show Floor: GDC 2011
- 32 **THE INNER PRODUCT** [PROGRAMMING]  
*By Michael A. Carr, Noel Llopis, Anonymous Programmers*  
Programming Sins
- 36 **PIXEL PUSHER** *By Steve Theodore* [ART]  
The High Art of Games
- 38 **DESIGN OF THE TIMES** *By Damion Schubert* [DESIGN]  
A Player's Stories
- 41 **AURAL FIXATION** *By Jesse Harlin* [SOUND]  
Indie Audio Jonesing
- 42 **THE BUSINESS** *By David Ederly* [BUSINESS]  
Team Players
- 43 **GOOD JOB!** *By Brandon Sheffield* [CAREER]  
Tom Russo Q&A, Who Went Where, and New Studios
- 45 **EDUCATED PLAY** *By Jeffrey Fleming* [EDUCATION]  
One Man Down's SOLACE
- 48 **ARRESTED DEVELOPMENT** *By Matthew Wasteland* [HUMOR]  
How To Annoy Your Producers!

## POST MORTEM

- 14 **SUPER MEAT BOY**  
SUPER MEAT BOY represents the work of two persons (plus a musician), which went on to sell some 400,000 units on XBLA and PC worldwide. Fighting through illness and fatigue, this is a story of success from a team that urges you to "have fun" making your game.  
*By Edmund McMillen and Tommy Refenes*

## FEATURES

- 7 **10TH ANNUAL SALARY SURVEY**  
Our popular survey is back, now in its 10th year of sharing the average salaries of game developers worldwide across all disciplines (though focusing on North America). Indies, for the second time, are also reported herein.  
*By Ryan Newman and Brandon Sheffield*
- 21 **CREATOR OF WORLDS**  
MINECRAFT-like procedural terrain generation is an interesting subject, but where to start? Joshua Tippetts poses that through combining various functions, you can create a tweakable procedural world.  
*By Joshua Tippetts*





# LIVING LUMINARIES

## WHY DO WE HOLD UP SOME VETERAN GAME DEVELOPERS ABOVE OTHERS?

**AT THIS YEAR'S GDC—THE 25th—** there were 11 postmortems of vintage games from important creators, all of them incredibly inspiring. This prompted me to think—why do we hold these folks in such high regard? They deserve to be looked upon with respect, there's no doubt in my mind. But when you break it down, Toru Iwatani created PAC-MAN, and that's pretty much it. Other luminaries like Jordan Mechner (PRINCE OF PERSIA) and ERIC CHAHI (ANOTHER WORLD) continue to work on games, but have few titles to their credit.

Why do we revere certain people in the game industry? Is it because of the high quality of the games they created? Is it because of how influential they were? Do we color those past experiences with our memories? All of these are factors—but I think overall, our reverence for these people has to do with a pioneering spirit.

### WILD, WILD WEST

» I think most of us in the game industry have some sort of predisposition to thoughts of fantasy. To imagine, fantasize, and dream is human. But as working game developers, we actually make these visions come to life in ways that most people can only, well, dream about. So to us, that fantasy is closer at hand.

We idolize cowboys, samurai, explorers, inventors, and astronauts because they fulfill that need for adventure and exploration we all feel. I don't know about everyone else, but part of why I work on games, and help to create worlds, is because of an urge to explore. I want to make my mark, and be the first one to step onto that alien soil and discover its secrets. Because after all, no matter how carefully we craft our games, there are always secrets, little twists of the world that we never anticipated.

MMOs, open-world games, and sandboxes like LOVE and MINECRAFT are fantasy generators. They

make us feel like the pioneers we admire. Coming upon some area you didn't know existed feels like you're discovering the lost city of Atlantis—if only for a moment. Forging those worlds has a similar feel, but most of us can't do that alone. We need a MINECRAFT to give us the tools, or else we need a team of artists behind us to create the assets that bring our code to visual life. Or as artists, you need coders and designers to help craft that universe.

We have become extremely compartmentalized in our work, which is to the benefit of the large-scale games we create. Having a dedicated writer has proved to be very successful for certain teams. Dedicated network coders are hard to find, but an absolute necessity for most persistently online games. But sometimes that compartmentalization can diminish the feeling of creation and exploration.

So we do hold up these luminaries for their pioneering spirit, but also for the fact they did nearly everything themselves, before tools to do so were even invented.

### THE WILL TO CREATE

» Iwatani's PAC-MAN was an early effort in fooling players into believing games had complex AI, while also proving the power of distinctive character combined with tight control. Jordan Mechner knew he wouldn't be able to create perfect animation with the tools he had at his disposal at the time, so he filmed his brother running and jumping, then shrank the data and made pixel versions that have an incredible fluidity to them. Eric Chahi pushed polygons on early computers in an early stage, and where assembly failed him, constructed code language of his own, to create a one of the most cinematic games of the era.

Of course, it helps that all these games were excellent examples of what could be done with

technology and design innovation in their respective eras. But I think what really fascinates us is that these creators managed to make these innovative games largely by themselves. The tools, designs, and art techniques they needed didn't exist, so they willed them into existence. These are persons of vision, with the ability to back that vision up with hard work and provable results. And who wouldn't respect that?

The do-it-yourselfer will always be someone to learn from, no matter what his or her era of prominence. But when they can continue those ideas through to the future, and their original pioneering ideas are still applicable, you see that they're not just one-hit wonders. Sure, there are those in the industry who happened to hit on something at the right time, and made an impact in their day, but cease to retain relevance. I think those are fewer and further between than those who do something great because that's what's in them, and whether they move on to education as Iwatani has, or continue to pioneer fluid gameplay like Chahi with his new game FROM DUST, their ideas remain strikingly relevant.

This is what I've taken away from the vintage postmortems at GDC. What these fellows did back then was amazing—but what they think and do now is just as interesting. Every time I've spoken to Eric Chahi about modern game design, he's revealed to me something I wouldn't have thought of, even in genres he's never worked on.

We can aspire to greatness ourselves, and this is part of current the indie fervor, I believe. If we apply what we know and try to make something different on our own—something that comes from within—we could be presenting our own vintage postmortems at the 50th GDC. Would anyone be surprised to see MINECRAFT's Markus Persson there? 🙄

—Brandon Sheffield  
twitter: @necrosotfy



United Business Media  
303 Second Street, Suite 900, South Tower  
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)

FOR DIGITAL SUBSCRIPTION INFORMATION  
[www.gdmag.com/digital](http://www.gdmag.com/digital)

### 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)

**PRODUCTION INTERN**  
Tom Curtis

### CONTRIBUTING EDITORS

Jesse Harlin  
Steve Theodore  
Kim Pallister  
Dave Cowling  
Soren Johnson  
Damion Schubert

### ADVISORY BOARD

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

### ADVERTISING SALES

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

**MEDIA ACCOUNT MANAGER**  
John Malik Watson | [jmwatson@think-services.com](mailto:jmwatson@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 MEDIA  
Ryan Pratt | [rp Pratt@wrightsreprints.com](mailto:rp Pratt@wrightsreprints.com)  
t: 877.652.5295

### AUDIENCE DEVELOPMENT

TYSON ASSOCIATES Elaine Tyson  
e: [Elaine@tysonassociates.com](mailto:Elaine@tysonassociates.com)

LIST RENTAL Merit Direct LLC  
t: 914.368.1000







Our programs and initiatives are helping Ontario's interactive digital media companies create a thriving industry, a global audience and a wealth of opportunity. Be part of it. **OMDC.on.ca**

# We've got it going



Ontario Media  
Development Corporation



## ninth annual g.a.n.g. award winners announced

### Game Developer's Jesse Harlin wins "Best Article"

\\ The winners of the ninth annual Game Audio Network Guild awards have been announced, with Rockstar Games' RED DEAD REDEMPTION and DICE's BATTLEFIELD: BAD COMPANY 2 taking home multiple awards. The awards, held at the recent GDC 2011, represent "the best audio in video games from 2010." The final winners were chosen by a 70-person advisory committee, made up of members of the non-profit GANG organization.

RED DEAD REDEMPTION was the big winner of the event, bagging the Audio of the Year, Music of the Year, Best Interactive Score, and Best Dialogue awards. BATTLEFIELD: BAD COMPANY 2 won two awards, while LucasArts' MONKEY ISLAND 2 SPECIAL EDITION: LECHUCK'S REVENGE, Blizzard's STARCRAFT II: WINGS OF LIBERTY and 2K Games' BIOSHOCK 2 took one award each.

Game Developer Magazine also won an award for Best Game Audio Article, Publication, or Broadcast for Jesse Harlin's "The Weight of Silence - How Silence Can Indicate a Character's Importance," published in our December 2010 issue.

GANG president Paul Lipson said "The quality bar from 2010 was so high across the board, it was impossible to predict specific wins. With over 350 submissions this year, just making it to the final nomination process is something all the teams and publishers can be proud of." — Mike Rose

#### Audio of the Year

##### RED DEAD REDEMPTION

- Rockstar Games
- Composers: Bill Elm & Woody Jackson; Lead Audio Designer: Jeffrey R. Whitcher; Audio Designers: Steven von Kampen, Christian Kjeldsen, Corey Ross; Audio Programmers: Corey Shay, Robert Katz

#### Music of the Year

##### RED DEAD REDEMPTION

- Rockstar Games
- Bill Elm, Woody Jackson

#### Best Audio Other:

##### HALO: WAYPOINT "THE RETURN"

- 343 Industries/Microsoft Game Studios
- Kristofor Mellroth, Senior Audio Director, Microsoft Game Studios; Paul Lipson, Audio Director, Pyramid Studios; Peter Steinbach, Steve Heithecker, David Earl, Michael Roache

#### Sound Design of the Year

##### BATTLEFIELD: BAD COMPANY 2

- Electronic Arts/DICE
- Stefan Strandberg, Ben Minto, David Mollerstedt, Thomas Danke, Mari Saastamoinen, Olof Stromqvist

#### Best Interactive Score

##### RED DEAD REDEMPTION

- Rockstar Games
- Bill Elm, Woody Jackson

#### Best Handheld Audio

##### MONKEY ISLAND 2 SPECIAL EDITION: LECHUCK'S REVENGE

- LucasArts
- Tom Bible, Jesse Harlin, Wilbert Roget II, Jeff Ball, Dan Reynolds, Andrew Aversa

#### Best Soundtrack Album

##### VIDEO GAMES LIVE - LEVEL 2

- Tommy Tallarico, Jack Wall

#### Best Original Instrumental:

##### "ATHENS HARBOUR CHASE" – JAMES BOND 007: BLOOD STONE

- Activision
- Richard Jacques

#### Best Cinematic/Cutscene Audio

##### STARCRAFT II: WINGS OF LIBERTY

- Blizzard Entertainment
- Russell Brower, Paul Menichini, David Farmer

#### Best Use of Licensed Music

##### BIOSHOCK 2

- 2K Games/2K Marin
- Michael Kamper, Audio Lead and the 2K Marin Audio Team

#### Best Original Vocal - Pop

##### "I'LL TAKE IT ALL" - JAMES BOND 007: BLOOD STONE

- Activision
- Dave Stewart

#### Best Original Vocal - Choral

##### "INVINCIBLE" – WORLD OF WARCRAFT: CATAclySM

- Blizzard Entertainment
- Music by Russell Brower, Jason Hayes; Lyrics by Derek Duke, Neal Acree

#### Best Dialogue

##### RED DEAD REDEMPTION

- Rockstar Games
- Lead Audio: Matthew Smith; Additional Dialogue Editing: Will Morton, Allan Walker, Jon McCavish; Audio Designer: George Williamson; Dialogue Assistant: Lindsay Robertson

#### Best Game Audio Article, Publication or Broadcast

##### "THE WEIGHT OF SILENCE - HOW SILENCE CAN INDICATE A CHARACTER'S IMPORTANCE" – GAME DEVELOPER MAGAZINE

- Jesse Harlin

#### Best Use of Multi-Channel Surround In a Game

##### BATTLEFIELD: BAD COMPANY 2

- Electronic Arts/DICE
- Stefan Strandberg, Ben Minto, David Mollerstedt, Thomas Danke, Mari Saastamoinen, Olof Stromqvist

#### G.A.N.G. Recognition Award

- Sumthing Else Musicworks

#### G.A.N.G. Distinguished Service Award

- Dren McDonald, Jacquie Shriver

#### Rookie of the Year Award

- Woody Jackson, Bill Elm

#### Lifetime Achievement Award

- Chris Huelsbeck

## vietnamese government puts curfew on online gaming

\\ As of March 3, gamers in Vietnam might be getting a bit more sleep. A government ministry has asked internet service providers to block access to all online games between the hours of 10 PM and 8 AM.

According to a report in Viet Nam News, the Ministry of Information and Communication is concerned about the impact of online activity on the nation, particularly when it comes to young

people and online games.

The government group will be monitoring online game activities during the prohibited hours and could cancel services that allow people to play, according to the ministry's Deputy Minister, Le Nam Thang.

Service providers argue that the access block is unfair to their customers who've paid for access to entertainment, and also have argued that it makes the

maintenance of online games more difficult for the region's operators.

The report pegs Vietnam's number of internet users at about 23 million, or 23 percent of the total population. Southeast Asia in general is a hotbed for MMO growth; Pearl Research has projected that the Vietnamese and Indian online gaming populations together will reach 25 million by 2014.

— Leigh Alexander





# igf, game developers choice award winners announced

At the Independent Games Festival Awards and the Game Developers Choice Awards at this year's GDC, Mojang's indie-hit MINECRAFT earned the IGF's Seumas McNally Grand Prize, while Rockstar San Diego's RED DEAD REDEMPTION took home the Choice award for Game of the Year.

At the IGF ceremony, MINECRAFT and Frictional Games' horror title AMNESIA: THE DARK DESCENT collectively took home five of the event's ten awards, with MINECRAFT winning the Audience award in addition the Grand Prize, while AMNESIA won for Technical Excellence, Excellence in Audio, and was granted the Direct2Drive Vision Award.

Rockstar Games' RED DEAD REDEMPTION was the biggest winner at the Game Developers Choice Awards, taking home three additional awards for Best Audio, Best Game Design, and Best Technology.

MINECRAFT took home awards in three categories: Best Debut Game, Best Downloadable Game, and the Innovation Award, in addition to its awards at the IGF.

## IGF WINNERS

Best Student Game

**FRACT**

RICHARD E FLANAGAN

Excellence in Design

**DESKTOP DUNGEONS**

CF DESIGN

Technical Excellence

**AMNESIA: THE DARK DESCENT**

FRICIONAL GAMES

Best Mobile Game

**HELISING'S FIRE**

RATLOOP

Excellence in Visual Art

**BIT.TRIP RUNNER**

GALJIN GAMES

Excellence in Audio

**AMNESIA: THE DARK DESCENT**

FRICIONAL GAMES

Direct2Drive Vision Award

**AMNESIA: THE DARK DESCENT**

FRICIONAL GAMES

Audience Award

**MINECRAFT**

MOJANG

IGF Nuovo Award

**NIDHOGG**

MESSHOF

Seumas McNally Grand Prize

**MINECRAFT**

MOJANG

## GAME DEVELOPERS CHOICE AWARD WINNERS

Best Audio

**RED DEAD REDEMPTION**

ROCKSTAR GAMES

Best Debut Game

**MINECRAFT**

MOJANG

Best Writing:

**MASS EFFECT 2**

BIOWARE

Best Game Design

**RED DEAD REDEMPTION**

ROCKSTAR GAMES

Best Downloadable Game

**MINECRAFT**

MOJANG

Best Visual Art

**LIMBO**

PLAYDEAD STUDIOS

Best Technology

**RED DEAD REDEMPTION**

ROCKSTAR GAMES

Best Handheld Game

**CUT THE ROPE**

CHILLINGO

Innovation Award

**MINECRAFT**

MOJANG

Game of the Year

**RED DEAD REDEMPTION**

ROCKSTAR GAMES



# UNREAL TECHNOLOGY NEWS

BY Mark Rein  
Epic Games, Inc.



## STAN LEE AND HOUSE OF MOVES PICK UE3

Entertainment legend Stan Lee and Vicon House of Moves (HOM) recently enlisted Unreal Engine 3 (UE3) to create Lee's new franchise, *The Guardian Project*. In a special collaboration with the National Hockey League (NHL), Lee and HOM crafted the world of the Guardians, 30 new hockey-based superheroes that made their debut at the 58th Annual NHL All-Star Game. The Guardians' first appearance came in the form of a short film that kicked off a media blitz that is slated to include an online video game, a computer-animated TV show and a slew of merchandising.

HOM headed up development of the Guardians, relying heavily on performance-capture technology to create realistic characters that would work across a range of media for television and online broadcast, stadium displays and virtual reality experiences.

"We chose Unreal for its ease of use," said Peter Krygowski, director, HOM. "The learning curve to ramp up production and fit it into our pipeline was minimal."

Ease of use was particularly important to HOM since it's primarily a motion capture and animation shop, without a huge infrastructure for handling the rendering requirements for traditional high-end output.

Fortunately, Krygowski had been close to the video game world for more than a decade, had plenty of experience with a number of proprietary engines and knew exactly what his team needed to successfully pull off *The Guardian Project*.

UE3 delivered the flexibility the team was looking for.

"We wrote several pieces of code to help generate custom shaders and to be able to bring virtual cameras into and out of the Unreal Engine for the purposes of this project," said Alberto Menache, HOM's visual effects supervisor and pipeline developer. "As a result, we had incredible creative flexibility, and could render out 8,000 frames in a matter of seconds—not to mention the savings in gear costs without the need for a multi-CPU render farm."

CG assets for the short film were built using Autodesk Maya and Pixologic ZBrush, with Autodesk's MotionBuilder brought in to retarget animation and navigate environments during motion capture sessions.

HOM captured stunts and poses for each of the 30 Guardian superheroes at their 26,000 square feet of motion-capture stages, outfitted with more than 200 Vicon T160 cameras over nine days of mo-cap shooting. The project was completed over six months with a creative team that started at 10 and grew to 200 at the project's peak.

"I can't emphasize enough what an impact Unreal had on this project," said Krygowski. "The real-time lighting, ease of use, ability to iterate quickly and near-time rendering of final assets allowed us to accelerate an already compressed delivery schedule. For the short film, we needed to deliver a three-and-a-half minute animation short in two and a half months, from start to finish. It's a project that would normally have taken six months."

*The Guardian Project* brought some unexpected drama, in addition to the punishing timeline. Just 48 hours before the film was set to premiere at the NHL All-Star Game, an outside vendor delivered six shots that didn't fit with the video. But using Unreal, HOM was able to revise, reanimate and re-render the shots. The team

made the changes, passed them through Unreal, and composited the final animation in time for the final piece. According to Krygowski, without Unreal this wouldn't have been an option.

Krygowski says he expects more Hollywood productions to build Unreal into their pipelines, since it allows for collapsed production time when necessary, while still allowing for robust iteration. Plus, with Unreal the assets are more easily shared between different mediums, from games to broadcast.

According to Brian Rausch, HOM's vice president of production, "you have to think down the road of the possibility of extrapolating characters and environments into game assets, or a television series, making sure you can easily flow the CG creative elements between mediums. By building scenes in a game engine from the start, our options are just much broader."

Mark Rein  
Epic Games, Inc.



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

Epic's Unreal Engine 3 has won Game Developer magazine's Best Engine Front Line Award five times along with entry into the

Hall of Fame. UE3 has won three consecutive Develop Industry Excellence Awards.

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

Follow @MarkRein on Twitter.

### UPCOMING EPIC ATTENDED EVENTS

**East Coast  
Game  
Conference**  
Raleigh, NC  
April 13-14, 2011

**E3 Expo**  
Los Angeles  
June 7-9, 2011

**Casual  
Connect  
Seattle**  
Seattle, WA  
July 19-21, 2011

**Comic-Con  
International**  
San Diego, CA  
July 21-24, 2011

Please email: [mrein@epicgames.com](mailto:mrein@epicgames.com) for appointments.

WWW.EPICGAMES.COM





# TENTH ANNUAL SALARY SURVEY

**TEN YEARS AGO WE BEGAN SURVEYING EMPLOYEES THROUGHOUT THE INDUSTRY'S VARIOUS** disciplines regarding their salaries, including additional benefits and alternate sources of revenue, employment status, and general position, thereby creating a snapshot of the industry each year. We continue the tradition through to this, the 10th annual Salary Survey from *Game Developer*.

This year's big number is the rise in the average salary across all disciplines and experience levels, with 2010's \$5,244 gain having made for a strong rebound from the near \$4,000 loss in 2009, bringing the latest average above 2008. The number of respondents whose salaries increased in 2010 was up across the board from 2009, with the biggest increase coming from those in production, 73 percent of whom reported higher income than last year.

In addition, 47 percent of respondents agreed that there are more opportunities for developers than ever before, and 73 percent said that the industry is still great to work in, showing that there is plenty to be optimistic about as we head into 2011.

This year was one of proving for the social game space, and we believe that contributed somewhat to the overall raise in salary across all disciplines. Meanwhile, the indie segment

has continued to rise in prominence, as a source of opportunity and employment for those looking for a different path, after making their mark on downloadable and browser-based platforms. Last year, we included indie developers and independent contractors in their own listing, a practice we continued this year, though with a slightly lower response.

A major takeaway from the comments section of the survey reveals that while in general salaried developers are making more money, independent developers are a lot happier with their lot in life.

Those who lost their jobs this year may want to take heed of those words and find out what it is they truly want to do in games.

—Ryan Newman and Brandon Sheffield



# TENTH ANNUAL

## programmers

AVERAGE SALARY  
**\$85,733**

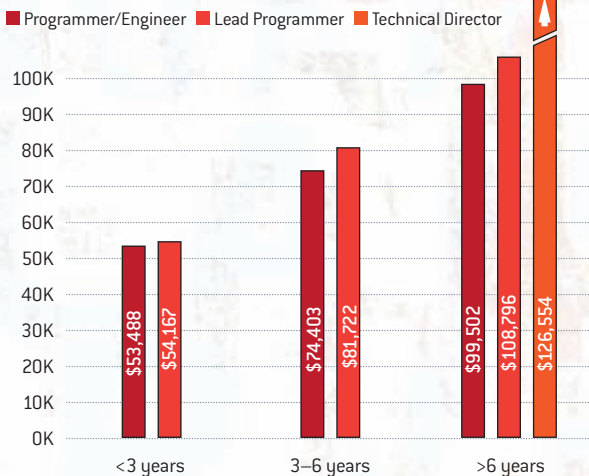
**PROGRAMMERS ARE THE BACKBONE OF THE INDUSTRY, AND THEIR** hard work is certainly rewarded, as the profession continues to be one of the highest paid in the industry, though this year coders have been eclipsed by producers in salary levels.

Overall, average programmer salaries increased some \$5,000 over 2009, with gains pretty evenly split across disciplines. But entry level programmers (those with less than three years' experience), saw an overall drop of around \$1,000.

Minor though that drop may be, this fall in salary combined with a rise in the number of respondents in the entry-level categories is likely an indicator that companies are hiring more fresh-faced computer science graduates at lower pay rates than before.

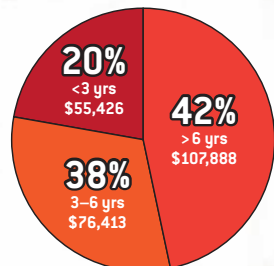
Programmers in Canada fared better in 2010, earning \$74,473 in 2010, up from \$67,937 (USD) in 2009. European programmers also saw a rise, earning \$48,230 (USD) on average.

### Programmer salaries per years experience and position



### ALL PROGRAMMERS AND ENGINEERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **77%**

Average additional income: **\$17,689**

#### Type of additional compensation received

Annual bonus	51%
Pension/Employer contribution to Retirement plan	49%
Profit sharing	16%
Project/title bonus	27%
Royalties	12%
Stock options/equity	37%

Percent receiving benefits: **94%**

#### Type of benefits received

Gender	Percent Represented	Average Salary
Male	96%	\$86,140
Female	4%	\$74,559

Medical	99%
Dental	93%
401K/Retirement	84%

## artists and animators

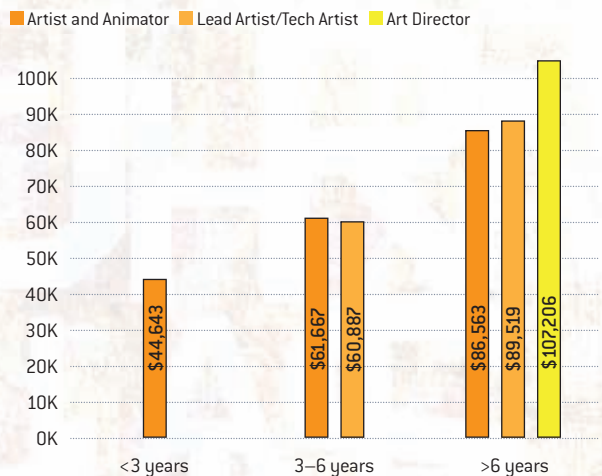
AVERAGE SALARY  
**\$71,354**

**THE AVERAGE SALARY FOR ARTISTS WAS UP ONLY SLIGHTLY FROM LAST** year, with the bulk of that increase coming from a bump in the income for art directors.

However, increased earnings were not across the board, as artists, animators, and leads all saw their average salaries drop. The exception was artists and animators with three to six years of experience, who saw a small increase from \$61,121 to \$61,667 in 2010. The biggest decrease was found amongst lead artists and tech artists with over six years of experience, with the average salary falling to \$89,519 in 2010 from \$97,206 in 2009.

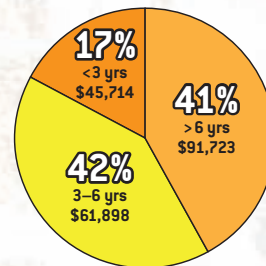
Canadian artists found their salaries increasing on average by \$3,877, up to \$63,277 (USD). The increase was largely found amongst artists and animators, whose salaries increased from \$50,565 in 2009 to \$56,630 (USD) in 2010. European artists also found themselves earning more, with an increase of \$3,459 from 2009, bringing the average salary up to \$41,611 (USD).

### Artist and Animator salaries per years experience and position



### ALL ARTISTS AND ANIMATORS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **74%**

Average additional income: **\$12,711**

#### Type of additional compensation received

Annual bonus	47%
Pension/Employer contribution to Retirement plan	49%
Profit sharing	16%
Project/title bonus	39%
Royalties	16%
Stock options/equity	33%

Percent receiving benefits: **94%**

#### Type of benefits received

Gender	Percent Represented	Average Salary
Male	89%	\$72,924
Female	11%	\$59,224

Medical	99%
Dental	93%
401K/Retirement	80%



# SALARY SURVEY

## game designers

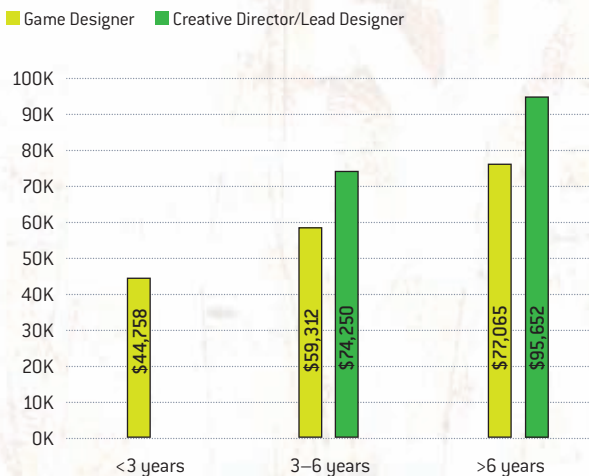
AVERAGE SALARY  
**\$70,223**

**GAME DESIGNERS, CREATIVE DIRECTORS, AND WRITERS RECEIVED** a slight boost from last year. Leads and creative directors with less three-to-six years of experience had an average increase of \$5,083, while those with over six years dropped from \$101,810 in 2009 to \$95,652. This could potentially be an indicator of some higher-level designers either leaving the industry, or moving up into management.

Overall, designers across all experience ranges saw little movement, as design has been one of the most stable positions as far as compensation throughout our survey. All told, 66 percent of those surveyed reported at least a slight increase in pay from last year.

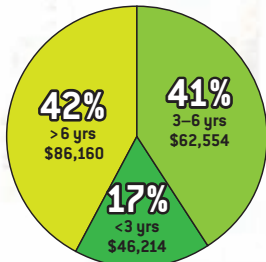
Designers working out of Canada experienced a decrease in pay, with the average salary falling from \$61,520 in 2009 to \$58,319 (USD) in 2010. European designers also had lower incomes but fared slightly better with an average salary of \$41,250 (USD), down \$1,173 from 2009.

### Game Designer salaries per years experience and position



### ALL GAME DESIGNERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **75%**

Average additional income: **\$14,259**

#### Type of additional compensation received

Annual bonus.....**42%**  
Pension/Employer contribution to Retirement plan.....**40%**  
Profit sharing.....**16%**  
Project/title bonus.....**34%**  
Royalties.....**17%**  
Stock options/equity.....**35%**

Percent receiving benefits: **96%**

#### GENDER STATS FOR DESIGNERS

Gender	Percent Represented	Average Salary
Male	93%	\$72,924
Female	7%	\$59,224

Type of benefits received	Percent
Medical.....	<b>96%</b>
Dental.....	<b>92%</b>
401K/Retirement.....	<b>80%</b>

## producers

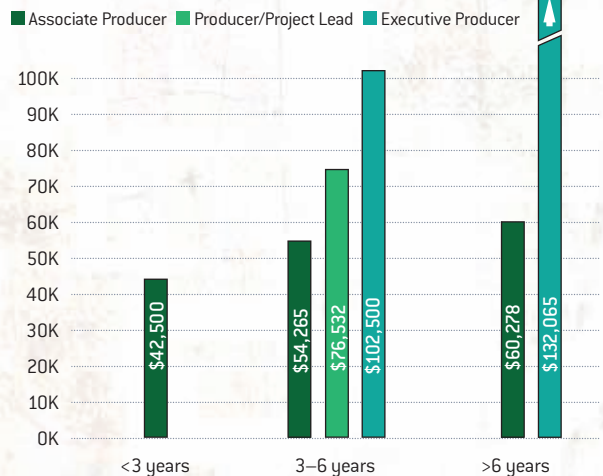
AVERAGE SALARY  
**\$88,544**

**AFTER AN OVERALL AVERAGE SALARY DIP IN 2009, PRODUCERS** rebounded with an increase of \$13,462. Seventy-three percent of respondents reported an increase in their salary. This could be due to the fact that over half our respondents reported having over six years of experience, but also may indicate the shift toward social games, which pay producers web 2.0 salaries. Executives, producers, and project leads with over six years' experience all had marked increases: \$28,454 and \$7,344 respectively.

Production also had the second-highest percentage of additional compensation, at 83 percent, second only to business' 85 percent.

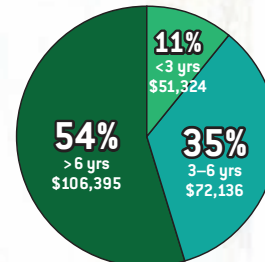
Canadian producers reported a significant decrease in salary, with the average dropping from \$87,130 in 2009 to \$72,500 (USD) in 2010. Producers in Europe had a slight increase in 2010 with an average of \$52,884 (USD) and 56 percent reporting a salary increase.

### Producer salaries per years experience and position



### ALL PRODUCERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **83%**

Average additional income: **\$16,223**

#### Type of additional compensation received

Annual bonus.....**61%**  
Pension/Employer contribution to Retirement plan.....**41%**  
Profit sharing.....**13%**  
Project/title bonus.....**32%**  
Royalties.....**7%**  
Stock options/equity.....**42%**

Percent receiving benefits: **96%**

#### GENDER STATS FOR PRODUCERS

Gender	Percent Represented	Average Salary
Male	83%	\$90,744
Female	17%	\$77,870

Type of benefits received	Percent
Medical.....	<b>97%</b>
Dental.....	<b>95%</b>
401K/Retirement.....	<b>85%</b>



# TENTH ANNUAL

## audio professionals

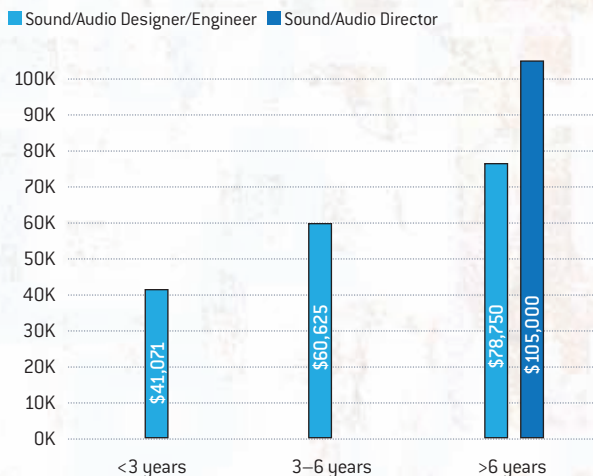
AVERAGE SALARY  
**\$68,088**

OF THE AUDIO PROFESSIONALS SURVEYED, 15 PERCENT REPORTED earning less than they did the previous year, the highest of any discipline. There was a slight uptick in respondents this year, in a category which typically has a low response rate due to the low number of full-time audio professionals in games, but numbers are still low, so it is difficult to gauge with absolute certainty.

Audio developers continue to be the least likely to receive additional benefits, such as health insurance. However, they were the most likely to receive royalties for their work, with the reported 25 percent significantly higher than other disciplines, with game design coming in second at 17 percent.

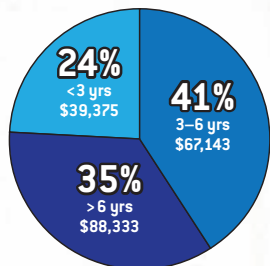
Canadian audio developers reported earning more in 2010, with the average salary increasing from \$61,250 to \$68,571 (USD). European audio developers reported an increase in average salary, up \$6,111 to \$46,944, with 50 percent earning more in 2010.

### Audio Developer salaries per years experience and position



### ALL AUDIO DEVELOPERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **82%**

Average additional income: **\$7,570**

#### Type of additional compensation received

Annual bonus	50%
Pension/Employer contribution to Retirement plan	46%
Profit sharing	21%
Project/title bonus	32%
Royalties	25%
Stock options/equity	18%

#### GENDER STATS FOR AUDIO DEVELOPERS

Gender	Percent Represented	Average Salary
Male	94%	\$70,469
Female	6%	\$30,000

Percent receiving benefits: **88%**

Type of benefits received	Percent
Medical	100%
Dental	100%
401K/Retirement	80%

## qa testers

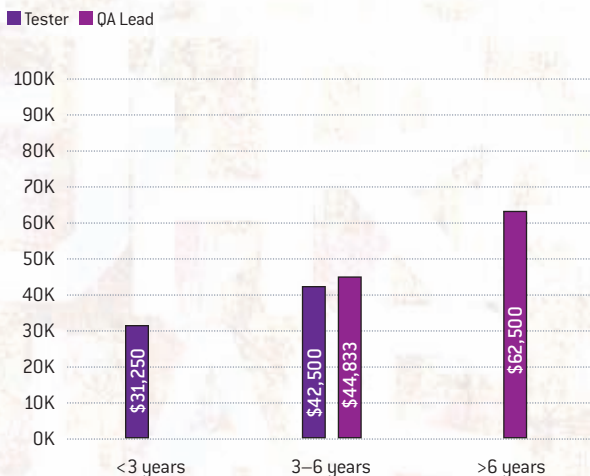
AVERAGE SALARY  
**\$49,009**

HOME TO MANY ENTRY-LEVEL POSITIONS, QUALITY ASSURANCE remains one of the lowest-paid disciplines. However, testers were rewarded in 2010 with an increase in salary and benefits.

Many QA professionals are on contract, so the entire range may not be represented here, and the fact that QA leads are the most likely to be salaried could potentially explain the increase. Like producers though, the bump could come from those working in the web industries, with companies such as Zynga having long-hours QA needs. Web developers in general tend to be paid a little better than their counterparts in traditional video games.

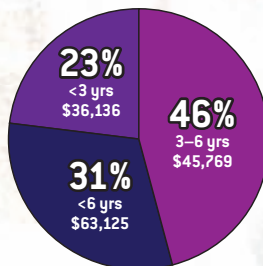
Canadian testers did not benefit as much as those in the United States with the average salary reporting as having dropped from \$39,375 to \$37,857 (USD) in 2010. European testers benefited from an increase of \$7,222, bringing the average salary to \$37,222 (USD).

### QA Tester salaries per years experience and position



### ALL QA TESTERS

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **73%**

Average additional income: **\$7,824**

#### Type of additional compensation Received

Annual bonus	69%
Pension/Employer contribution to Retirement plan	56%
Profit sharing	16%
Project/title bonus	19%
Royalties	9%
Stock options/equity	31%

#### GENDER STATS FOR QA TESTERS

Gender	Percent Represented	Average Salary
Male	95%	\$48,200
Female	5%	\$62,500

Percent receiving benefits: **93%**

Type of benefits received	Percent
Medical	95%
Dental	98%
401K/Retirement	90%



# SALARY SURVEY

## business and legal people

AVERAGE SALARY  
**\$106,452**

THOSE SURVEYED IN THE BUSINESS AND LEGAL DISCIPLINES INCLUDE chief executives and executive managers, community managers, marketing, legal, human resources, IT, content acquisition and licensing, and general administration staff.

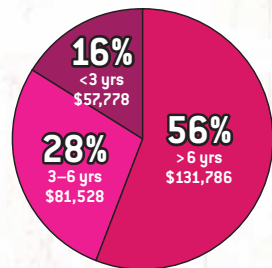
Those in business were most likely to receive any additional compensation (85 percent). Business not only had the highest average salary, but it also led in average salary across all experience levels. Of the disciplines surveyed, business also had the highest percentage of those with six or more years of experience, at 55.6 percent. It seems as though money always filters up.

The business, marketing, and legal arena is also where the second-most women can be found, dwarfed only by production's 17%.

Canadian business personnel fared well with an increased average salary of \$85,312 (USD). Business persons in Europe also saw an increase, up from \$59,231 to \$63,235 (USD) in 2010.

### ALL BUSINESS AND LEGAL PEOPLE

#### YEARS EXPERIENCE IN THE INDUSTRY



Percent receiving additional income: **85%**

Average additional income: **\$28,972**

#### Type of additional compensation Received

Annual bonus	73%
Pension/Employer contribution to Retirement plan	35%
Profit sharing	23%
Project/title bonus	17%
Royalties	7%
Stock options/equity	37%

#### GENDER STATS FOR BUSINESSPEOPLE

Percent receiving benefits: **93%**

Gender	Percent Represented	Average Salary	Type of benefits received
Male	86%	\$110,849	Medical: 100%
Female	14%	\$80,556	Dental: 95%
			401K/Retirement: 76%

## LAYOFFS

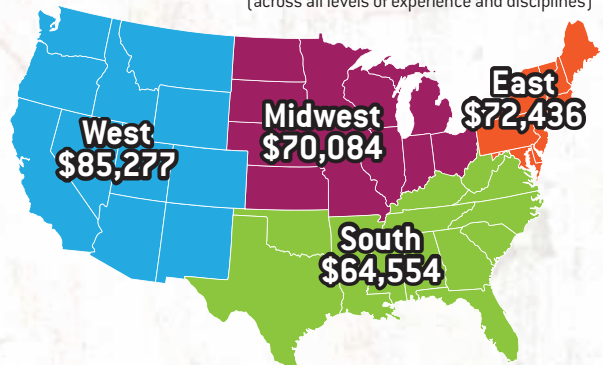
OF THE ALMOST 4,000 SURVEYED DEVELOPERS, 14 PERCENT HAD been laid off at one point or another in 2010. That is a 5 percent decrease from 2009's 19 percent, but it is still higher than 2008's 12 percent.

Fifty-two percent of those laid off were able to find employment at a game studio or publisher, while 16 percent were unable to find new jobs in the industry. More developers (23 percent) also found themselves going into contracting and consulting in 2010, up from 17 percent in 2009. Thirteen percent went on to found or cofound a company, up from 10 percent in 2009.

Developers also went into indie development in greater numbers (19 percent), up from 16 percent in 2009. The increased amount of developers going into independent and contract work combined is up 9 percent over 2009, another strong indicator in the rise of development outside the traditional developer-and-publisher venue.

## 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	\$86,772	35%	\$108,061
2 Washington	\$85,536	51%	\$103,343
3 New Jersey	\$73,409	54%	\$79,167
4 Virginia	\$92,000	58%	\$95,833
5 Oregon	\$71,288	51%	\$94,265
6 Maryland	\$74,583	39%	\$94,605
7 Florida	\$57,500	33%	\$81,500
8 Massachusetts	\$74,049	39%	\$90,081
9 Illinois	\$70,288	53%	\$85,000
10 Wisconsin	\$69,891	60%	\$80,714

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

	EAST	MIDWEST	SOUTH	WEST
Programmer	\$77,630	\$71,000	\$68,636	\$96,651
Art and Animation	\$62,756	\$52,500	\$62,692	\$77,942
Game Design	\$67,125	\$68,889	\$58,032	\$76,560
Production	\$80,900	\$62,500	\$69,444	\$94,929
Audio	\$62,500	\$85,000	\$50,000	\$73,636
QA	\$49,643	—	\$35,833	\$47,167
Business	\$109,265	\$106,667	\$91,944	\$111,645

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

	EAST	MIDWEST	SOUTH	WEST
Homeowners	\$90,479	\$82,917	\$79,754	\$103,917
Non-Homeowners	\$61,113	\$54,625	\$50,733	\$71,365

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

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

	U.S.	CANADA*	EUROPE**
Programmer	\$85,733	\$74,474	\$48,231
Art and Animation	\$71,354	\$63,278	\$41,611
Game Design	\$70,223	\$58,320	\$41,250
Production	\$88,544	\$72,500	\$52,885
Audio	\$68,088	\$68,571	\$46,944
QA	\$49,009	\$37,857	\$37,222
Business	\$106,452	\$85,313	\$63,235

\*Most Canadian respondents were from British Columbia, Quebec, and Ontario.

\*\*Most European respondents were from the United Kingdom (26%), France (15%), Germany (10%), Spain (9%), The Netherlands (5%), and Italy (5%).



# TENTH ANNUAL

## AVERAGE SALARY BY EDUCATION LEVEL AND DISCIPLINE

(across all levels of experience)

	PROGRAMMING	ART	DESIGN	PRODUCTION	AUDIO	QA	BUSINESS
High school/GED	\$93,929	—	\$67,500	—	—	\$52,500	\$119,167
Some College	\$94,457	\$79,000	\$71,667	\$89,224	—	\$49,286	\$111,750
Associates Degree	\$93,571	\$73,654	\$70,682	—	—	\$49,643	\$96,667
Bachelors Degree	\$80,908	\$70,299	\$68,772	\$82,310	\$64,250	\$42,717	\$101,379
Some Graduate	\$96,528	\$80,577	\$76,250	\$110,313	—	—	\$112,500
Masters Degree	\$92,703	\$58,056	\$74,352	\$90,000	—	—	\$131,563
Some Doctoral	\$78,750	—	—	—	—	—	—
Doctoral Degree	\$102,500	—	—	—	—	—	\$84,167

## METHODOLOGY

**NOW IN ITS TENTH YEAR,** the *Game Developer Salary Survey* was conducted in February 2011 for the fiscal year January 1, 2010 through December 31, 2010 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 survey.

We gathered 3,781 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. The usable sample reflected among salaried employees in the U.S. was 1,343, for Canada 276, and for Europe 404; and 473 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 2.7% at a 95% confidence level. The margin of error for salaried employees in Canada is plus or minus 5.9%, and is 4.9% for Europe.

## THE INDIE REPORT

### THIS IS THE SECOND YEAR OF

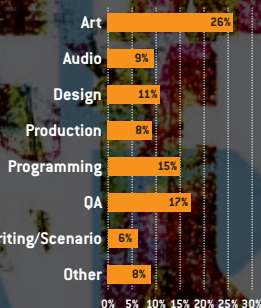
our indie section, which includes independent developers and contractors. Of those segments surveyed, it was independent contractors (not part of a team) who again find themselves at the top of the pile.

Last year's average compensation of \$45,137 was bested in 2010 by a significant margin, with independent contractors earning an average of \$55,493. Those who were members of a team also fared better in 2010, with an increase of over \$6,000 for an average of \$26,780. Individual developers were again at the bottom, earning less in 2010 with \$11,379.

Of those surveyed, the majority of respondents (52 percent) were designers, while the majority of independent contractors (26 percent) were involved in art. Of those individual developers or members of an indie team, 55 percent made under \$500 from the sale of their games in 2010.

Indie developers make money from sources other than their game, however. Eighteen percent of

### CONTRACTORS BY JOB FUNCTION



individual or team members made additional income from alternative game-related revenue streams. Of those, 16 percent made less than \$100, while 23 percent made over \$20,000. This additional revenue came in the form of promotions, non-game DLC content, sponsorships, ads, awards, and grants. Of those salaried and independent contractors who responded, 33 percent received an annual bonus, 7 percent royalties, and 10 percent profit sharing, of which 25 percent made under \$1,000 while 3 percent made over \$100,000.

Interestingly, of almost 500 non-salaried respondents, 63 percent have never worked at a traditional, salary-based game developer.

### JOB FUNCTIONS

For contractors, we asked respondents to choose the capacity in which they primarily worked in 2010, but for indies, it's a little more complex. Given the "many hats" nature of small-scale development, asking an indie to choose just one discipline is unreasonable. As such, the indie chart should be read as "what percentage of indies do at least this job function," rather than "how many indies do this job exclusively."

### INDIES BY JOB FUNCTION

Art	41%
Audio	18%
Design	52%
Production	37%
Programming	40%
QA	31%



### AN EXTENDED VERSION OF THE 10TH 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 UBM's TechWeb Game Group. Visit [www.gdmag.com/research](http://www.gdmag.com/research) for more information. This detailed report, *The Game Developer Salary Report: 2004—2010*, will be available in April.



# SALARY SURVEY

We leave a space at the end of our survey in order for developers to let us know what they think about the state of the industry, especially as regards jobs, in their own words. We have included a few of the notable anonymous responses from those who allowed their comments to be shared.

## THE BAD

“I’m getting frustrated working at large studios that are located in areas where the cost of living is too high to live within less than an hour of work. Commuting time, high rent, and crunch make it hard to focus on making a good game. My company president had the audacity to remind the dev team we’re not in this industry for a new Benz. I wish I would have responded that I am in it to hopefully provide the best for my new born son while doing what I personally love to do.”

“I got laid off twice, moved from California to Florida and worked for a total of three months. Not the best year for game developers.”

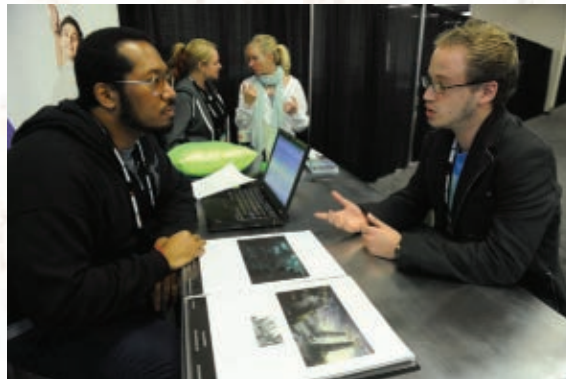
“It’s a scary year. Big publishers are trimming good talent instead of just trimming the fat. The new status quo seems to be rolling over temporary full-time contractors instead of hiring regular full-time employees. There is very little that looks attractive about the games industry from an employee’s point of view.”

“Quality of life is still a huge issue spanning the breadth and width of the industry. I’ve worked in both triple-A and social, and nowhere have I truly felt like ‘my time’ was respected. Instead, I was constantly barked at to spend more hours working, more hours producing, and more hours away from my family and friends.”

“The game industry is shifting, becoming polarized. Development seems to be shifting away from the middle as companies either invest in

cheap, fast mobile games or large, expensive blockbusters. This is creating a greater divide as entry-level positions at smaller companies aren’t preparing developers for the paradigms of larger studios.”

“The industry is poorly thought out and only benefits the publishers. Constant layoffs and no job security mean the talent moves on into other industries. How is anyone working in the industry supposed to have a life or a family if they are constantly going from contract to contract in between layoffs? You can’t buy a



home/apartment for fear you have to move to another city in order to find work. If game companies actually planned the productions out better, there would be no layoffs—just a transition into the next project.”

“I’ve turned the corner and realized that employers treat their employees like garbage (and this was before being laid off). You are an expendable asset in terms of your personal life and health. If most game industry jobs were properly advertised as an hourly wage job, there would be no way a self-respecting software engineer

would choose the game industry as their career path.”

“2010 was definitely the toughest of the last 4 years for my studio. We crunched at least 30 weeks at 12–16 hours per day 5–7 days a week, while our overall compensation decreased 20% from 2009. We also did not receive any merit increases or raises in 2010. Our company also had layoffs in 2010.”

## THE GOOD

“2010 seemed to be the best year yet for solo/small-team

“Despite overhanging economic gloom (and thanks in no small part to the arrival of new markets/models/platforms), the video game industry continues to provide abundant opportunities for success to passionate and dedicated developers.”

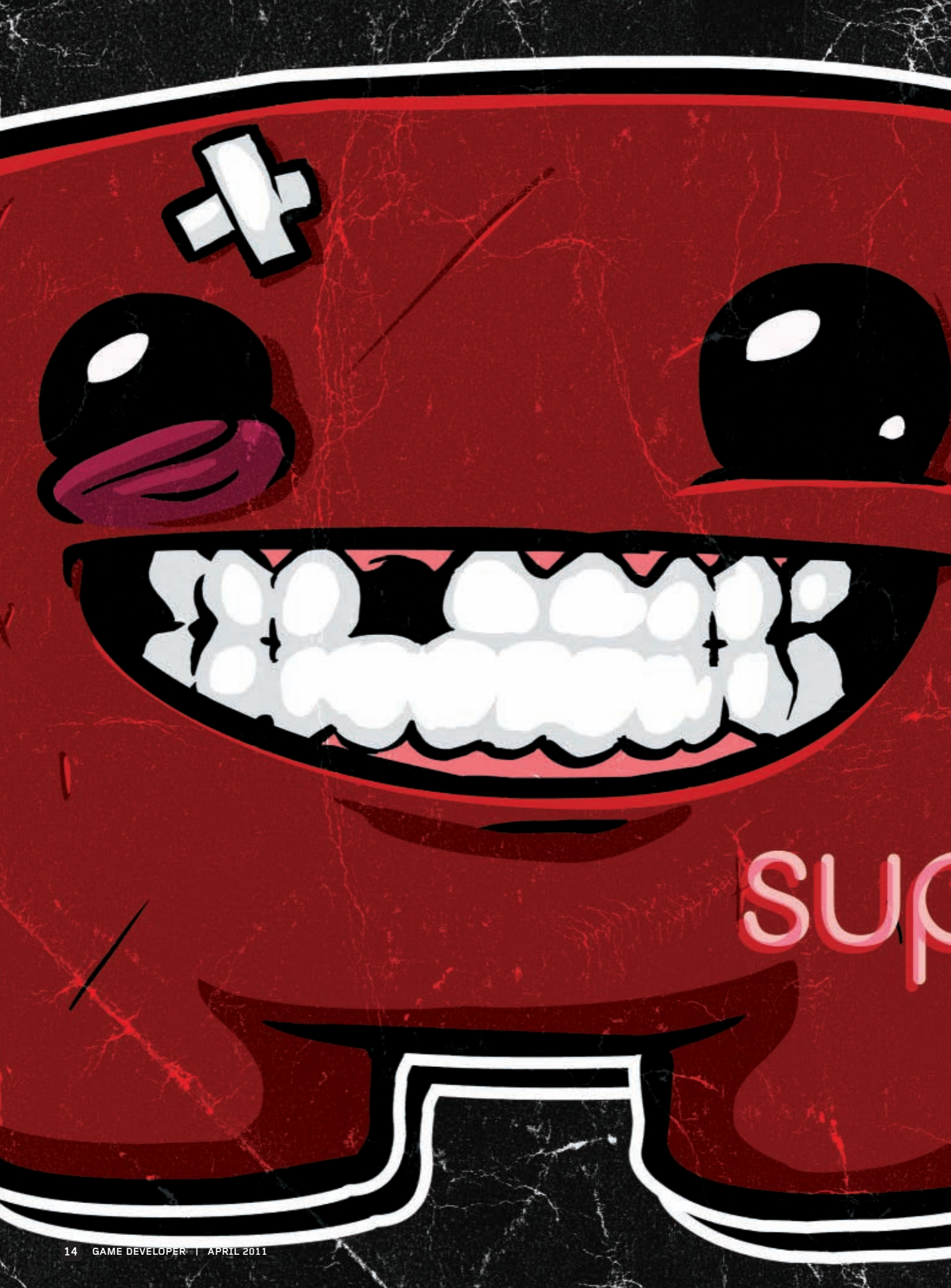
“I feel immensely grateful to all of the brilliant and interesting people I’ve worked with and who enable me to continue to make a living making games. The game industry still feels like a fertile, innovative place to be in 2010–2011 and I wouldn’t want to be doing anything else.”

“With mobile gaming exploding and new devices like the 3DS and the NGP entering the market, 2011 will be an exciting year. Exceedingly high-quality indie games will complement the efforts by big developers who have had ample time to work with the consoles and learn the best way to utilize the Xbox 360, PlayStation 3, and Wii.”

“2010 was the year of the indie. There are now more opportunities for indie developers than ever before. Working for big gaming is no longer the ultimate career. There is no reason a motivated person cannot make a good living developing games on their own.”

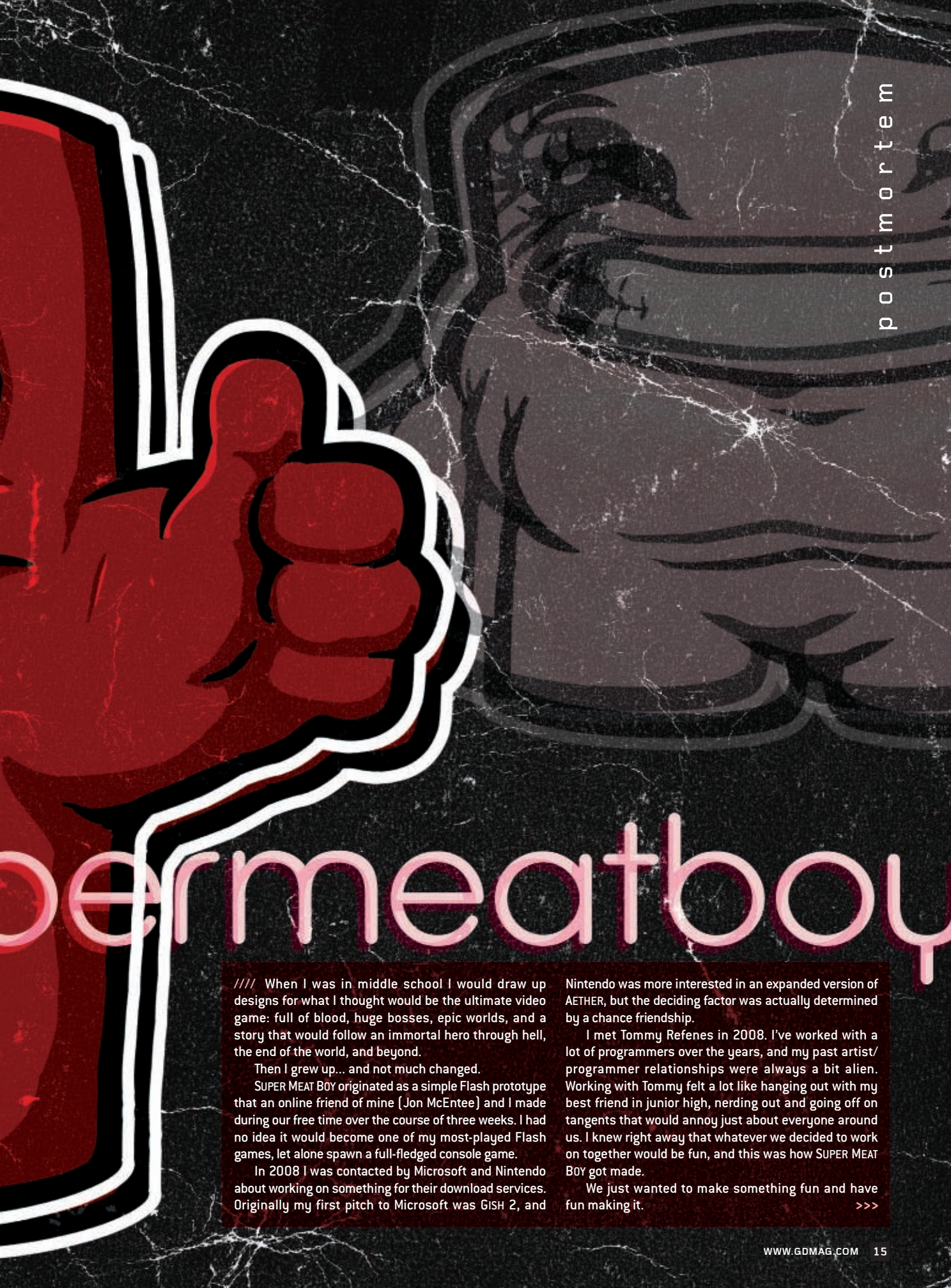
“It’s as rewarding a job as it gets if you are a creative type of person. Not every idea makes it into even your own games, but the ones that do, when they succeed, are an amazing reward! There is just nothing like it anywhere else!”





sup





# supermeatboy

//// When I was in middle school I would draw up designs for what I thought would be the ultimate video game: full of blood, huge bosses, epic worlds, and a story that would follow an immortal hero through hell, the end of the world, and beyond.

Then I grew up... and not much changed.

SUPER MEAT BOY originated as a simple Flash prototype that an online friend of mine (Jon McEntee) and I made during our free time over the course of three weeks. I had no idea it would become one of my most-played Flash games, let alone spawn a full-fledged console game.

In 2008 I was contacted by Microsoft and Nintendo about working on something for their download services. Originally my first pitch to Microsoft was GISH 2, and

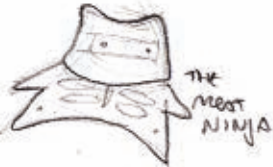
Nintendo was more interested in an expanded version of AETHER, but the deciding factor was actually determined by a chance friendship.

I met Tommy Refenes in 2008. I've worked with a lot of programmers over the years, and my past artist/programmer relationships were always a bit alien. Working with Tommy felt a lot like hanging out with my best friend in junior high, nerding out and going off on tangents that would annoy just about everyone around us. I knew right away that whatever we decided to work on together would be fun, and this was how SUPER MEAT BOY got made.

We just wanted to make something fun and have fun making it. >>>



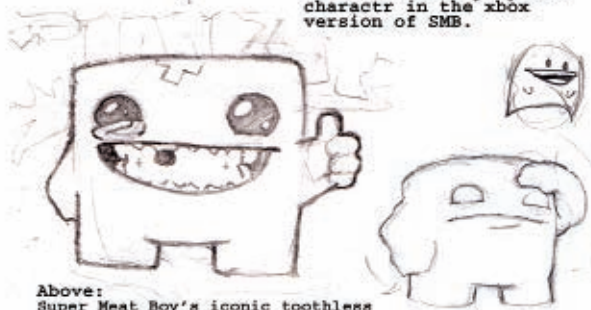
# MEAT BOY



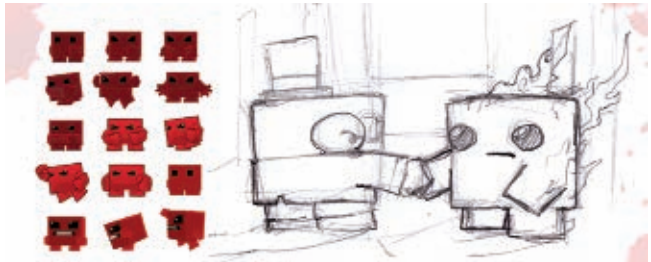
Originally Meat Boy's name was The Meat Ninja, or Insideout Ninja.

I removed his ninja hood because I felt the game already had too many similarities to the indie title N+.

The Ninja from N+ would later become a playable character in the xbox version of SMB.



Above: Super Meat Boy's iconic toothless face was inspired by the toothless grin of the MAD magazine mascot Alfred E. Newman.



Meat Boy's character design went through quite a lot of revisions when he made his way from the web to console, most were quite horrible...



Above: Example of what would happen if SMB had Sequels.

Left: Sketch of the cover of issue 2 of the SMB comic series. A play on Tales From The Crypt.

Getting this console deal was basically our one big break, our one shot to show everyone who we were and what we could do. No pressure.

## WHAT WENT RIGHT

### 1 // USING OUR OWN ENGINE AND TOOLSET

**Tommy:** When I tell most people that I made the engine and tools myself, they usually ask, "Why did you do that?" My friends over at FlashBang try to cram Unity down my throat every single time I talk to them, but I stand by the decision to make our own tools and engine.

One huge reason is control. I'm sort of a control freak when it comes to code, I like to understand everything that's going on in my codebase. That way, if something breaks, I know exactly where and how to fix it. Also, I got into games to program games, not to script them. I enjoy all aspects of game programming, from the engine to the gameplay. Since we're indie and can do what we want, and since I had the skillset, I simply enjoyed doing the engine.

Development of SUPER MEAT BOY took 18 months from the first line of engine code I wrote to the last line of error messaging code I wrote before final submission to XBLA certification. Personally, I think that's record time for a game made by two guys with as much content as it has. I honestly feel the reason we were able to do this is because I was so involved with the code. When a bug would pop up, I could track it down immediately no matter how low to the hardware it was.

There weren't many tools used with SUPER MEAT BOY. The in game level editor was invaluable because it provided Edmund the ability to make levels with a "what you see is what you get" mindset.

The only other tool we had was the Flash Exporter I made. Basically it was a script that packed all the flash symbols into one texture and exported animation information with sound cues. This paid for itself with the very first export of MEAT BOY that Ed did. We had sounds, animations, and everything with one quick export that the engine could easily manipulate and call when needed.

### 2 // THE DESIGN ENVIRONMENT

**Edmund:** Very early on, both Tommy and myself became a bit frustrated by the very rigid work environment most developers told us we needed to have in order to be taken seriously and get things done.

I remember the day we got an email from Nintendo asking for head shots and a developer bio. It suddenly seemed so insane how serious everyone takes an industry whose goal is supposed to be entertainment.

Tommy and I went out that day in search of the most ridiculous sweater vests we could find, broke into Sears Photos and used their setup to take what would become our team headshots [see Pg. 18]. I believe we also submitted some totally ridiculous dev bio to Nintendo that was printed in their press release alongside our photo.

The point I'm trying to make is that everything about our design environment was fun. It was important for us to always enjoy what we were doing, and let the love of our work come through





in interviews, videos, conventions, and even the game's design.

Tommy and I bonded over the course of development, and SUPER MEAT BOY was an expression of that. We had fun making this game and didn't hold those feelings back when it came to the decisions we made. SUPER MEAT BOY was a schoolyard inside joke that just got out of hand. I think one of the things that is most appealing about SMB is anyone who plays video games gets to be in on that joke.

### 3 // BACK TO BASICS DESIGN INNOVATION

**Edmund:** When Tommy and I talked about attempting to remake the MARIO formula, we didn't really discuss it publicly. Nothing could ever touch MARIO, and nothing has ever come close, but as a designer I desperately wanted to at least try.

SUPER MEAT BOY is SUPER MARIO BROS. if Tommy and I made it. If we had made a design doc, it would have been as simple as that.

So looking at it from that perspective, we had a very solid foundation design-wise, but video

games have changed a lot in the past 20 years. Difficulty has kind of been thrown out the door and replaced with accessibility over all else, erasing any real challenge.

It was vital for us to bring back the difficulty of the retro age, but also reinvent the idea of what difficulty meant. Frustration was the biggest part of retro difficulty and something we felt needed to be removed at all costs in order to give the player a sense of accomplishment without discouraging them to the point of quitting.

At its core, this idea was quite basic: Remove lives, reduce respawn time, keep the levels short and keep the goal always in sight. On top of these refinements, we added constant positive feedback, and even death became something to enjoy when you knew that upon completing the level you would be rewarded with an epic showing of all your past deaths. The replay feature was a way to remind the player that they were getting better through their own actions and reinforce that feeling of accomplishment of doing something difficult and succeeding.

### 4 // SOUNDTRACK

**Edmund:** Danny Baranowsky is an amazing musician, but one of the reasons why I believe his music was received so well in SMB lies in how things worked behind the scenes.

From the start, I felt it was important that Danny own the rights to all the music he made for the game. It seemed logical that an artist would put more into his work if he felt it was his and it represented himself. We wanted Danny to receive 100 percent of the profits from his work, and it only made sense that he would be that much more personally invested in his work if this were the case.

Danny's work comes from the kind of person he is. It's manic, obsessive, complex, and full of life. These were all elements we wanted for the SMB soundtrack, and making that happen was as easy as allowing Danny to make music he was proud of with little direction.

The SMB soundtrack was an amazing addition to the game—it gets your heart rate up, complements every aspect of its gameplay, and stays with you for days. I believe the reason for this



was respecting and trusting Danny as an artist and simply letting him do what he does so well.

## 5 // STEAM

**Tommy:** Steam is amazing. I can't stress that enough. The ability to quickly update within hours of a bug popping up made the entire PC launch much easier than it could have been if Steam had a different system in place to update code.

Also, Steam listens to its developers. They listened to us when it came to our suggestions for how we should push the sale, and in return we listened to them. Working with Steam never felt like a publisher / developer relationship. It felt like a mutual partnership to make the most money and put the best game out there.

We love Steam.

## WHAT WENT WRONG

### 1 // PERSONAL EXPENSES

**Edmund:** It's hard to say our personal expenses were something that really went wrong, due to the fact that it was a HUGE motivator to getting the game done, but it was definitely an issue as we moved into the last few months of development.

There was one point where I had emergency gallbladder surgery that put me in the hole \$50k due to the fact that I couldn't afford health insurance.

We had no real money at all, and even all the comics we had printed for GDC and PAX were attained through a barter system where my wife would make plush toys to sell in the NewGrounds store in exchange for the cost of printing.

Our situation was quite dire at several key points of development, but I've been on the poverty line for the past 10 years, so going without wasn't much of an issue, and honestly, we had much bigger issues to worry about anyway.

**Tommy:** At one point I had negative \$800 in the bank. It's bad when you go to a 7-Eleven to buy a Coke Zero and get rejected. Turns out, each one of those Coke Zeros cost me about \$40.

### 2 // LOSING SIGHT OF WIIWARE

**Tommy:** When we initially announced SUPER MEAT BOY for WiiWare, we were planning 100 levels at maximum, no cutscenes, and no unlockable characters. We were planning on just doing a straight port of the Flash game with a few extras and nothing more. We obviously got carried away, but I wouldn't call it a bad thing because we made the game we wanted to make. The bad part is we couldn't possibly do the game on the Wii.

As we were building the game and kept adding more to it, it became clear that it would be nearly impossible to fit within the size limits of WiiWare. It was always in the back of my mind to try to make sure we could, but cutting down to 50MB meant removing a lot of



Tommy Refenes (L)  
Edmund McMillen (R)

content that made the game what it is.

**Edmund:** Not releasing on the Wii still bothers me, and I wish we could have done it. After WiiWare became an impossibility, we looked into getting SMB published on Wii retail, but sadly, there wasn't one publisher we talked to that saw the Wii as a smart investment at this point in its life cycle. So we closed the book on the Wii.

### 3 // PC LAUNCH

**Tommy:** A two man team putting out a game on several platforms is pretty tough. The PC launch was a little rocky because of testing. I had what I felt was a wide range of test machines. I had range from our minimum specs (an Acer netbook) to a beefy quadcore. I thought I had everything covered; I had ATI cards and NVidia Cards. This obviously wasn't enough.

The day of PC launch we were inundated with tons of bugs, crashes on startup and shut down, and more. I think I answered about 2,000 emails during the first few days of launch. I felt similar to how I did during the crunch for the XBLA launch—every time I would fix something, it seemed like something else broke.

It was hard to go from the stress of XBLA launch to the PC launch in the same month. It was a feeling of accomplishment followed by an immediate feeling of failure. For our next game we'll do more extensive PC testing, and probably actually farm it out to a company that specializes in testing.

### 4 // LAST TWO MONTHS OF CRUNCH FOR XBLA LAUNCH

**Edmund:** In late August 2010, we got a phone call from our producer at Microsoft, explaining that there was going to be a fall promotion similar to Summer of Arcade. At this point, we were about four months from being done, but in order to release during this promo, we needed to pass certification in two.

The deadline seemed a bit impossible. We were told if we didn't make it into the fall promo, we would have to push the game back until spring or attempt to launch the game ourselves without much support, and risk a sizable loss. Microsoft explained that all games in the promo would get an exclusive launch week, very high spotlight advertising, reviews by Major Nelson, and face time at PAX and other events. This promotion was going to be called Game Feast.



At this point, both of us were going into the red financially and felt like if we didn't get into this fall promotion, there was no hope for us. We couldn't push to spring, and releasing without Microsoft support seemed like suicide, so we went all in and attempted to do what would take any team four months, within two. These two months were easily the worst months of my life.

The pressure, workload, and overall stress of development was extremely overwhelming. In those two months, neither of us took a single day off of work, working 10–12 hours a day everyday. There was a point at the end of development where I was getting less than five hours of sleep for several weeks. I remember having a breakdown in September where I actually thought I was stuck in some nightmare where I was repeating the same day over and over.

**Tommy:** Because we were so time compressed, we were basically developing features during bug checking, which meant every single time I turned on the computer and checked the bug database, the work I did the night before was pretty much rendered irrelevant. I would work and fix 100 bugs in a night and get it down to 50, then wake up the next morning and have 200 bugs to fix. This lasted for weeks and weeks. I felt sick, angry, and totally stressed. My parents were bringing me dinner because I literally didn't leave the house for those two months. I remember just saying to myself over and over, "Don't die until the game is done," because it was a real concern of mine. I felt miserable, my blood sugar was all over the place, but I absolutely had to press on and crush the bugs as they came up. I don't know if it made me stronger or not ... all I know is that somehow I survived!

**Edmund:** I think both of us were trying to keep from the other just how bad things were getting to avoid stressing the other out any more than we already were.

I had many nights where I would tell my wife that I was done, that I didn't want to make the game anymore, that it wasn't worth it, and that I would gladly bow out and take the loss just to go back to my normal life. She would "talk me off the roof," I'd go to sleep, wake up five hours later, and repeat the same day again.

5 // XBLA LAUNCH

**Edmund:** Development was over, SUPER MEAT BOY had taken home a few awards at PAX, and the press was starting to focus their lights on us. Many websites and magazines said SUPER MEAT BOY was easily the hit of the Feast, and possibly the next big indie hit, but the business side of Microsoft wasn't convinced.

We were told our price was too high, our visuals too rough and simply not as eye catching and flashy as the other Game Feast games COMIC

JUMPER and HYDROPHOBIA. Our hearts sank when we were informed that we were projected to sell as much if not less than HYDROPHOBIA, which would be the second-highest grossing game of the Feast in their minds.

This projection became that much more soul crushing when HYDROPHOBIA launched and its overall leaderboard had less than 10k players in the first week. If Microsoft's projections were correct, we were fucked.

A week later, COMIC JUMPER launched with a similar public reaction but slightly better numbers— still very low for XBLA standards. The Game Feast seemed to be a huge bomb, and quite a few news sites were already writing it off as a failure.

SUPER MEAT BOY launched Oct. 20th alongside COSTUME QUEST. It was placed third on the spotlight for four days. We never received any of the promotional launch bonuses that the previous Game Feast games had gotten (exclusive launch week, #1 spotlight, and a review by Major Nelson) but were told if we performed well in terms of Metacritic score and sales, we would move up and be more heavily advertised.

By day three of our launch, we had already outperformed HYDROPHOBIA and COMIC JUMPER's launch weeks combined, our Metacritic was the second-highest rated XBLA game of all time, and the word of mouth was insane.

Our spotlight placement was gone by day five and never came back. We never got a review by Major Nelson nor did we get an explanation for why they launched SMB alongside COSTUME QUEST, or for why, even though we exceeded their expectations for sales and score, we weren't given the treatment we were promised, even while they continued to heavily promote other Game Feast titles like COMIC JUMPER.

In the end, we felt very confused and taken advantage of. To this day we are still unsure of why things went down the way they did. Was it that Microsoft simply wanted to detach itself from

the Game Feast? Was it that they didn't believe we would perform as well as we did? Or was it just horrible luck at the most competitive time of the year for the video game industry?

Either way, by far the biggest mistake we made during SMB's development was killing ourselves to get into a promotion we would gain basically nothing from.

M E A T Y B I T S

**Tommy:** It's hard to talk about any kind of conclusion ... we aren't done with it yet! We have the editor, portal, and Mac version to finish. It's hard because it already feels like we are finished, like we ran the race. But then someone asks, "Hey, do you wanna do a whole other race?" and we're like, "Yeah, sure, that sounds like it could be fun."

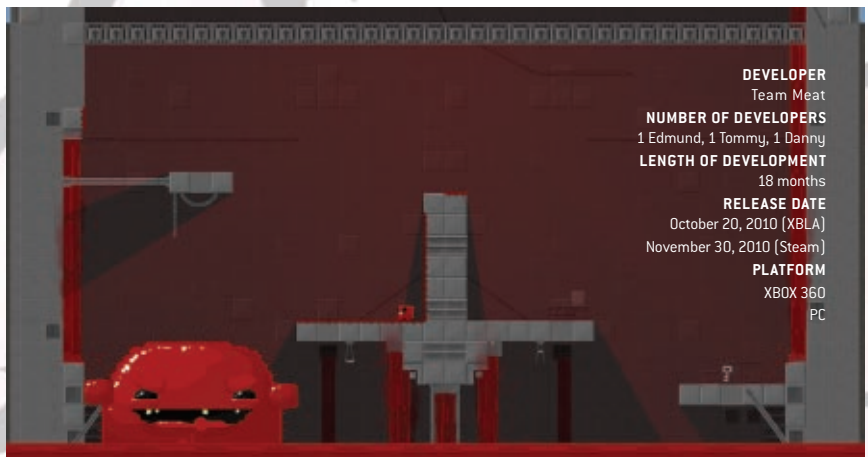
**Edmund:** Then you get there and you realize it's the same race, there's no prize at the end, and at this point you've lost control of your bowels.

Honestly, it was worth it to me because I got to make this game with a friend. It's as simple as that. If I had made it with someone I wasn't close to or couldn't joke around with, I would have had a miserable time and regretted the whole thing.

**Tommy:** I feel overall, that the game was worth all the stress. We went in as two guys with no games under our belts and left with the fourth-highest-rated PC game of 2010, sold over 400k copies worldwide, and received over 15 game of the year awards, which is a surreal thing to think about.

**Edmund:** It was an honor to make a game that we put so much of ourselves into, and that so many people appreciated. It's nice to be living proof that two college dropouts with no money can make a multiplatform console game and come out the other side with only minor head trauma. 🍌

EDMUND MCMILLEN draws stuff and designs things.  
TOMMY REFENES programs and macs on the ladies.



<b>DEVELOPER</b>	Team Meat
<b>NUMBER OF DEVELOPERS</b>	1 Edmund, 1 Tommy, 1 Danny
<b>LENGTH OF DEVELOPMENT</b>	18 months
<b>RELEASE DATE</b>	October 20, 2010 (XBLA) November 30, 2010 (Steam)
<b>PLATFORM</b>	XBOX 360 PC





WE ARE THE

igda

Developers helping developers

[www.igda.org/join](http://www.igda.org/join)



# creator of worlds

procedural terrain generation  
in a sandbox environment

J O S H U A T I P P E T T S

WITH THE RUNAWAY SUCCESS OF THE GAME MINECRAFT, there has been a bit of a resurgence of interest in the idea of procedurally generated worlds. The idea of a very large/infinite sandbox world is very appealing. Unfortunately, once you get beyond simple random numbers, procedural generation of a world isn't always the most approachable subject, and a lot of people don't know where to start. As with most things, the best approach is to start simple and build a complex model out of basic parts.

A theoretically infinite Minecraftian world is typically built up of easily managed chunks, and there is no practical limit on the size of the grid of chunks that can be generated on the X/Z plane. That is to say, the world is only maybe 128 layers deep but "infinitely" long and wide, limited by the precision of the machine's floating point types.

Doing a chunked approach like this enables you to build your world in pieces, and to only build the pieces you currently need to display or interact with in your game. Once generated, a chunk can be saved to a file to be loaded the next time that chunk is needed, rather than being generated from scratch. The world save file would dynamically grow as chunks are visited, taking up only as much disk space as needed to remember the currently visited world. To save disk space, you could save only the parts of a chunk that were modified. Then when loading, you would generate the level from the generator and apply the changes from the file to bring it up to date.

This article is concerned with the task of generating the initial geometry; First, a little groundwork.

>>>



# creator of worlds

## Implicit vs. Explicit Methods

An implicit procedural method is highly self-contained, expressible as a mathematical abstraction. You call a function with a set of coordinates, and you get a result in return. The value of the function at a given point or cell is not dependent upon or derived from any surrounding cells or points; it is self-contained. An explicit method, on the other hand, is typically implemented across large areas of the function at a time, and the value at a given point is usually highly dependent upon the values of surrounding points. It is often not possible to simply evaluate one point of the function; instead, an entire neighborhood must be evaluated.

An example of an explicit method would be using the Diamond Squares or Midpoint Displacement algorithms to generate a fractal heightmap using an allocated array. A large buffer of data is allocated, then iterated a number of times to generate the features for that particular area. By nature, these algorithms can only generate a chunk of data, and cannot generate a single point by itself. The size of the chunk produced also directly impacts the overall nature of the function.

An example of an implicit method would be using a Perlin noise function to generate a heightmap. The values of the heightmap are drawn directly from a "pure" mathematical process, rather than a process of iteration and filtration performed on a large array. There is no need to store large blocks of data. You can simply call the function with any possible coordinate point and obtain the value of the function at that point.

While on the face of it these techniques frequently produce similar results, their macro behavior is completely different. For one thing, with an explicit terrain generation method it can sometimes be difficult to ensure continuity across the borders between blocks of data.

An explicit method takes into account other points in the neighborhood of a point as long as those points exist within the chunk. It does not take into account points outside the chunk. Thus, it is possible for discontinuities or regularities to develop if we are generating a vast world, since it is not possible to generate the entire world explicitly all at once; at least, not without some highly expensive calculations and large-scale use of disk space. By subdividing the world into chunks, we are creating discrete pieces of world that conceptually have no knowledge about their neighbors, and possibly do not relate or correlate to them in any fashion. It is necessary to ensure that chunks will align with one another in a meaningful and cohesive manner, and sometimes this can be difficult to achieve using explicit methods, oftentimes requiring intricate hacks, kludges, workarounds, and storage of unnecessary states.

In contrast, all of the form and feature of an implicit method is inherent to the inner workings and nature of the function(s) upon which the method is founded—intimate knowledge of neighboring chunks is not necessary in order for a chunk to build itself. In most cases, we can simply store a simple random seed for our world and, as long as we do not change the underlying generator, this seed can be used to fully reconstruct the world, or any segment thereof.

While both types of algorithms are typically used in a generation scheme of any complexity, a large portion of the work can be done using implicit methods in a highly compact fashion. If we hold to our goal of preferring implicit methods over explicit, the end result of our efforts, ideally, will be a comprehensive set of functions governing every single cell in the world. We'll have functions that can tell us if a cell is stone, sand, dirt, is steeply inclined or flat, and so forth. The domain of these functions will be limited only by the precision of the underlying floating point format. By using double-precision floats we can achieve a domain so large as to be practically infinite in scope.

## Functions

Functions are the fundamental building blocks of our world. They come in a wide variety of shapes and sizes, and we'll be using Perlin noise fractal functions extensively, of course, as integral parts of the process. We can also use functions to generate directional gradients, create sharp edges or discontinuities, generate repeating patterns, and more. All these are potential tools in our toolbox for the building of large and diverse worlds.

Mathematically, of course, a function is an abstract entity or process that associates an input with some output. The same input will always produce the same output; in this manner, a function is deterministic. In our case, all our functions will be of the three-dimensional variety, accepting input of an (X,Y,Z) coordinate location representing a single cell in the world.

A function can be made as a composite of a number of other functions. They can operate on the output of another function or set of functions, or they can transform the input to another function or set of functions in some fashion. In this way, complex functions (equating to complex worlds) can be built up, a piece at a time, from simpler building blocks.

Functions that are classed as generators do not take inputs. These include gradient generators, fractal octave basis functions (value noise, gradient noise, simplex noise, white noise), and so forth. They may have parameters that alter their behavior, but these are typically just scalar values and not inputs obtained from other functional modules.

Functions that are classed as combiners, modifiers, or transformers accept arbitrary numbers of inputs, specified using a `setSource()` type of function convention. Some functions accept only a single source. Examples of these include `Invert` (multiplies the source by -1), `Bias` (modifies the output using a bias function), and `MapToCurve` (maps the source output to a user-specified spline curve). Other types accept a specified number of inputs greater than 1. Examples of these include `Combiner` (which can Add, Subtract, Multiply, Max, Min, or Average a set of inputs) and `Turbulence` (which distorts the domain of a function input based on up to three other functions, one per axis with each being optional).

Rather than build a function tree out of a sequence of actual function calls and code, I will adopt a notational scheme that demonstrates how the module types are chained. Here is an example of such a notation, expressed as a Lua table:

```
{name="Fractal1", type="fractal", fractal_type=RIDGEDMULTI, fractal_basis=GRADIENT, fractal_interpolation=QUINTIC, num_octaves=8, frequency=2}
```

The above definition declares a fractal function of type Ridged Multifractal, using Gradient noise as a basis and quintic interpolation for smoothing, specified in 8 octaves with a frequency of 2. Another example:

```
{name="Turbulence1", type="turbulence", main_source="Fractal1", x_axis_source="Fractal2", x_power=0.5}
```

This table sets up a turbulence modifier that acts upon our Fractal1 and uses a second fractal, Fractal2, as the noise source for the X axis turbulence.

Expressing an entire module tree as a sequence of Lua tables allows one to build the tree using concise notation, and to then feed the sequence to a parsing function that actually builds the module tree.

## The Basic Terrain

The world is going to be fundamentally split into two basic types of area: Solid and Open. Open, of

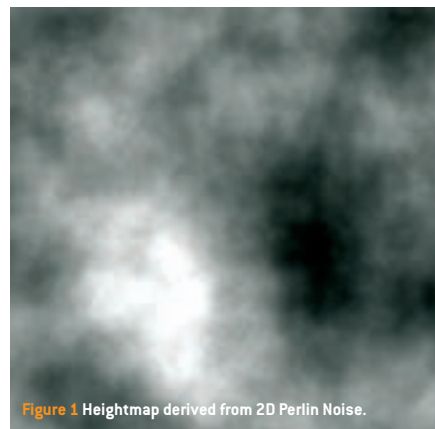
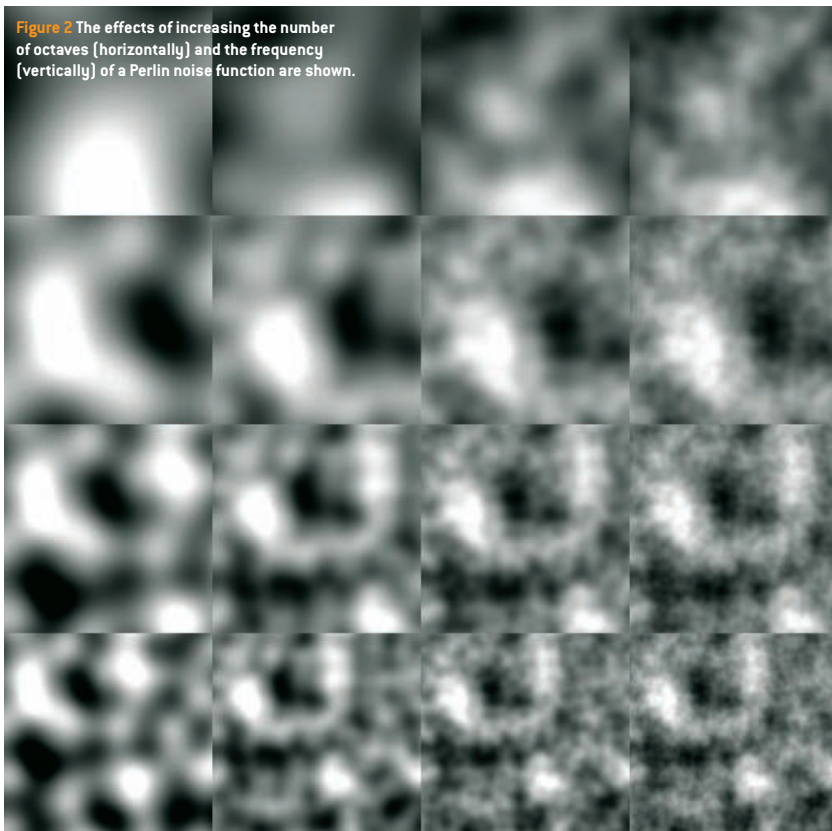


Figure 1 Heightmap derived from 2D Perlin Noise.



**Figure 2** The effects of increasing the number of octaves (horizontally) and the frequency (vertically) of a Perlin noise function are shown.



course, is air, or empty space (or water, at a later stage), whether above ground or deep in a cave. Solid is anything of a solid nature: rock, dirt, sand, and the like. So a good first step is to build a function that will separate the land from the air.

The easy and obvious way to do this is to represent the ground terrain as a heightmap, derived directly from a 2D Perlin noise fractal (see Figure 1).

Heightmaps are great; in effect encode volume information (terrain) as a single value per location (height). Heightmaps have been used for a long time to represent terrain for a number of reasons, such as efficient storage space, rapid rendering of large areas with level of detail, and easy terrain texturing. However, if you look at games such as MINECRAFT, they are very volumetric in nature. They have cliffs, overhangs, caves, tunnels, and mines. You name it, it's in there. A traditional heightmap just doesn't quite cut it for a volumetric world. What is needed is a function that operates in 3D space to determine whether a given cell is solid or open, and even what type of solid cell it should be.

A good basis for dividing Solid from Open is a simple gradient function. This type of function will assign a smooth gradient of values from  $-1$  to  $1$  along an axis defined by two arbitrary endpoints. In this case, endpoints are chosen to align the gradient along the Y axis. Points at

$Y=1$  output  $1$  and points at  $Y=0$  output  $-1$ , as below.

```
{name="GroundGradient",
type="gradient", y1=0, y2=1}
```

We can couple this function with a threshold function that outputs  $-1$  for anything less than or equal to a threshold, and  $1$  for everything greater than  $0$ . The result is a function that makes Solid everything in the space where  $Y \leq \text{threshold}$ . We can use a Select module to act as this threshold function.

```
{name="Constant1", type="constant",
constant=1},
{name="Constant0", type="constant",
constant=0}, {name="ConstantNeg1",
type="constant", constant=-1},
{name="GroundGradient",
type="gradient", y1=0,
y2=1}, {name="GroundBase",
type="select", main_
source="GroundGradient",
low_source="ConstantNeg1", high_
source="Constant1", threshold=0.2,
falloff=0},
```

This particular bit of code works by creating some constant sources that output a given

constant regardless of the input. Then we create a selection function. A selection function will select value from either its low source or its high source, depending on the value output by its third source, in this case the gradient function, GroundGradient. If the value of the third (control) source is less than a specified threshold, the value of low source is output; otherwise, the value of high source is output. A second parameter, falloff, can be used to implement a smoothing zone around the threshold, to gradually ease from one function to the other. In our case, we want a sharp divide between ground and air, so set falloff to  $0$ .

If we visualize a chunk made from this function, setting any cell that is equal to  $-1$  to solid, we'll get a flat plane. It's a good representation of a flat stretch of ground, certainly, but it's definitely not very interesting. What we need to do is apply some more functions to add surface features. To create these, let's look at the technique commonly called "turbulence."

Turbulence, in the context of noise functions, is simply a method for transforming the inputs of a function based on the outputs of another set of functions. To begin with, we will transform the Y coordinate of the input of our baseline function to create some basic hills and valleys. To do so, we need another function to act as the turbulence source.

A good place to start with this might be a basic Perlin noise fractal, also known as an fBm (fractional Brownian motion) fractal. fBm is the type of fractal most commonly used for generating heightmaps, and in this case it is going to act in a very heightmap-ish manner, since we are going to use it to adjust the value of Y passed to the baseline function. The behavior of a fractal can be tweaked in a number of different ways. First, a fractal is composed of layers of noise functions of different frequencies summed together. We can change the number of layers using the `setNumOctaves()` function; with fewer octaves resulting in smoother, less jagged noise; the more octaves we add, the more detailed the noise becomes.

We can also modify the frequency of the function. A higher frequency means that the features of the function—the crests and troughs of the wave, if you will—are closer together. Lowering the frequency in effect spreads the function out. In Figure 2 you can see a composite of images showing the effect of increasing the number of octaves of an fBm fractal (shown here increasing horizontally) as well as increasing the frequency of the function.

The first input of a turbulence function is obtained from the output of the function to perturb, in this case the thresholded gradient function forming our ground plane. There is also a set of inputs representing each axis of the coordinate system, so we can have a separate function perturb each of the X, Y and



# creator of worlds

procedural  
generation  
in  
sandboxes

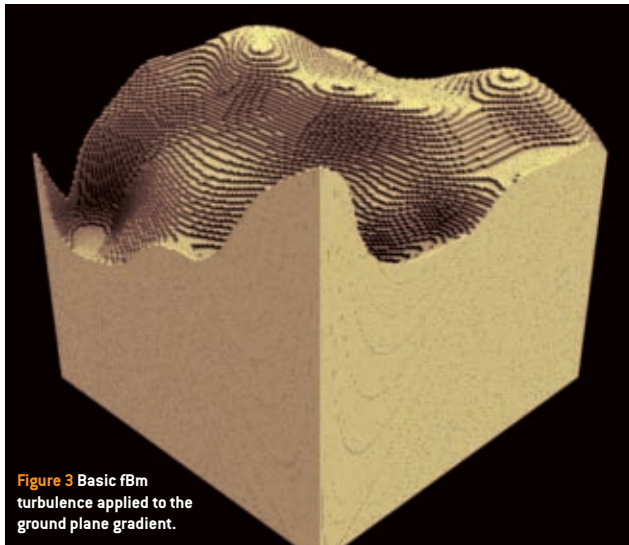


Figure 3 Basic fBm turbulence applied to the ground plane gradient.

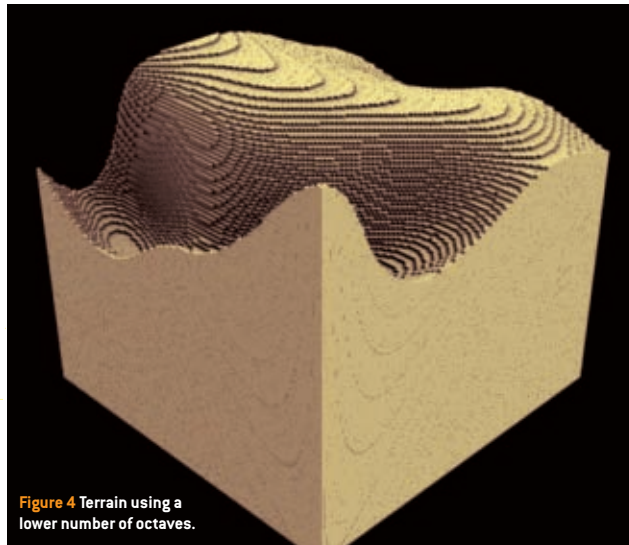


Figure 4 Terrain using a lower number of octaves.

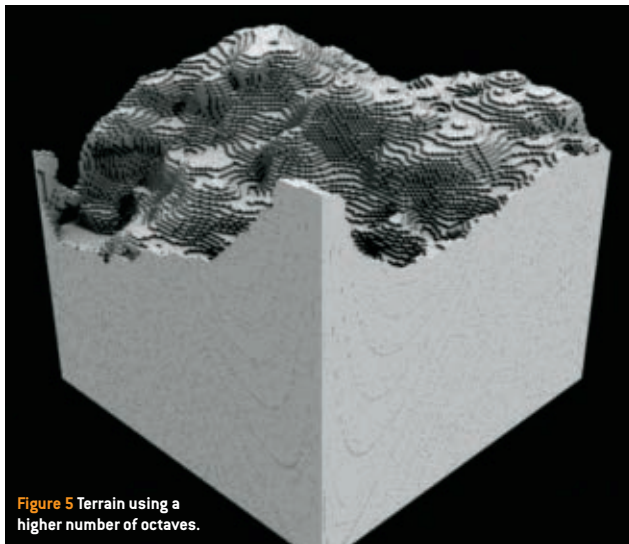


Figure 5 Terrain using a higher number of octaves.

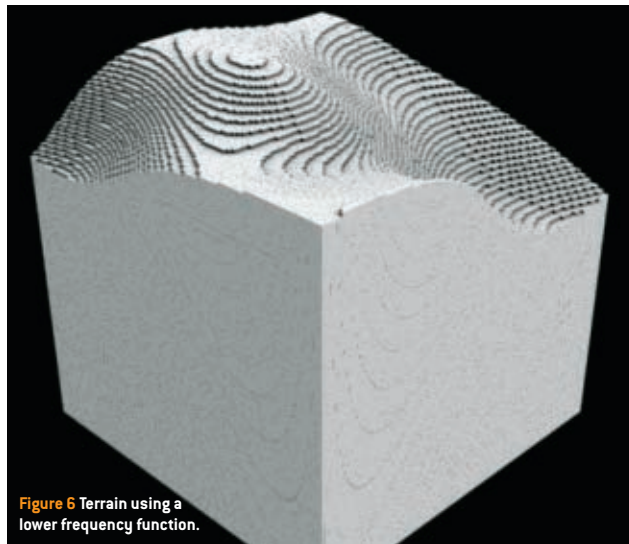


Figure 6 Terrain using a lower frequency function.

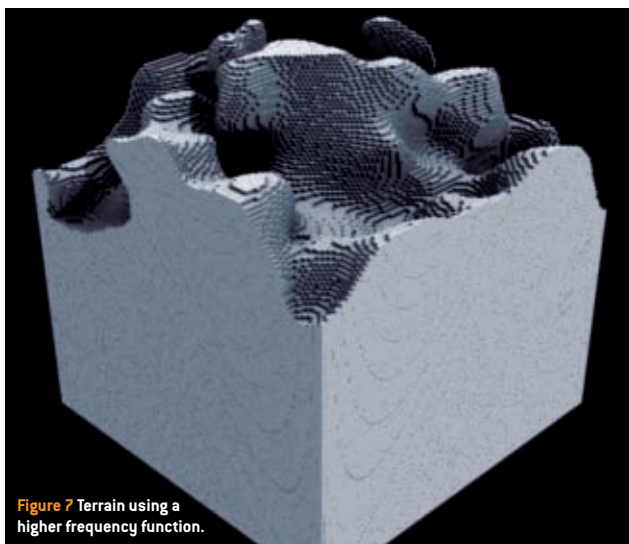


Figure 7 Terrain using a higher frequency function.

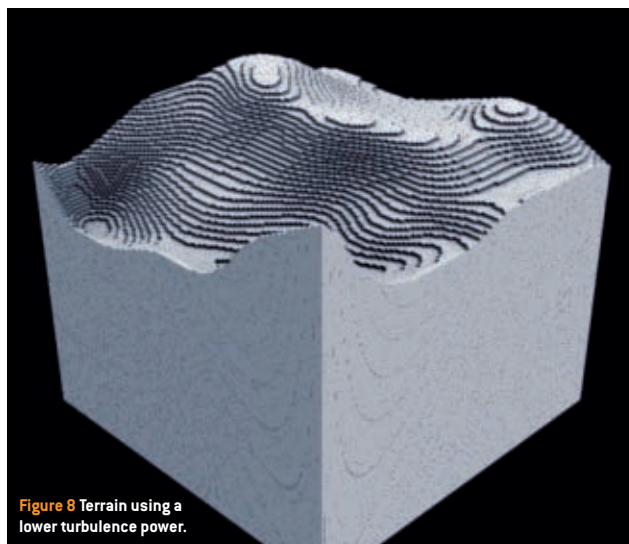


Figure 8 Terrain using a lower turbulence power.



Z coordinates. In this case, we are only perturbing Y, so we'll set the Y axis source to be our fBM fractal.

A turbulence function operates by taking the output values of each of the axis sources, and using those output values to modify the input coordinates to the main source function. The amount or magnitude of variation is specifiable by using the `setPower()` method of the turbulence function. The higher the value you set for power, the more the axis function will affect the corresponding coordinate value. Let's go ahead and set up our turbulence function, and set a few preliminary values for our number of octaves, frequency, and power of turbulence, and see what we get.

```
{name="GroundGradient", type="gradient", y1=0, y2=1},
{name="GroundShape", type="fractal", fractal_type="FBM", basis_type="GRADIENT",
interp_type="QUINTIC", num_octaves=2, frequency=1.75},
{name="GroundTurb", type="turbulence", main_source="GroundGradient", y_axis_
source="GroundShape", y_power=0.30},
{name="GroundBase", type="select", main_source="GroundTurb", low_
source="ConstantNeg1", high_source="Constant1", threshold=0.2, falloff=0},
```

We first set up our gradient as before, only this time, before we apply the select function to split the gradient range, we create a fractal, `GroundShape`, specifying a number of parameters to determine the characteristics of the fractal function. All these parameters, of course, are tweakable.

In the code above, we set the number of octaves to 2. This results in a rather smooth function. Adding more octaves contributes to a more chaotic, highly turbulent effect. We set the frequency to give a good sample of the character of the function; changing the frequency changes the distribution of features across the terrain. Finally, we set up the turbulence module, apply sources, and set the power to 0.5 on the Y axis. By adjusting the power of the Y turbulence, we adjust the effect the turbulence source fractal has upon the gradient basis function—a higher power results in a more highly turbulent ground surface. In Figure 3 you can see what kind of ground surface we get from this.

We're using a relatively low octave count to make the contours of our terrain smoother. We can decrease or increase the octave count and see how it affects the output by making the terrain less or more complex. In Figure 4, you can see the result of lowering the octave count. Contrast that with Figure 5, in which the octave count is increased.

We could also modify the frequency of the function and see how it tightens the features or spreads them out, as seen in Figure 6 [lower frequency] and Figure 7 [higher frequency].

You can see that the turbulence function is acting in a manner very similar to a heightmap, raising the terrain in some places and lowering it in others. In Figure 7, though, you can see that our approach is actually fundamentally different from a heightmap approach. In a typical heightmap, entire vertical columns of terrain are displaced, but in our approach, each individual cell or unit volume is individually displaced, allowing the formation of overhangs and other complex forms. You can see it in action further in Figures 8 and 9. Figure 8 shows the effect of lowering the turbulence strength, while Figure 9 shows the effects of increasing it.

The higher the power is, the more "frothy" the surface of the terrain becomes. Higher powers can create an extremely convoluted and alien landscape; which, depending on your scenario, may be exactly what you want. Turn up the turbulence power high enough, and you can end up with floating rocks and islands, as you can see in Figure 8. Of course, if having a "frothy" surface is not desirable, you can apply a domain transformation module that will clamp the Y coordinate to 0, giving the turbulence function the same value for all values of Y. This forces a 3D function to act as a 2D function, and will result in entire columns of terrain being displaced, just as with a traditional heightmap. And of course, this function could be combined with other fully 3D functions using any sequence of module chains, to allow for terrain as varied as you require.

Now, the beauty of composing functions out of combinations of other functions is that we can drastically alter the behavior of the system merely by changing a few parameters, or by swapping out one set of functions for another set. In this case, we can alter the character of the landscape by changing the basic fractal type to another variety, for instance a ridged multi-fractal:

```
{name="GroundShape", type="fractal", fractal_type=RIDGEDMULTI, basis_type=GRADIENT,
interp_type=QUINTIC, num_octaves=2, frequency=1.75}
```

In Figure 10 you can see how drastic the results are, just from this simple change.

## Caves and Tunnels

I've watched plenty of MINECRAFT videos of people out tooling around the countryside, riding pigs and chasing chickens, when all of a sudden the ground sort of opens up before them into a shadowed,

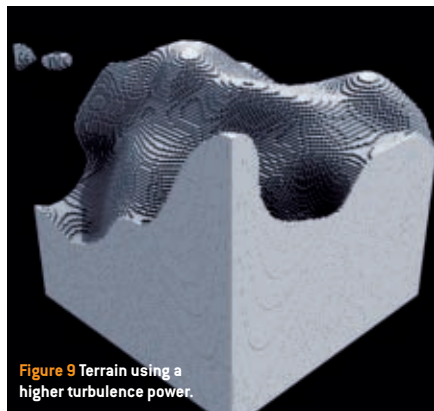


Figure 9 Terrain using a higher turbulence power.

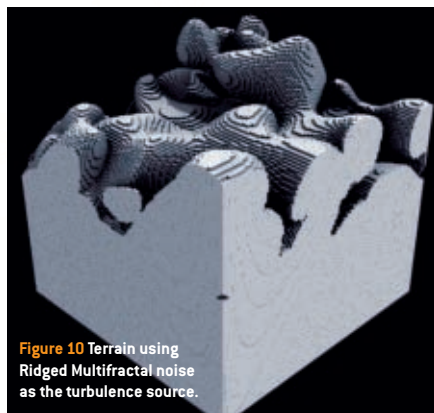


Figure 10 Terrain using Ridged Multifractal noise as the turbulence source.

mysterious tunnel twisting down into the depths. A rolling landscape covered in hills and trees is great; an enigmatic, dark cave to explore is the epitome of sheer awesomeness.

All mystery and excitement aside, a cave is pretty simple. It's just an open space. Typically, from what I've seen in MINECRAFT, the tunnels are relatively narrow and long, with few large caverns or openings. I'm not sure how MINECRAFT does it, but from where I sit, a low-octave Ridged Multifractal with some tweaks just might do the trick. However, it is going to take some massaging to get it to look right.

To begin with, a basic Ridged Multifractal in two dimensions with a single octave looks like Figure 11.

If we apply a threshold function to it, mapping the function to either solid or open, we get a series of contoured areas that sort of fit what we want. Figure 12 shows the results of varying the threshold parameter to obtain different cave configurations.

In 2D, this seems to work great, so let's take a look at Figure 13 to see what it's like in 3D. That's not really what we want. In three dimensions, the Ridged Multifractal doesn't carve lines or tunnels or tubes like you might expect. Rather, it carves a network of curved surfaces or shells. However, what we can do is set up another identical ridged noise source function, give it a different seed, and multiply the two sources together. This has the result of keeping the portions of the shells wherever they intersect, and discarding the rest of the areas.



# creator of worlds

CONTINUED FROM PAGE 25

```
{name="CaveShape1", type="fractal",
fractal_type="RIDGEDMULTI", basis_
type="GRADIENT", interp_type="QUINTIC",
num_octaves=1, frequency=2},
{name="CaveBase1", type="select",
main_source="CaveShape1",
low_source="Constant0", high_
source="Constant1", threshold=0.7,
falloff=0},
{name="CaveShape2", type="fractal",
fractal_type="RIDGEDMULTI", basis_
type="GRADIENT", interp_type="QUINTIC",
num_octaves=1, frequency=2, seed=1323},
{name="CaveBase2", type="select",
main_source="CaveShape2",
low_source="Constant0", high_
source="Constant1", threshold=0.7,
falloff=0},
{name="CaveMult", type="combiner",
combiner_type="MULTIPLY",
source_0="CaveBase1",
source_1="CaveBase2"},
```

Now, take a look at Figure 14 to see the results. That's more like it. We have plenty of interconnected narrow little tunnels to explore, as well as some areas where the caves open up a bit into larger caverns. We can play around with the various thresholds of the two cave sources in order to tweak the thickness of the caves. Note that the selection functions for the cave networks output values of 0 or 1, rather than -1 or 1. This is because we are using the cave network as a multiplicative source, used to "mask" off areas of the final base function. We want the base function to be open anywhere the cave function evaluates to 1, and solid where it evaluates to 0.

This gives us the interconnected system of tubes, but since we are using a 1-octave fractal for the basis, the caves seem sort of weirdly smooth. We can roughen them up by applying some turbulence.

```
{name="CaveTurbX", type="fractal", fractal_
type="FBM", basis_type="GRADIENT", interp_
type="QUINTIC", num_octaves=3, frequency=3,
seed=1001},
{name="CaveTurbY", type="fractal", fractal_
type="FBM", basis_type="GRADIENT", interp_
type="QUINTIC", num_octaves=3, frequency=3,
seed=1201},
{name="CaveTurbZ", type="fractal", fractal_
type="FBM", basis_type="GRADIENT", interp_
type="QUINTIC", num_octaves=3, frequency=3,
seed=1301},
{name="CaveTurb", type="turbulence",
main_source="CaveMult", x_axis_
source="CaveTurbX", y_axis_
source="CaveTurbY", z_axis_
source="CaveTurbZ", x_power=0.25,
y_power=0.25, z_power=0.25},
```

This time we are using three noise sources and a turbulence function to perturb our multiplied

caves network. Each axis source is set with a different seed. Rendering the output of this on our cave network gives us something similar to Figure 15. That gives a nice, chunky, natural look to the caves, eliminating the smooth curves and lines and making it rougher and more "cave-like."

To wrap up the process, we need to invert the cave function (since it currently acts as a solid function where the caves are; we need the caves to be the open space, and the surrounding function to be solid) and multiply it by our ground function to get the final open/solid function for our ground formation, as below.

```
{name="CaveInvert", type="scaleoffset", source="CaveTurb", scale=-1, offset=1},
{name="GroundCaveMult", type="combiner", combiner_type="MULTIPLY",
source_0="GroundBase", source_1="CaveInvert"},
```

Figure 16 shows the final result of multiplying our basic ground function by the inverted cave function. Now we have a nice, hilly chunk of ground laced with a network of caves and cracks, ripe for exploring.

As you can see, the full specification for the module is relatively simple: 18 different modules to get a complex terrain. Of course, we can easily modify the way we do things at any step of the way. In particular, the ground shape function should probably be tweaked a bit to provide more variety. If you look at a topographical map of a section of the Earth's landscape, you can see that the surface of the ground is not homogenous. There are flat areas, hilly areas, areas of mesas and tabletop mountains, deep canyons, steep mountains, and so forth. Our current implementation only uses one simple fBm fractal to perturb the surface. To get more varied results, we could replace the fBm fractal with a more complex tree using select functions, blend functions, and more in order to create a non-homogenous function that could produce wildly different terrain types, from flat to hilly to mountainous.

We can also tweak the cave generator to produce more varied and intricate caves. A possible tweak might be to add another fractal source that is scaled by the ground gradient, so that as it draws nearer the surface it approaches zero. By tweaking the frequency of this, and adding a thresholding function, then combining it with the cave network function using a combiner such as Add or Max, we can add larger voids and caverns near the bottom of the world that scale down and disappear nearer to the surface. We can fill these deep caverns with lava and demons to create dangerous, hellish depths to test the player's survival skills.

There's still more we could do. Tying the ground shape gradient function to a curve could give us layer types. The top three or four layers of the ground should be dirt, then stone meta-types all the way down to the final layer, which should be unbreakable bedrock to keep us from digging through to the Abyss. Further functions drawing off the ground gradient can define layers where mineral deposits may occur, using the gradient to scale the likelihood of finding rarer minerals deeper down.

Procedural generation of worlds is a complex undertaking, but the complexity can be easily handled by breaking the work down into smaller, more manageable tasks, and using simple mathematical constructs and functions to build it up, layer upon layer, until the final result is a complex, interesting, living and breathing world filled with variety and depth. ☺

JOSHUA TIPPETTS has been an indie game developer for almost fifteen years. He currently lives in the mountains of northern Wyoming. You can email him at [vertexnormal@linuxmail.org](mailto:vertexnormal@linuxmail.org).



Figure 12 Changing the threshold parameter results in larger or smaller cave systems.

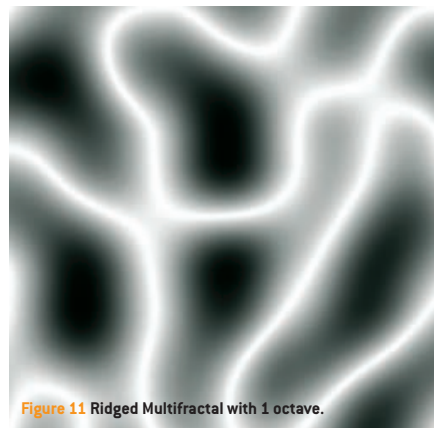


Figure 11 Ridged Multifractal with 1 octave.



**LISTING 1**

**The entire set of modules in Lua table form to generate the function from this article.**

```

minecraftlevel={
  {name="Constant1", type="constant", constant=1},
  {name="Constant0", type="constant", constant=0},
  {name="ConstantNeg1", type="constant", constant=-1},
  {name="GroundGradient", type="gradient", y1=0, y2=1},
  {name="GroundShape", type="fractal", fractal_type="FBM", basis_
type="GRADIENT", interp_type="QUINTIC", num_octaves=2, frequency=1.75},
  {name="GroundTurb", type="turbulence", main_source="GroundGradient",
y_axis_source="GroundShape", y_power=0.30},
  {name="GroundBase", type="select", main_source="GroundTurb",
low_source="ConstantNeg1", high_source="Constant1", threshold=0.2,
falloff=0},
  {name="CaveShape1", type="fractal", fractal_type="RIDGEDMULTI", basis_
type="GRADIENT", interp_type="QUINTIC", num_octaves=1, frequency=2},
  {name="CaveBase1", type="select", main_source="CaveShape1",
low_source="Constant0", high_source="Constant1", threshold=0.7,
falloff=0},
  {name="CaveShape2", type="fractal", fractal_type="RIDGEDMULTI", basis_
type="GRADIENT", interp_type="QUINTIC", num_octaves=1, frequency=2,
seed=1323},
  {name="CaveBase2", type="select", main_source="CaveShape2",
low_source="Constant0", high_source="Constant1", threshold=0.7,
falloff=0},
  {name="CaveMult", type="combiner", combiner_type="MULTIPLY",
source_0="CaveBase1", source_1="CaveBase2"},
  {name="CaveTurbX", type="fractal", fractal_type="FBM", basis_
type="GRADIENT", interp_type="QUINTIC", num_octaves=3, frequency=3,
seed=1001},
  {name="CaveTurbY", type="fractal", fractal_type="FBM", basis_
type="GRADIENT", interp_type="QUINTIC", num_octaves=3, frequency=3,
seed=1201},
  {name="CaveTurbZ", type="fractal", fractal_type="FBM", basis_
type="GRADIENT", interp_type="QUINTIC", num_octaves=3, frequency=3,
seed=1301},
  {name="CaveTurb", type="turbulence", main_source="CaveMult", x_
axis_source="CaveTurbX", y_axis_source="CaveTurbY", z_axis_
source="CaveTurbZ", x_power=0.25, y_power=0.25, z_power=0.25},
  {name="CaveInvert", type="scaleoffset", source="CaveTurb", scale=-1,
offset=1},

  {name="GroundCaveMult", type="combiner", combiner_type="MULTIPLY",
source_0="GroundBase", source_1="CaveInvert"} -- Map this function for
final output
}

```

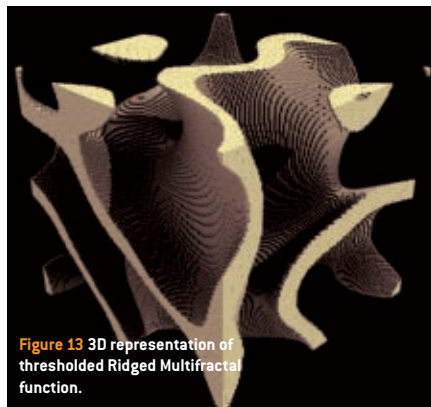


Figure 13 3D representation of thresholded Ridged Multifractal function.

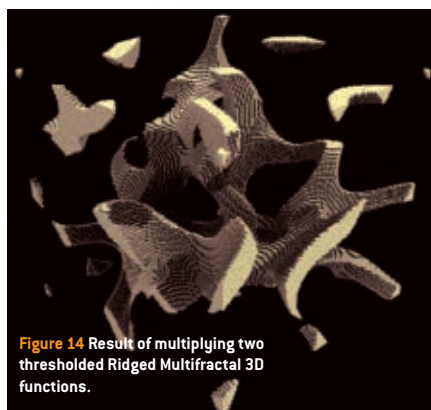


Figure 14 Result of multiplying two thresholded Ridged Multifractal 3D functions.

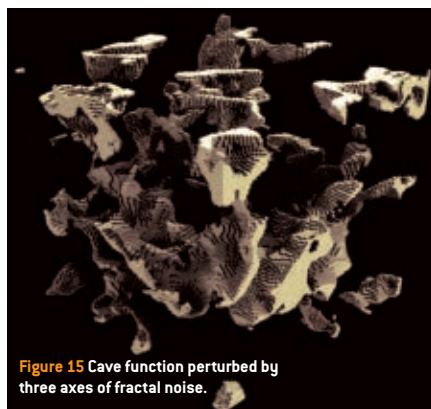


Figure 15 Cave function perturbed by three axes of fractal noise.

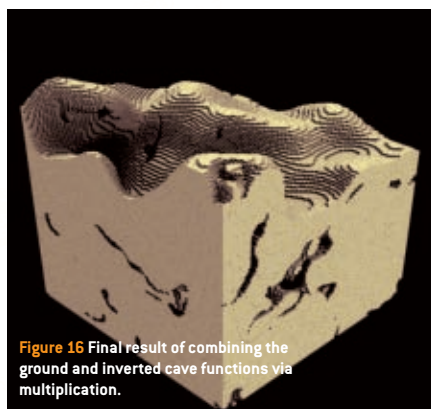


Figure 16 Final result of combining the ground and inverted cave functions via multiplication.





THE 13TH ANNUAL  
INDEPENDENT  
GAMES FESTIVAL

# AWARDS



## CONGRATULATIONS TO THE 13TH ANNUAL IGF AWARD WINNERS

### IGF MAIN COMPETITION

» **SEUMAS MCNALLY GRAND PRIZE**

Minecraft, by Mojang

» **EXCELLENCE IN VISUAL ART**

BIT.TRIP RUNNER, by Gaijin Games

» **EXCELLENCE IN AUDIO**

Amnesia: The Dark Descent, by  
Frictional Games

» **EXCELLENCE IN DESIGN**

Desktop Dungeons,  
by QCF Design

» **TECHNICAL EXCELLENCE**

Amnesia: The Dark Descent,  
by Frictional Games

» **BEST MOBILE GAME**

Helsing's Fire, Ratloop

» **AUDIENCE AWARD**

Minecraft, by Mojang

» **DIRECT2DRIVE VISION AWARD**

Amnesia: The Dark Descent,  
by Frictional Games

### IGF STUDENT SHOWCASE

» **BEST STUDENT GAME**

FRACT, by University of Montreal

### NUOVO AWARD

» Nidhogg, by Messhof



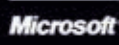
VIEW THIS YEAR'S FINALISTS AND WINNERS AT [WWW.IGF.COM](http://WWW.IGF.COM)

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

Platinum Sponsor



Gold Sponsor



Gold Sponsor



Hardware Platform  
Sponsor



Platform Sponsor



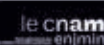
Distribution Partner



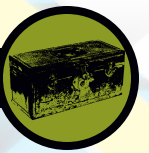
Student Showcase  
Platinum Sponsor



Student Showcase Gold Sponsors







REPORT FROM THE SHOW FLOOR

# GAME DEVELOPERS CONFERENCE 2011

AT THE MILESTONE 25TH GAME DEVELOPERS CONFERENCE, THE MAJOR THEMES SEEMED TO BE THE CONVERGENCE OF GAME PLATFORMS, AND THE INCREASING FOCUS ON SMALLER TEAMS AND MOBILE DEVICES. WE SPOKE WITH A NUMBER OF TOOLS AND TECHNOLOGY INNOVATORS IN THESE SPACES TO SEE HOW THEY WOULD ADDRESS THE RAPIDLY CHANGING INDUSTRY.

— Tom Curtis

**GAMESPY TECHNOLOGY**  
**GameSpy Industries**  
[www.poweredbygamespy.com](http://www.poweredbygamespy.com)

]]]] GameSpy showcased a number of new titles using the company's suite of online services, demonstrating what the tools are capable of across multiple platforms. Among the games on display was Trendy Entertainment's DUNGEON DEFENDERS, the first GameSpy-powered title for Android devices, and also the first to support cross-platform play with the PC, PlayStation 3, Xbox 360, and iOS. In addition to the platforms already supported by GameSpy Technology, the company revealed plans to expand its services to Sony's upcoming NGP hardware.

work with developers on smaller-scale titles, so to do that we completely changed our licensing model," explained GameSpy Technology vice president Todd Northcutt. Smaller developers will pay for the service based on their game's monthly active user count, and as games rise or decline in popularity, the fee for using GameSpy tools will change accordingly. "We don't want to turn anyone's game off—ever. There will always be the lower level and free tiers, so as your game sunsets, people can still play and the service will still operate. As a developer, you might not be seeing any new revenue from the game, so you shouldn't have to pay for the service," Northcutt added.

teams some extra office space in order to better understand indie development and to give indies some extra exposure and experience. "We've learned a lot from these indie teams," said Northcutt. "We understand how a team of 200 or so works, but it's totally different when you look at a three-to four-person team, where everybody does everything." Based on feedback from the teams in the Indie Open House, the company says it plans to further simplify its cloud storage and other services to better accommodate smaller projects.

"At Unity, Fridays are dedicated to working on whatever you want, as long as it pushes the envelope," Francis explained, allowing Unity's employees to find new and perhaps unexpected ways to make the Unity platform more robust.

**AUTODESK 2012 PRODUCT UPDATES,**  
**PROJECT SKYLINE**  
**Autodesk**  
[www.autodesk.com](http://www.autodesk.com)

]]]] Game tools firm Autodesk discussed a slew of its upcoming products at GDC, including 2012 versions of 3ds Max, Maya, Softimage, the updated versions of Autodesk Beast, HumanIK, and Kynapse, as well as the company's latest animation pipeline solution, Project Skyline. For the latest incarnation of its established software suite, Autodesk worked to improve interoperability between its products to ensure that developers could more easily transition from one set of tools to another. Specific changes include more homogenous and robust f-curve editors across 3ds Max, Maya, Softimage, and MotionBuilder, as well as a one-step solution that transfers projects between the various programs. The company's middleware products, which include HumanIK and Kynapse, will offer support for NGP, iOS, and Android, and will include a number of changes related to performance optimization.

The company also discussed its recent acquisition of 2D UI middleware provider Scaleform, explaining how the partnership will help Autodesk reach out to 2D game developers working on mobile platforms. "In the mobile space, you can kind of see where things are going. You have things like Unreal Engine running on iOS, but not all developers are going

**UNITY GAME DEVELOPMENT TOOL**  
**Unity Technologies**  
[www.unity3d.com](http://www.unity3d.com)

]]]] In the midst of GDC, Unity Technologies released the Unity Android add-on for its development platform, enabling developers to port Unity-developed projects to Android devices. Since Unity made a pre-release version of Unity for Android in 2010, developers have shipped nearly 50 games for the platform.

"Developers just want to get their game out there," said Unity COO Nicholas Francis. "We want developers to be able to make their game, and put it out where they like. Personally, I want developers to be able to switch platforms by just re-programming their game's controls and tweaking it a bit."

Francis noted that when integrating new features into the Unity engine, the company takes note of what developers want via an online voting system, and even dedicates one day a week at the Unity office to creative experimentation in order to find new ways to add features and improve the overall package.



GDC 2011

The company also announced its GameSpy Open initiative, which will make the company's services available to start-up and independent teams for free, at least until their games start making money. "For a long time we've been focused on AAA developers and publishers, but we've never really been able to

The company also showed off a number of titles from the indie teams that participated in its recent Indie Open House Program, where five teams worked from the GameSpy offices in the Bay Area to develop their games in a collaborative environment. Northcutt explained that the company decided to offer these





to transition to 3D development overnight. We thought that partnering with Scaleform would help create a transition from 2D development to 3D," said Marc Stevens, vice president of Autodesk Games. In order to help integrate the Scaleform middleware into the company's current lineup of products, Autodesk hopes to allow developers to view and augment Flash movies in Maya, for instance, so they can add 3D effects to games primarily developed in 2D. As the mobile space matures, Autodesk predicts that 3D development will become the norm, and the company's integration of the Scaleform tools will help incentivize teams to start developing in 3D. "Working

maneuver through game worlds via flight, with pathfinding behaviors to limit collisions and traffic jams. Havok AI has been around for about two years now, and our MMO clients seem to like it a whole lot, and now it has a number of features that are MMO-specific," said Havok VP of engineering Dave Gargan. In order to best suit MMO titles, Havok AI supports a large number of AI controlled characters, and allows for instancing as well as stitching and streaming, so games can stream in parts of the environment while incorporating AI pathfinding across the streamed sections.

The latest update to Havok cloth includes support for more realistic hair, using simulated layers rather than specific hair



in a 3D environment allows you to make changes to 2D content a bit easier, whereas it's harder to make those customizations in a 2D environment. The question is, will people have to do this to be competitive from an efficiency point of view, or will they have to do it to create certain effects and the like that can only be done in 3D?" Stevens posed.

#### HAVOK AI, CLOTH, BEHAVIOR TOOLS

### Havok

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

]]]] In the most recent update to Havok AI, the popular physics engine provider added a number of new features primarily intended to benefit MMO titles. Havok AI now supports navigation in space in addition to surface navigation, allowing AI-controlled characters to

strands. This means of rendering hair allows developers to create believable-looking characters without putting too much strain on modern hardware. The tool allows developers to change the number of hair layers to tweak fidelity and performance, and sliders control the length of these layers, making the tool suitable for titles that include detailed character customization.

Havok also showcased improvements to its behavior tool, which helps artists author runtime animations by helping characters transition between animations based on their position in a game environment. Havok's demo showed a character leaping from ledges and finding cover, transitioning between actions in various ways depending on the player's position in the game world.

Finally, Havok noted that it has expanded its platform support to iOS and Android, and is in the process of extending to upcoming platforms like Sony's NGP and Nintendo's 3DS.

#### UNREAL ENGINE 3

### Epic Games

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

]]]] In the latest incarnation of Unreal Engine 3, Epic Games added a number of new features to boost the visual fidelity of future titles that use the popular engine. The new additions include improved depth of field effects, image-based reflections on surfaces, and improved bloom. Several of the engine's newest features are available only on DirectX 11, such as sub-surface scattering for improved lighting and dynamic tessellation and displacement.

To demonstrate the engine's latest update, Epic showed a tech demo, dubbed "Samaritan," running on three off-the-shelf Nvidia GTX 580 graphics cards that they believe could feasibly run on just a single card, with enough optimization.

The tech demo also showed off improved cloth simulation using Nvidia's Apex framework, as well as deferred shading and skeleton-based motion blur, which can be used to affect very specific parts of a character, such as an arm in mid-punch.

Epic's Mark Rein also confirmed that Unreal Engine 3 is currently in development for Mac, though the company did not provide an estimate regarding its release on the platform.

#### PRIMESENSE GESTURE RECOGNITION

### PrimeSense

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

]]]] After working with Microsoft to make the Kinect motion sensor a reality, PrimeSense's goal at this year's GDC was to teach developers how to create intuitive interfaces for the company's latest depth cameras. In anticipation of upcoming Smart TVs that will support the PrimeSense hardware, the company demonstrated how to navigate simple television menus using gesture control. Much like

menu navigation using Kinect, users control an on-screen cursor with their hand as they glide over a number of on-screen buttons for movie playback, on-demand content, and more. While Kinect's interface requires users to hold their hand in place for several seconds to select an option, PrimeSense's demo interface tends toward users pushing forward to make their selection without having to wait.

Also at the booth were the winners of the PrimeSense Developer Challenge, a contest that tasked entrants with creating a functional web browser that uses only gesture-based control, with the winner earning a grand prize of \$20,000. The top entry, dubbed SwimBrowser, tracks both hands, allowing users to click links using a distinct diving motion. The browser was awarded first place because it provided the most fluid and natural-feeling interface. Other entries in the contest included a browser that displayed a virtual room with floating navigation buttons in the corner of the screen, and another interface that relied on its own version of sign language.

#### TRINIGY VISION ENGINE

### Trinigy

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

]]]] After expanding its business to include Europe, North America, and Korea in 2010, Trinigy has since turned its focus to expanding platform support for the company's Vision engine. Trinigy announced it is working to add support for Sony's NGP within the next few months, followed by support for iOS a few months later. "We decided to first add support for the NGP because the platform is interesting, and because it's technically challenging," said Trinigy CEO Felix Roeker, "Sony approached us at a very early stage, and we wanted to be one of the very first companies to get onto that bandwagon, so that's the reason we started with NGP. iOS and Android are slightly less technically challenging to program for than the NGP, and we estimated it would take less time to add support for them, so that's why we started with our mobile versions slightly later [than the NGP version]."



Trinigy also discussed several new features that have been added to the Vision engine, including a new shadowing system and particle system, as well as preparations to launch networking integration later this year. In addition, the Vision engine now includes a sample game with its SDK to help developers determine what they can accomplish with the latest version.

**INTEL SANDY BRIDGE PROCESSOR, SSDS, DEVELOPER TOOLS**  
**Intel**  
[www.intel.com](http://www.intel.com)

]]]] Intel's booth on the GDC show floor dedicated much of its space to showing off the company's Sandy Bridge processors, which integrate graphics performance directly into the CPU, allowing for high-performance gaming even without a dedicated GPU. Nearly all of Intel's demo machines ran on these new processors, showing games like PORTAL 2 and WORLD OF WARCRAFT running without a discreet video card.

Alongside the Sandy Bridge hardware, Intel highlighted its newest solid-state drives, emphasizing the advantages they offer for PC gaming, such as improved speeds for loading and rendering content and decreased texture pop-in.

As far as software tools, Intel showed off version 4.0 of its Graphics Performance Analyzer, which allows developers to assess performance and identify hardware bottlenecks via a HUD overlay that runs over their games. Intel also demonstrated its Platform Analyzer, which shows developers how much processing time their game spends managing elements such as HUDs and character renders.

Intel's project management lead, Roger Chandler, also discussed how Intel's products make the PC a viable platform for big-budget developers and smaller indie teams alike. "The PC is one of the most innovative platforms out there, and provides developers with the most ways to monetize their content," he explained. "We've reached the point in the ecosystem where digital distribution has

surpassed retail, opening up tons of business models, and we want to support that with our hardware."

**WEBGL WORKING GROUP**  
**Khronos Group**  
[www.khronos.org](http://www.khronos.org)

]]]] Near the end of the show, the not-for-profit industry consortium Khronos Group announced the final spec for the 1.0 version of the WebGL API, which enables HTML5 web browsers to handle hardware accelerated 3D graphics without plug-ins. The WebGL working group currently includes Mozilla, Google, Apple, and Opera, and the API ships with Firefox 4 and other supported browsers by the end of the year. The WebGL API exists on top of OpenGL ES, thereby applying its capabilities to web-based content. "We think this will help bootstrap Web GL developers and get more and more WebGL-based content on the web," explained Vladimir Vukicevic of Mozilla.

Khronos demonstrated how WebGL will benefit online content using Google Body, which creates 3D renders of the human body alongside traditional HTML HUD elements. "We think more applications are heading toward the web, since that will improve the impact of their delivery, so we want to make sure the web has the capabilities that all those applications need," said Vukicevic.

Khronos also aims to make WebGL available on mobile devices, a space which the group believes will increasingly favor Android handhelds in the coming months. "I think history is in the process of repeating itself," said Neil Trevett, president of the Khronos group. "You have Apple on one side, who is successful but has complete vertical control and a closed and controlled business model, while the other 90% of the industry piles in on the other side. In the desktop space, Apple takes up a small fraction of the current market share, even if they have good margins. The other side of the industry is traditionally more open and allows developers to more easily add value and innovate, and I think Android fills that role in the

mobile space. I think we're headed very quickly to an 80-20 split, with Android taking 80 percent."

With the WebGL API, Khronos hopes to give developers ample control over their applications, and give them a means of making their projects available to anyone with an HTML5-enabled web browser.


**NVIDIA DEVELOPER TOOLS, MOBILE GPUS**  
**Nvidia**  
[www.nvidia.com](http://www.nvidia.com)

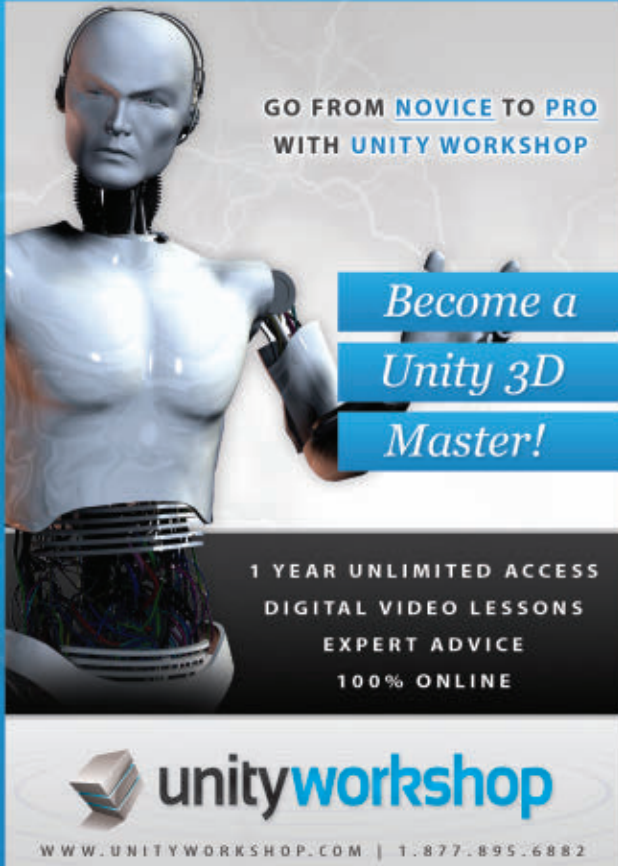
]]]] GPU and chipset provider Nvidia's booth showcased a wide variety of developer tools, services, and hardware. Among the company's most recent projects is the Nvidia 3D Vision technology, which adds 3D stereoscopic awareness to games via a new driver. This technology allows games to support stereoscopic 3D even if developers don't specifically integrate support for the feature themselves.

In terms of its middleware and developer tools, Nvidia revealed that its PhysX technology will now ship

with products such as Autodesk's 3ds Max and Maya. The company also showcased PhysX's Apex framework, which helps artists better integrate cloth animation into their games.

True to its hardware origins, their booth also showcased the latest Tegra 2 series of mobile GPUs. These dual-core processors are now shipping with a host of tablets and Android devices, and quad-core versions of these chips now in development. With this latest series of mobile hardware, Nvidia hopes to streamline the process of moving games from PC to mobile platforms, limiting the need to trim down game content to suit less powerful hardware.

In order to further improve performance across all its developer tools and hardware, Nvidia says it aims to make access to parallel processing more general, and find ways to keep more data in memory to help developers switch between applications far more quickly. 



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

*Become a Unity 3D Master!*

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

**unityworkshop**  
[WWW.UNITYWORKSHOP.COM](http://WWW.UNITYWORKSHOP.COM) | 1.877.895.6882





# PROGRAMMING SINS

## COMMON ERRORS FROM DOWN IN THE TRENCHES

////////// FROM THE VERY FIRST LINE OF CODE AN ENGINEER WRITES, he or she starts to develop their personal list of “dos and don’ts” that, even if they are never written down, have a tremendous effect on how we design and build our games. The list evolves and changes over time — we add to it, delete from it, and re-evaluate the lists of others. Experience is the driving force behind a lot of the changes; in short, we make a mistake, and we learn from it. These game programming sins are essentially a number of dos and don’ts with some recollections of how and why they came to be on this list.

MICHAELA A. CARR

### CREATING OBFUSCATED CODE

» For those that don’t know me, one of my major flaws is that I simply don’t remember everything. My brain, it would appear, is completely incapable of storing all the facts and information I ask it to. Over the years, I have used different methods to help me remember, all with varying degrees of success. My current system is to write everything down. I carry a black leather writing notebook with me, and I take lots of notes. I use a P.D.A. for some things like contacts and mind maps, but when it comes to making lists and notes, paper and pen have yet to be beaten.

One of the effects of this forgetfulness is that I couldn’t tell you the intimate details of a function I wrote three weeks ago, let alone six months or a year ago, without at least re-reading it and refreshing my memory. It’s because of this that I consider obfuscated code a sin. This also fits nicely with working in a team of programmers where someone might have to debug and/or add new functionality to someone else’s code. The quicker it is to understand, the easier it is for them to make the required modifications.

```
int m_MyMumIsBetterThanYours;
```

No it wasn’t a game about mums ... although I wonder if there is a game there somewhere ...

```
bool m_BumCheeks;
```

Enough said.

```
float m_Saving;
```

Is this a flag to indicate saving, in which case why a float? Or is it a percentage of save completed?

```
void * m_pAudioSample;
```

Not very useful. Wouldn’t `SAudioSample * m_pTheWarCry`; be more useful?

```
int CCharacter::GetLife( int y )
```

Nothing wrong with this function ... only why is there a variable passed in? More importantly, `y` isn’t exactly very descriptive. Turns out this function did indeed get the amount of life and return it; and while it was there, it also updated the life value by applying the damage modifier to the life counter, and also applied the adjustment of `y`. When this function wasn’t called every tick, the whole life counter on the character broke.

**Abusing ternary operations.** Consider the following code. (Remember that this would normally be on a single line, so you would have to scroll to see the entire line.)

```
if (CPhysicsManager::Instance().RayCast(m_Position + (CVector::Up * METRES(2.0f)), m_Position - (CVector::Up * METRES(2.0f)), &contact_data, pActor->GetPhysicsActor() ? pActor->FindRealActor()->GetPhysicsActor() : NULL, ePhysicsShape_Static))
```

Would you have spotted the use of `?` and `:` inside the function parameter list?

Some coding standards I have worked with ban the use of `?` and `:` altogether, mainly because it’s easy to abuse. As you can see, it contributes handily to the jumbled code in the example above. However, there are cases where I consider them to be fair enough. That’s usually where the use is obvious. For example:

```
m_Level = level_specified ? start_level : default_level;  
result = a > b ? b : a;
```

**Abbreviating English.** There was a time when the length of our variable or function names would have a significant effect on the performance of the compiler. This has not been the case for a very long time, but some engineers seem to like using shorthand.

```
int NmbrChars( );
```

vs.

```
int GetNumberOfCharacters( void );
```

```
int m_LCnt;
```

vs.

```
int m_LifeCount;
```

English is my first spoken and written language, and I find it easier to read code that says what it is in plain English.

### NOT FAILING GRACEFULLY

» The world would be a dull place if we all had the same thoughts, the same ambitions, ideas, and methods of working. But there are some thoughts and methods that should be discouraged whenever and wherever they are encountered.





ILLUSTRATION BY JUAN RAMIREZ

Working on a project that was just over halfway through its development cycle, I was investigating why the game kept crashing whenever a specific sound event was triggered. I quickly tracked the problem down to some missing audio files, an easy fix. Still, it was the fact that a missing file was causing the entire game to crash that got me looking a little more closely at the sound manager. It turned out that the manager never validated its data; even though a file had failed to load, it carried on processing the non-existent sound data as though the file load had been successful.

Talking to the engineer who maintained the system, I thought he was pulling my leg when he said he considered it acceptable behavior for the

code to crash when something goes wrong. After he repeated himself, I realized that he was serious. From his perspective, the problem was the missing data, not that the manager didn't handle itself in a graceful fashion.

There are always going to be unexpected issues that arise, certainly during development and very possibly after our games have shipped. There might be a missing texture, a corrupt file, an out-of-range data value, or even a lack of resources. We can and should make our code as bulletproof as we possibly can. The game should be continuously fighting to keep itself running. This makes the game experience more stable for the player, and if something does go wrong, there should be





fail-safes in place so that he or she will not even notice.

Here are three basic coding practices that I advocate as a minimum. You might also recognize what I'm talking about as essentially being "defensive programming."

**Pointers.** Pointers are pointers because it's possible they might be NULL (otherwise they would have been references). Validate pointers at least once before accessing them.

**Validate Data.** Sanity checking data can help prevent a lot of issues. Clipping them to valid ranges means you are less likely to have invalid calculations later on down the line.

**Default State.** Some data can't be clipped or ignored, an example might be a texture, in this scenario having a memory resident fallback is a good solution. During development it can be bright pink and yellow and stands out a mile while in shipped mode it can be transparent.

#### IGNORING CUSTOMER SATISFACTION

» During the early days of a very old project, I had added support for a trigger box to the game's editor. A few days later, a designer asked for a trigger sphere in addition to the box. I was busy, I had a pile of other work to do, and I didn't think it was that important for the milestone. I explained that I would add it to my list and implement it as soon as I could. Unfortunately, I didn't get to it soon enough; the designer announced to me the next day that he had solved the problem and didn't need my implementation anymore.

Have you ever stood on the edge of a very tall building and slowly looked over the side? Remember how your stomach felt as you slowly peeked over the edge? That was how I felt looking over the designer's shoulder at his solution. What I observed was a script that created 100 square trigger boxes, each rotated slightly more than the last. It made a pretty spiral effect in the editor and took a huge chunk out of the CPU when running in-game, but it worked exactly as he wanted it to.

This was a kick up my bum, and the designer got his sphere trigger very quickly afterwards. Designers, like everyone else in the games industry, are incredibly creative. The main thing about them in particular is that they seem to get far too much enjoyment out of abusing game systems. I consider it a sin to ignore any designer's valid request. Failure to pay attention to them might result in something creatively "ugly."

#### FRAGMENTING MEMORY

» During the development of many projects, there's been a critical moment when everything started falling apart. The game crashes constantly, levels are broken, data builds take twice as long, the coffee machine is out of order, and you're supposed to be going to a family event on the weekend, and you must desperately try to find the right moment to tell your better half that you won't be going. For me, this usually happens around six weeks before the end of the project.

What is going wrong in these instances is varied, but I've noticed a trend that usually revolves around the fragmentation of memory. There's nothing like spending time hunting down and defragmenting memory to make you realize how many of these issues could have been prevented.

**Temporary Buffers.** Reading this simplified example, it might seem obvious, but it happens more than I would expect. Having looked over file histories, cases of memory fragmentation seem to evolve over time when multiple engineers introduce additional initialization between the allocation and de-allocation. This has led me to consider temporary buffer allocations a sin—at least until they have proved themselves trustworthy.

#### Function A

```
Create a temporary buffer.  
Do something with the buffer.  
Call Function B.  
Finish processing the temporary buffer.  
Release the temporary buffer.
```

#### Function B

```
Allocate non-temporary data.
```

The allocation in Function B might simply be the loading of a file, creating a new entry in a link list, or allocation of a string. The result however is identical; fragmented memory. There are a number of ways to deal with this, not least of which is using static memory, a unique heap, or even a scratch buffer.

**Leaks.** Okay, it makes sense that memory leaks cause fragmentation. Luckily, with some good tracking tools, these are usually easy to find and plug-up.

**Keep It Simple.** Each allocation should have one de-allocation in a logical place. For example, if you allocate memory in the Init function, you de-allocate it in the **Deinit** function. Some engineers seem to think it's fun to hide the de-allocator in obscure parts of the code, or to have multiple de-allocate commands for the same piece of memory. Keep it clean, keep it simple!

#### NOEL LLOPIS

#### DON'T SYNC AND LOAD

» Late in a console project, I took on the gargantuan task of reducing level load times. They had been slowly creeping in throughout development and were up to a minute and a half. The goal was to bring that down under 30 seconds.

There were some obvious things that I was going to tackle: parsing and processing of text files, wild memory allocations, and so forth. Once I took care of all the low-hanging fruit, things were better, but load times were still way over a minute. Something was clearly wrong.

Curiously, the profile wasn't showing any huge hotspots, yet we were still spending over a minute loading levels. Where was all that time going?

After some more digging, I noticed that during the level load, we were drawing a progress bar on the screen. The loading code wasn't architected from the beginning to be multithreaded, so instead, we would load one file, update the progress bar, load another file, update the progress bar again, and so on until all files were loaded. We had about 3,000 tiny files per level (fortunately packed in a larger container file and laid out sequentially), so that made it possible to update the progress bar in a smooth way.

Other than being a bit clumsy and not very elegant, there was nothing horribly wrong with that approach. Except for one thing: the rendering code drew the progress bar, and then did a present call with vertical sync on. That meant that most of the time the console was waiting for vertical sync instead of doing an actual load. Once I removed the vsync, loading times went down to about 20 seconds!

**Bonus:** that weekend I went for a bike ride with a friend who is a game developer at another nearby company. I told him about our vsync issue and how they affected our loading times—we both got a good laugh out of it. It turns out, when he went back to work, he checked out of curiosity and they were doing the same thing, so he was also able to cut down their loading times in half!



## ANONYMOUS

### CUT-AND-PASTE

» When programming something up, I often copy a line or more of code, sometimes several. I then need to maintain both pieces of code identically.

What I should have done is move the code into its own function (or template, or at a pinch, a macro). The reason I don't is simply laziness. If I have some code like

```
DrawLine(a-w,b-h);
DrawLine(a+w,b-h);
DrawLine(a+w,b+h);
DrawLine(a-w,b+h);
```

and I need to do it again for c and d, it's really easy to just cut and paste, and change the variables.

```
DrawLine(c-w,d-h);
DrawLine(c+w,d-h);
DrawLine(c+w,d+h);
DrawLine(c-w,d+h);
```

When really I should do:

```
void DrawSquare(float x, float y, float w, float h) {
    DrawLine(c-w,d-h);
    DrawLine(c+w,d-h);
    DrawLine(c+w,d+h);
    DrawLine(c-w,d+h);
}
DrawSquare(a,b,w,h);
DrawSquare(c,d,w,h);
```

It's fewer characters of code, but it's more typing, and I have to think about it more. So I frequently go for the cut-and-paste "solution" first simply because it gets me the result I want quicker. Both will work the first time, but I pay for my sins, and usually end up having to refactor it away later, sometimes after a few more needless duplications.

### "PRINTF" DEBUGGING

» I've always found the most useful tool in debugging code has simply been to print out various values and labels that indicate where in the code we are, and what we are doing. I always feel a little guilty doing this, knowing there's a debugger with thousands of functions all designed to help me debug, but all I use it for is to look at the call-stack when I crash. It's like having a toolbox with a thousand tools, and all I use is the hammer.

The problem with this is that the console output quickly gets cluttered with pointless debug strings, and it becomes hard to spot the ones that are important. So I've got to track down the code that's spewing output, which can be hard to do if it's just something like `printf("%d\n", x);`.

Then, when I find the offending `printf`, I don't want to remove it, as it often took me several seconds to type and I might need it again in the future. So I just comment it out. Eventually, I started writing my `printfs` as if they were comments, so in a lazy kind of way, I documented something about the code. Of course I'd rarely re-use a `printf`, and when I did, it usually ended up needing re-writing anyway as the code would have been refactored and variables would have been added or removed.

### THOU SHALT NOT OVERENGINEER

» This one is unfortunately so common that a lot of people might not even think of it as a sin. It's just the way things are done. In several of my past projects, the whole company consisted of one team working on

a single game. Yet somehow, programmers were separated into game programmers and core technology programmers.

The idea was that the core technology team would write all code as reusable, game-independent libraries and tools that could be used in any other project in the future. The game team would use those libraries and then build any game-specific code they needed on top of them. Each of them had their own leads, and, of course, each of them had slightly different goals and preferences.

In practice, the division did more harm than good. The code the core technology team produced was overly general (for projects that didn't yet exist), and didn't solve the exact needs of the game team. It added extra dependencies and delays, and made things that should have been very simple much more complicated.

Was it worth it in the long term? Not at all. That code was used for direct sequels, but new games never reused the libraries and tools.

The lesson we learned the hard way is that before you can write reusable code, it first needs to solve the needs of a project. Or, put another way, there's no point to reusing code if it doesn't do the right thing in the first place.

Making a game is hard enough. Now, I just concentrate on making the best game we can. Later on, we can talk about extracting and refactoring some existing code that might benefit another project, but never try to predict and future-proof technology.

### PREMATURE OPTIMIZATION IS THE ROOT OF ALL EVIL ... OR IS IT?

» It began on a project a long time ago, in a galaxy far away ... OK, maybe not that far away but still quite a few years ago. I was a fresh college graduate and a bit wet behind the ears, with a brain full of computer science goodness.

Throughout the whole project, I kept pushing optimizations off until the end. "We aren't going to need that," and, "We'll just optimize the hot spots on the profiler," I kept repeating to my co-workers. So work went on, and performance never became a priority.

Every so often, the frame rate would tank because of some new feature that was just introduced, so we would fix that to make it playable again, but just barely. Frame rates were around 10–15 FPS for most of the project.

Then the day finally came. We were in beta and we had to bring frame rate up. I fired up the profiler and ... there it was! A big hot spot. I optimized it, proudly ran some benchmarks again, and noticed I saved half a millisecond. Not bad. I repeated it a few times, but curiously, the profiler soon reported a rather worrisome flat graph. Apparently, there was no single place in the code that was accounting for more than 0.05 percent of the frame time. What was going on? That's not what they taught in university!

It turns out our bottleneck wasn't CPU cycles so much as it was cache misses. And because of our lovely, heavy object-oriented design, we had constant pointer dereferencing and traversing graphs all over memory.

We managed to gain some performance back by changing some lists to contiguous arrays and doing some prefetching, but overall, it was very difficult to meet the performance requirements we wanted for the game.

Lesson learned: there are some things that are better thought of from the beginning, and memory layout and cache coherency is one of those. Don't paint yourself into a corner by waiting until the last minute to start thinking of them.

### THOU SHALT NOT...

» These are just a handful of sins, and there are certainly a lot more. But at one point in our careers, most of us in the programming field have committed almost all of them. If we hadn't, we wouldn't have learned our lessons! 📖

**MICHAEL A. CARR** has been an engineer in the games industry for over twenty years, publishing titles on all platforms from the 8-bit Amstrad CPC to the latest gaming consoles.

**NOEL LLOPIS** has been making games for just about every major platform in the last twelve years. He's now a one-man band making iPhone and iPad games.

**ANONYMOUS** developer is anonymous.



# THE HIGH ART OF GAMES

## DO GAME ARTISTS WARRANT RECOGNITION FOR INDIVIDUAL ACHIEVEMENTS?

**IN CASE YOU HAVEN'T HEARD, THE** venerable Smithsonian Institution in Washington, D.C. will be hosting an exhibition on video game art for the next six months. Curated by game archivist Chris Melissinos, with the advice of a panel of familiar industry names, the exhibition is intended to celebrate games as "as one of the most expressive, dynamic, and powerful canvases of expression in the past century."

What to say about this milestone in the history of our medium? First and foremost, "Take that, Ebert!" And Jack Thompson, Judge Limbaugh, and all the old fogies who can't tell the difference between GALAGA and RED DEAD REDEMPTION. We're legit!

It's pretty funny that this recognition has taken so long. Those interminable debates about "are games art?" always start from something that everybody agrees must be art, whether it's *War and Peace*, the Mona Lisa, or *Citizen Kane*. But somehow, they always ended up holding us up to standards that other media have long since abandoned. It's been more than a hundred years since Marcel Duchamp undermined the whole "art/not art" distinction by hanging bicycle parts and urinals in Paris galleries. For at least the last

50 years the academic and high-art worlds have agreed that the only objective test of whether something is "art" is whether somebody has hung it up in a gallery. So now, games can finally take their place alongside paintings, sculptures, Campbell's Soup cans, and sharks in formaldehyde in the pantheon of art. Congratulations, folks!

Treating games as an art form is old news. It's interesting to note, though, that when the show opens, the stars will be games and game designers, not pixel pushers like us. The great paradox of our business is the fact that our most modern of art forms is almost medieval in its approach to creativity. The popular idea of the "artist" as a gifted individual with a unique vision is an invention of the Renaissance. The cathedrals, frescoes, and manuscripts of the Middle Ages were created by artists who labored in anonymity, and the same is true for most of us. We labor communally, like monks illuminating manuscripts [granted, monks with too much caffeine and pizza in their monastery — but still]. Very few people, even among the most rabid fans of our games, will ever be able to identify the work of individual artists.

There are exceptions, of course. ArenaNet, for example, has built a very powerful identity for its concept artists so that names like Daniel Dociu and Kekai Kotaki are familiar to GUILD WARS fans, and also to readers of the Fantasy/SciFi award book *Spectrum*. Communities like ConceptArt.Org and DeviantArt have their named stars. The Ballistic press books have done a lot to popularize the work of individual CG artists, including game artists like Jan-Bart van Beek of Guerilla. And, of course, there are the marketing books from the "Art of ..." genre that provide artists a chance to speak directly for themselves.

Despite this small number of celebrity names, most of us are just credit list fodder. It's probably significant that most of the "name" artists in the business are concept artists, as reviewers, critics, and audiences can easily slot a creator of beautiful paintings into a traditional understanding of what it means to be an artist. The rest of us tend to contribute to our games in ways that are harder for the uninitiated to comprehend. If you're a character rigger, a shader artist, or you do the complex magic that makes your game's vehicles drivable, it's a lot

harder for a journalist or a fan to understand or appreciate your work. Behind all of that is the industry's high rate of churn. With so few people staying beyond a tenth anniversary in games [see the last year's Salary Survey, April 2010, for details], we don't have a strong sense of our own history or traditions.

The Smithsonian exhibit is the highest profile effort to help us create a shared history of games and the people who make them. It's not the only one, of course, and John Andersen's Gamasutra series on games preservation [see References] illustrates in great detail how hard it is to create the institutions that keep a tradition alive. [Editor's note: One of the best preservation efforts is undertaken by the Strong Museum of Play.] If it's difficult with games, it's almost impossible to trace the history of individual game creators, outside the handful of star designers. The games may remain but the people who make them come and go— anonymously, for the most part.

### STUDIO SYSTEM

» The best analogy for our contributions is, of course, film and TV show crews. We labor behind the scenes and our names scroll by during the credits, but only the hardcore fans know the names below the title. The trade crafts associated with film and stage production are closer to our collective kind of "art" than the traditional starving-genius-in-a-loft idea, so it's interesting to see how they've evolved ways of recognizing individual and group effort. Now that games get some uncontroversial cultural recognition, what about individual game artists?

The capstone of the film and stage worlds' approaches to recognition isn't museums, it's awards: including the Oscars, Tonys,



ETERNAL SONATA (360,PS3)





MICROSURGEON, DRAGON WARRIOR II, CHRONO TRIGGER, VIRTUA RACING, and GEARS OF WAR 3 illustrating the art of games through the ages.

and a bevy of less famous honors. These are typically put together by an industry group, such as the Academy of Motion Picture Arts and Sciences (which runs the Oscars). Earning an award can be the highlight of a career—even fairly secretive disciplines that don't get much public attention relish a moment in the spotlight.

Of course, we have our own awards. The Game Developers Choice awards are our closest approach to an Oscar-like seal of approval. Likewise, the round of "historical" postmortems at this year's GDC was an important step toward anchoring today's games in a historical context. But, as always, the focus is more on the games than the individuals who make them. Despite all the social prominence games have earned over the last decade, you're still a lot likelier to know a minor sitcom actor from the 1990s than the name of the modeler behind John Marston or the animator who gave life to EPIC MICKEY.

### YOU ANIMATED WHAT, NOW?

» Does this matter? We still get to make games, after all, and we get vicarious bragging rights through our part in the games we make. Isn't that enough compensation? The thrill of hanging around Best Buy on a release day to watch people snap up your baby is certainly hard to forget (although, in this era of direct digital distribution it's also getting rarer).

It's not exactly the same thing as individual recognition, though. After you've shipped a couple of titles, you become resigned to the fact that nobody outside your team

will ever really understand the things you contributed. The hidden dramas and covert heroism that every game goes through on its way to shipping are never going to be appreciated, not even by the small band of fellow game artists who might really understand them.

Obviously, this doesn't do much for our tortured artistic egos. However, the anonymity we work under also has other, more concrete consequences. Official recognition, awards, and other public honors don't just make you feel better: they enhance your bargaining power, and make you a more valuable asset to a team. It would be very handy to be able to tell a potential employer that you've gotten the equivalent of an Oscar. A team seeking publisher funding would love to be able to trot out a list of awards that proved their maturity and effectiveness.

Every established business finds ways to respect individual contributions. Call them Oscars, testimonials, or "employee of the month" parking spots, these pats on the back reflect a very basic human instinct. When you're just starting out in games, the rush of going off to work making monsters, animating talking animals, or detailing out lavish historical settings is enough to keep you going. When you've shipped a few titles and are starting to look at the long arc of your career, though, you may start to wonder where it all ends up. If management or art direction don't appeal to you, it's hard to imagine what comes next. Is it surprising that so many people

drop out of the industry after eight to ten years?

If we did do a better job of recognizing individual contributors, we might do a better job of retaining veteran talent. There are a lot of people in Hollywood, on Broadway, and in the music business who shape their careers around the recognition of their peers. To be the world's tightest film editor or most sought after session musician, for however long, is not the same thing as achieving red-carpet stardom, but it is a significant achievement. Such accomplishments are the kind of thing that helps you feel like the work is worth more than just a paycheck. One thing everybody in the games business should understand is the importance of positive feedback. As the people who gave the world Achievement mongering, we should do a little better for ourselves.

### RECOGNITION FROM WITHIN

» Creating a system that actually does recognize all of us down in the trenches is not easy. Creating the showy side of the setup is hard enough. Veteran Oscar watchers know how much politicking goes on behind the scenes, and how a popular title carries undue weight even in the technical awards. Excellent individual work in a sub-par game is going to be hard to spot, and it's hard to separate appreciation of discrete pieces from your feelings about the gameplay as a whole. It's easy, though, to get involved in the nomination process for the GDC Choice Awards (via [www.gamechoiceawards.com](http://www.gamechoiceawards.com)), so there's

no excuse for not making your views heard.

The other side of building up our collective sense of achievement is community building. Unlike Broadway, Hollywood, or even Nashville, our business is geographically scattered, and doesn't have the common fabric of a union to tie it together socially and professionally. That means it's up to us to create the communities we want to be part of. Get involved with IGDA SIGs or other groups like TechArt.org, ConceptArt.org, and CgTalk.com that cater to people in your specialty, to help create a community that fosters talent and shares knowledge. Go to GDC, particularly to small forums, like the art roundtables, where you can help create a common conversation about the realities of our fledgling art form. Most of all, pay attention to your teammates. Appreciate their individual contributions the way you'd like to have your own work appreciated. That's how it all really starts. 

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

### REFERENCES

"WHERE GAMES GO TO SLEEP: THE GAME PRESERVATION CRISIS," by John Andersen, Gamasutra, 2007: [www.gamasutra.com/view/feature/6271/where\\_games\\_go\\_to\\_sleep\\_the\\_game.php](http://www.gamasutra.com/view/feature/6271/where_games_go_to_sleep_the_game.php)



# A PLAYER'S STORIES

GAMEPLAY NARRATIVE IS OFTEN THE MOST IMPORTANT STORY IN THE GAME

**IN MY PREVIOUS TWO COLUMNS, I DISCUSSED** how designers could craft and integrate stories into their game. Narrative can take many forms in games, ranging in practice from the backstory paragraph that serves as context for many titles, to the branching, integrated stories found in story-based games like DRAGON AGE. There is one additional kind of storytelling to be found in and around interactive media: the stories players themselves choose to tell.

Much like developer-created narratives, players' stories can take a dizzying number of forms. Players stories might be inclined to borrow, mesh, and interweave with the game's narrative—or they may choose to ignore it in favor of their own narrative. Further, the stories could be entirely mechanical—about game rules rather than game fiction—or even purely social, in the case of multiplayer gaming.

While these stories vary in many respects, they all have one thing in common: they are the player's own. They star his own (and his friends')

often described by enthusiasts as communal storytelling. The players' options are limited primarily by their own imaginations, which are often desperately being reined in by a dungeon master who is trying to get them back on track and into the front door of the dungeon he spent all night designing.

It should come as no surprise that tabletop gaming experiences vary wildly from group to group, with the quality of the storytelling within based on how the imaginations of the party members interact. I hear a lot about pen-and-paper groups forming and dissolving in my circle of friends. I can usually pick out the ones that will meet more than once. A telltale sign is whether the participants feel obligated to post updates to their Facebook feed, because the stories that are arising are compelling enough for them to want to share.

## ROLEPLAYING IN MMOS

» Given that many massively multiplayer games have their roots in RPG design and offer virtual worlds in which players can live their virtual lives, roleplaying servers and guilds exist in almost every MUD and massively multiplayer game, with mixed success. A sizable minority wants to roleplay in MMOs, but some surprising obstacles emerge.

Shortly after the launch of ULTIMA ONLINE, players started to form roleplaying guilds fully composed of elves. The problem was, according to the lore at the time, there were no elves in the ULTIMA universe, so the mere existence of these guilds was upsetting to the roleplaying purists. Fundamentally, you had a clash of imagination.

In a tabletop game, you have a gamemaster in order to arbitrate these clashes. In many of the freeware text MUDs, MUSHes, and MOOs with a roleplaying focus, the people running the games help to manage the shared illusion. However, this issue is harder to manage with paying customers, and doesn't scale once you get to truly large MMO populations.

The second odd problem is that the definition of roleplaying varies from player to player, and even more frustratingly, the devotion to full-time roleplaying tends to degrade over time. Players play these games a lot, and it is very hard for them to maintain a barrier between their real self and their avatar, especially if they really connect with friends and guildmates online and want or need to relate real-life feelings, problems, or triumphs. As such, the quality of roleplaying on an RP shard tends to erode, day by day, as players exercise their need to connect with friends, and their persona more and more becomes a mix of their virtual identity and their real one.

In the current state of things, the designer should help players who want to roleplay find each other, and more importantly, find those with compatible shared fantasies. Help the players who want to roleplay as elves find each other, and let guildmasters take the responsibility for maintaining consistency (at least internally). However, finding better ways to foster and encourage better roleplay is an area of opportunity for the enterprising MMO designer.

## SIMS: THE ULTIMATE STORYTELLER'S GAME

» True roleplaying games like D&D are considered to be the geekiest of geek hobbies, and yet to find a computer game that truly captures the spirit and imagination of roleplaying, you need to go to the most casual friendly game on the best-seller's list: THE SIMS.

Dismissed by many hardcore gamers as a toilet cleaning simulation, this virtual dollhouse is indeed a hotbed for player stories, and the design has been crafted that way. The rules are kept light, and the players are granted relatively easy access to whatever items, architectural elements, and character appearances they might want. Later sequels also give players easy access to tools to take screenshots and movies, add captions, and upload them to the community—and so they did, uploading thousands of diaries and stories up to the net. Like all player-created content, a few are excellent, some are good, and most are very, very bad, and there's a design challenge in being sure browsers find the good stuff. But all of them represent a true level of player investment into not just THE SIMS, but the culture and community that surrounds it.



imagination and events. When the designer's narratives have to compete with these stories for attention and brainspace, he faces an uphill battle. Rather than fear or fight these narratives, the designer should look for how to integrate and leverage them.

## PLAYING WITHIN THE LINES

» One way that a player can contribute their own narratives is when the game has rules designed to allow them to do so, contributing to the narrative within the confines of the game rules. The classic example of this is, of course, tabletop roleplaying games such as *Dungeons & Dragons*, which has been





Browsing through the stories, it quickly becomes clear that they do not come purely from the players' imaginations. In many cases, it is clear that these stories are writing about events that occurred to their Sims over the course of gameplay: "Mary started to flirt with the firefighter, and to her surprise, the firefighter started to flirt back." This is not a story straight from the mind of the player. It is closer to being a diary of the player's experimentations within the game, and how the Sim chooses to respond. And it is wildly successful, largely because THE SIMS is so broad and open-ended that the player can continually be surprised by the results of his actions.

### THE STORYTELLING IN MECHANICS

» You don't need designer narrative at all to get players to share stories about the games they are playing. You just need interesting game mechanics. Indeed, one only has to look online to find reports about *Chess*, *Scrabble*, and *Magic the Gathering* tournaments that have rapt readership. The storytelling is all mechanics: she left her queen exposed; he dropped the Q on the double letter score but left access to the triple-word; the magic player was going to die in one turn, but topdecked (i.e. drew) the one card that could save him.

We had piles of lore in *Shadowbane*, all of which we were very proud of, but the stories we put on our website that were the most gripping were the guild reports of city sieges from "in the trenches." They gave a real sense of what it was like to take part in the front, with full blow-by-blow accounts of bravado, logistics, war, desperation, treachery, triumph, and defeat. Little or no mention was made of the real backstory; it was all ancillary to the real action.

### MECHANICS GOING VIRAL

» The nice thing about the *Shadowbane* war reports was they had an immense ability to go viral. While the exact mechanics of a *Shadowbane* city siege were somewhat obtuse and hard to grasp, the general basics were not

too far removed from real life — build catapults, amass some armies, and have at it — which allowed them to inflame the imagination. Stories are easier to tell, and have better resonance, when non-players can easily grasp the gist of it.

If the mechanics are not immediately evocative, then you depend on the listener knowing the rules for him to have any appreciation for the story. If you tried to tell me about your epic *Go* match, I'd be utterly baffled. I don't know the mechanics, and thus probably lack any appreciation for the subtleties of your position to understand what the big deal was about. This tends to be true of many games, especially board games, where the rules are abstract or the competition relatively indirect.

Two such games are *Kingsburg* and *Agricola*. These are fantastic, top-rated board games, but describing a closely fought match is very difficult to do due to the nature of the rules and backdrop. By comparison, *Pandemic* leaves the players with great stories of being trapped in Asia when a viral outbreak wipes out the Eastern Seaboard and loses the game with one turn to go. Here, the narrative backdrop the game uses isn't wasted but it creates a player narrative interweaved with the game's backdrop, making it tangible, easy to grasp, and evocative to non-players.

Figuring out how to let players communicate to non-players can be a big win. Blizzard went to great lengths to compress the size of STARCRRAFT 2 replays so that watching a shared movie of a high level STARCRRAFT 2 match essentially tells a story for fans of the eSport. These replays can also give the player more context and details to their story: for example, the replay at the end of the original CIVILIZATION, for example, showed what was happening to the other players beneath the fog of war, which helped flesh out the details of the player's triumphs and travails.

### DRAMA IS COMPETING CONTENT

» In any multiplayer setting, but especially in MMOs, there is one potential source of incredibly

powerful narratives that the designer has very limited control over. This is a type of narrative that arises when other players group and interact with each other. Sometimes these interactions are positive—becoming smitten, for example—but often they're negative. One obvious manifestation of this that many have encountered is guild drama.

Put simply, if your guild's best healer is cybering the guildmaster's girlfriend and gets caught, well, at that point, any narrative that the game tries to provide is fighting an uphill battle to get any kind of attention.

One part of the magic of multiplayer is that other players are, in fact, content. The designer should encourage interactions between players, especially positive ones, but he should also be mindful that a player's attention is limited, and in a multiplayer environment, he is prone to distraction at unexpected times from unexpected directions. An amount of designer narrative that is wholly appropriate and well paced in a single-player game might prove to be overwhelming when combined with the additional stimuli of a multiplayer environment.

### EMBRACING PLAYER STORIES

» Designer narratives are important to a game, though this is truer in some cases more so than others. In story-driven games, especially, these narratives are vital, and a designer's first instinct is to keep the player on the rails to ensure that they experience his story, in the right way.

But emergent stories can be as powerful as the handcrafted stories designers create—sometimes even more so. Even though they frequently lack the quality or polish that is wrapped around the designer's narratives, the personal investment that the player has in these stories is difficult to compete with. The designer is well advised to ensure that these player-driven narratives have room to breathe, especially in multiplayer environments.

Designer narratives and player stories can, and should, coexist. A story that is completely rigid and on rails misses out on all of the magic that comes with the interactive entertainment medium. On the other hand, depending entirely on players for your story is effectively hoping all your players serendipitously stumble upon greatness on their own. The designer narrative should be the backbone of the experience, but we should also recognize that player storytelling within or about a game can take the experience to a whole new level. 🎮

---

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



EXCEEDING  
IMAGINATION

E3 Expo is the preeminent global trade event for computer and video games. It's all about the innovation, creativity, business, and imagination of the most compelling sector of the entertainment industry.

EXPERIENCE ALL THAT E3 EXPO HAS TO OFFER BY REGISTERING TODAY AT [WWW.E3EXPO.COM](http://WWW.E3EXPO.COM).



LOS ANGELES CONVENTION CENTER

**JUNE 7-9, 2011**

[WWW.E3EXPO.COM](http://WWW.E3EXPO.COM)



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.

© 2011 ENTERTAINMENT SOFTWARE ASSOCIATION





# INDIE AUDIO JONESING

ADVENTURES ON THE NEW FRONTIER OF AUDIO

**WHILE MAJOR PUBLISHERS AND** developers are seeing layoffs and economic contraction, the app market is thriving on both the iOS and Android platforms. Fueling this boom in handheld gaming are hundreds of independent developers who have found the smaller, faster pace of development a boon to their ability to quickly create, publish, and sell. These are relatively new companies with small teams, members of which may be spread across the globe. Already, though, indie games like Rovio's *ANGRY BIRDS* are responsible for seismic shifts in distribution, design, and the make-up of game audiences.

To get a sense of how these companies are handling the process of audio content creation, I spoke to a number of developers, all of whom have had games in the iTunes Top 20 charts within the last month.

## FORTUNE AND GLORY

» The first thing that becomes clear when talking to indie developers about audio design is that they're not working from a standardized playbook. Like many aspects of their development process, they're writing the rules as they go along. Audio professionals working with indie developers need to be flexible enough to work within a variety of different corporate structures, pay schemes, and milestone schedules. For Moscow-based ZeptoLab, developer of *CUT THE ROPE*, contractors are brought onboard once the game is well into development. "Usually, we start working with audio once we've reached the Alpha milestone," says Semyon Voinov, Zeptolab's Creative Director. "At that time we have a playable game with all major features implemented." Once Alpha is reached, audio professionals are brought onboard in a traditional full buy-out model.

For South Carolina developer Thunder Game Works, audio contractors are equal partners in the game's development. "To be as successful as we have been, we provided each of our members with a portion of the revenues earned for the games, which encourages them to give it their best, as their own success depends on how well they do," explains *TRENCHES* developer Kris Jones. "Audio is no exception as it is critical in helping portray an emotion that we want the player to experience."

For those hiring external contractors, talent seems to often come from their local areas; but not all developers are hiring external contractors. Some teams have audio duties performed in-house, usually a double-duty task akin to the earliest days of game development. For Toronto's Get Set Games, Inc., artist Nick Coombe is also the company's resident audio guru. According to Coombe, "Audio plays a big part in the overall experience of our games, so audio production kicks off even at the earliest stages of development." For the iPhone's *MEGA JUMP*, Coombe and his team approached "game design holistically; each element—from game design and feel, to artwork and visual effects, to sound effects and music—works with and enhances the other aspects of the game to create the full experience, so audio is never an afterthought."

Still others like *SKYBURGER* developer NimbleBit find the availability of online sound effect marketplaces to fill all of their needs and tackle audio development themselves through sites like SoundDogs.com or SoundRangers.com.

## THE NEXT CRUSADE

» These aren't large developers trying to figure out how to shrink a triple-A console game down to

a touch screen. These are small teams fully embracing touch screen development and the specific quirks of the smartphone platform. They're scrappy, imaginative, and dedicated to creatively getting as much audio into their games as possible.

"The only real limitation we are always considering while creating iOS games is game package size," explains *CUT THE ROPE*'s Voinov. "There's a 20MB limit on the AppStore for applications which can

felt restricted by the smartphone platform's limitations. "Quite the opposite," he explained. "The iPhone platform allows for streamlined integration of audio. In *TRENCHES*, we've even allowed players to speak with each other over chat during a multiplayer match. A benefit to the iDevices is that players can mute in-game music and play music from their own library to suit their mood," a feature still not standardized across PC and console games.



ILLUSTRATION BY JUAN RAMIREZ

be downloaded through the cell networks, and that doesn't allow us to insert a lot of audio content." Nick Coombe adds, "The platforms we use to develop—Cocos2D and Cocos Denshion for audio—handle a variety of formats, which we experimented with before settling on AAC for both effects and music, which gives us the compression we need and the quality we want without having to resort to low sample rates or mono samples."

Despite challenges like small download packages and a tendency for mobile gamers to turn the sound off, it's also clear that the developers I spoke with all regard audio a fundamental element to their games. I asked Kris Jones if the Thunder Game Works team

With so many Apps vying for attention, smart indie developers are searching for every memorable hook they can get, and audio is part of that strategy. Says Voinov, "In the early days of the AppStore, you could find some very simple applications which play funny sounds doing very well at the top of the charts. We noticed that, and in *CUT THE ROPE* came up with the idea of using a fart-like sound for the air cushion elements spread across levels. That worked out well. We regularly read positive reviews mentioning this small trick." 🍑

**JESSE HARLIN** has been composing music for games since 1999. He is currently the staff composer for LucasArts. You can email him at [jharlin@gdmag.com](mailto:jharlin@gdmag.com).



# TEAM PLAYERS

## WHAT YOUR SMALL, DISTRIBUTED TEAMS DO AND DON'T NEED

**AS I MENTIONED IN MY PREVIOUS** column, my company, Spry Fox, currently has several original free-to-play games in development, not including ports of our existing IP. Each game is being produced by wholly separate teams that are geographically dispersed, using different technologies and tools, under different contractual arrangements. And each team is compensated entirely via their future royalty; none are being paid cash in advance.

While we won't know for a while to come whether our development strategy has been wise or flawed, we've already learned a great deal about the ideal composition of small, geographically-dispersed development teams. Some of our active teams have exceeded our expectations in terms of game quality and development time, while some are significantly behind where we expected them to be by now. A few of the characteristics shared (or not) by the high-performing and slower groups may be obvious to you, but some may surprise you.

### CHARACTERISTICS OF HIGH-PERFORMING TEAMS

» *Communicating clearly and frequently.* Our teams with a predisposition toward communicating more rather than less are getting much more accomplished. We've found that it's hard for a small team to over-communicate (as opposed to a huge team, which can easily become bogged down by too much pointless communication). But it's very easy for a small team to grind to a halt when a solitary, isolated member encounters difficulties of any kind.

Takeaway: To some extent, communication risks can be minimized by scheduling regular meetings, encouraging a social atmosphere, and so on, but in a distributed environment there's

only so much you can do. If someone on your team is a hermit or lone wolf, you may be in trouble. You need team players who can communicate.

» *Iterating rapidly.* It's common knowledge that rapid iteration is the key to "finding the fun" in any game development project aspiring to originality. However, our view on this has become relatively radical; we shoot for daily iteration. We've found that iteration times of even just a week (speedy for most larger studios) will severely hamper the progress of our projects. Iteration times of two weeks or more usually signal that a project is in severe jeopardy. There are exceptions to every rule (for example, sometimes it may be necessary to invest in technical infrastructure, which will temporarily slow down iteration) but in general we've found fast iteration to be vital to team success and morale.

Takeaway: It's everyone's responsibility to ensure that development is progressing in such a manner as to permit rapid iteration. Don't allow the team to commit to a development path littered with the kinds of technical challenges that might cripple the iterative cycle. And don't allow the team to become mired in the unexpected challenges that will inevitably arise despite your best efforts; find a creative way to work around them or change your design as necessary. Forward momentum is a small team's best friend.

» *Committed and reliable.* Another characteristic of strong teams is that their members tend to be comfortable negotiating reasonable commitments and generally follow through on those commitments. While this should be obvious to anyone, the extent of its importance cannot be understated. We've found reliability to be the single biggest predictor of a team's success—far above intelligence, passion, or experience in importance. A small

team with a single unreliable member is in greater jeopardy than a team lacking all the other positive characteristics noted in this list.

Takeaway: Reliability is one of the most difficult characteristics to screen for in an interview. One of the major benefits of working with someone as a contractor, as opposed to full-time hire, is that you learn from experience just how reliable (or not) they are before making any major commitments to them. But whether you're working with contractors or employees, helping people resolve the issues that make them unreliable—or gracefully parting ways with those who can't be helped—will be one of your most important challenges as a studio manager.

### THINGS A HIGH-PERFORMING TEAM DOES NOT NEED

» *Willingness to work long hours or to crunch.* Not one of our high-performing teams has resorted to working unusually long hours for any period of time. We have already shipped four games without ever crunching, and this year we'll ship several more without ever crunching. In fact, we have just one team that has ever engaged in anything even remotely resembling crunch; ironically and not coincidentally, doing so did not appear to actually move the project forward significantly. The phrase "work smarter, not harder" may be a Dilbert punchline, but in the game development world we need to hear it more often.

» *Shared location.* As noted earlier, we work with people all over the world, from South America, to Europe, to Japan, to Australia. All our teams are composed of individuals who live nowhere near each other. What we've found is that a team with the positive traits noted earlier can easily overcome any challenge presented by such geographical dispersion.

» *Passion.* Our industry is obsessed with the stereotype of

the passionate indie, willing to work himself (or herself) to death in pursuit of a vision. But what we've found is that there's a base-level of passion that almost everyone we encounter shares (why else would you even be in this industry?), and exceeding that base-level of passion simply isn't necessary. Extreme passion, more often than not, seems to get in the way of compromise; specifically, the kinds of compromise that enable a team to function properly.

### WRAPPING UP

» It's worth noting that the bar for team composition goes up when you switch from developing single player content to F2P games with a real backend. The first four games launched by Spry Fox were all of the former type, and our transition to the latter has not been painless. We've found that even the simplest server-backed games can be exponentially more challenging to develop, especially when the team lacks some of the fundamental traits noted earlier.

Bottom line: When you transition from developing single player games to social or multiplayer games, and/or when your teams are geographically dispersed, character traits and team dynamics that were previously minor annoyances can suddenly become fatal. Watch out for poor reliability, poor communication, and slow iteration: These things will guarantee that your game does not ship in a reasonable state of quality and/or within a reasonable period of time. ☹

---

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





## new studios

✂ Activision recently unveiled its newest owned studio, Beachhead, which will be tasked with creating all of the company's new digital initiatives for the CALL OF DUTY brand, including online community, content and services initiatives.

✂ Germany-headquartered Idea Fabrik quietly purchased the HeroEngine development platform and technology from Simutronics late in 2010, and now the company has founded a new game development studio in northern Virginia called Second Star Interactive.

✂ Video game designer and industry veteran Don Daglow recently revealed Daglow Entertainment, LLC a studio focused primarily on Facebook and mobile titles.

## whowentwhere

✂ BioWare Austin's VP and co-general manager Gordon Walton is leaving the studio to take on an executive producer role at social game developer Playdom.

✂ GamesAnalytics, a datamining and monetization firm targeting online games, announced that Activision co-founder Alan Miller has joined the company as a strategic advisor and director of its North American operations.

✂ Disney Interactive Studios' former director of game design Frederic Markus, most recently credited for his work on EPIC MICKEY, has taken a new position as the studio creative director for LucasArts.

✂ After leaving Capcom, former producer Ben Judd is now heading a new Japanese base of operations for talent agency Digital Development Management.

✂ Longtime industry veteran Brenda Brathwaite has left Ravenwood Fair studio LOL Apps to join her longtime colleagues, John Romero, Robert Sirotek and Tom Hall, at Lot Drop, their new social gaming studio.

✂ id Software's digital distribution general manager Steve Nix has left the D00M and QUAKE developer for major U.S. game retailer GameStop

## FROM RUSSO WITH LOVE

### EX-NEXT GENERATION EDITOR MOVES TO BIZ DEV

*Tom Russo spent over a dozen years as a journalist, working at companies such as G4, and the seminal Next Generation magazine. Transitioning out of journalism, he spent three years as a consultant, before finally settling into a full-time position as a business developer at Foundation 9.*

**BRANDON SHEFFIELD:** *You were a journalist for a very long time, with stints in consulting—what made you decide to move to the studio side?*

**TOM RUSSO:** Three years of consulting provided an opportunity to leverage what I had learned as a 15-year student of the industry, and gave me visibility into the inner workings of the variety of publishers and companies I worked with. But ultimately, I felt the need to be part of something bigger. I was already very familiar with Foundation 9 from my former Next Generation magazine colleague Chris Charla, who had been with Foundation 9 for about 10 years. Having seen how the company had grown, I knew the pace would be more akin to a publishing environment, with lots going on at once, but offer the benefit of being closer to the products in a creative development environment.

Game developers are, without question, my favorite people in the world. As a former member of the enthusiast media, it's been a pleasure to try to shed some light on their brilliance. However, I have a lot of friends on the publishing side as well. As a game industry journalist, you become a student of both publishers and developers. Leveraging my relationships and understanding of both sides felt like the best use of my talents. Why work for one studio when you can work for six? And the team at Foundation 9 has been incredibly welcoming and supportive.

**BS:** *Foundation 9 has a lot of studios and moving parts. How do you approach that as a business developer?*


**TR:** You really can't do business development for "Foundation 9" per se, it's really all about the six studios—Griptonite, Sumo Digital, Double Helix, ImaginEngine, Pipeworks, and Backbone. Ultimately, it's up to each studio to decide what projects make the most sense for them, and our group works to create options for them. Being a business developer here is really about putting the right studio with the right proposals in front of the right content providers. The goal is to make everyone happy, and as cliché as it sounds, to create "win-win" scenarios. I've had the pleasure of meeting and working with a ton of people in

this industry, and I love it; there are so many great people in this business. I try to approach it from a very positive place—ultimately publishers and developers want the same things.

It's been fascinating because business development is really one of the few areas of the game industry that is particularly guarded from journalists, at least in my experience working at enthusiast outlets. In my first week with Foundation 9, I was exposed to proposals, milestone schedules, and so forth—things that you'd never see as a journalist.

**BS:** *Following from that, do you have to learn each company's strengths, focus, and history? Is there a learning curve there?*

**TR:** Absolutely, each studio is unique, with its own strengths in designing for different hardware platforms, genres, and games for different audiences. For example, Griptonite already has a great lead on 3DS with three titles in development there. Backbone has been a huge provider of content for XBLA. Not everyone knows that Double Helix was formerly Shiny and The Collective, and they are now a combined powerhouse that is going to ship the forthcoming Green Lantern game in time for the movie.

I just met all the studio heads in person at GDC, and got to see them in action. From my experience as a journalist, the best game developers have told me time and again that their games are passion projects. While we do a lot of "work for hire" at Foundation 9, our studio heads are definitely passionate. They presented some compelling new IP at GDC this year and received some very positive feedback. So even in the short time I've been here, it's been incredibly rewarding to help facilitate these opportunities. Beyond that, we have some companywide initiatives that involve every studio that are very exciting, you'll need to stay tuned for more on that. 



# SHOOT INTO THE FUTURE...

- 2000 TONY HAWK'S PRO SKATER 3
- 2001 TONY HAWK'S PRO SKATER 4
- 2002 TONY HAWK'S PRO SKATER 4
- 2003 TONY HAWK'S UNDERGROUND
- 2004 TONY HAWK'S UNDERGROUND 2
- 2005 TONY HAWK'S AMERICAN WASTELAND
- 2005 GUN
- 2006 TONY HAWK'S PROJECT 8
- 2007 TONY HAWK'S PROVING GROUND
- 2007 GUITAR HERO III: LEGENDS OF ROCK
- 2008 GUITAR HERO: AEROSMITH
- 2008 GUITAR HERO: WORLD TOUR
- 2009 GUITAR HERO: METALLICA
- 2009 GUITAR HERO 5
- 2009 GUITAR HERO 5
- 1996 SKELETON WARRIORS
- 1998 APOCALYPSE
- 1999 TONY HAWK'S PRO SKATER
- 2000 SPIDER-MAN
- 2000 TONY HAWK'S PRO SKATER 2
- 2001 TONY HAWK'S PRO SKATER 3
- 2002 TONY HAWK'S PRO SKATER 4
- 2003 TONY HAWK'S UNDERGROUND
- 2004 TONY HAWK'S UNDERGROUND 2
- 2005 TONY HAWK'S AMERICAN WASTELAND
- 2005 GUN
- 2006 TONY HAWK'S PROJECT 8
- 2007 TONY HAWK'S PROVING GROUND
- 2007 GUITAR HERO III: LEGENDS OF ROCK
- 2008 GUITAR HERO: AEROSMITH
- 2008 GUITAR HERO: WORLD TOUR
- 2009 GUITAR HERO: METALLICA
- 2009 GUITAR HERO 5
- 2009 BAND HERO
- 1996 SKELETON WARRIORS
- 1998 APOCALYPSE
- 1999 TONY HAWK'S PRO SKATER
- 2000 SPIDER-MAN
- 2000 TONY HAWK'S PRO SKATER 2
- 2001 TONY HAWK'S PRO SKATER 3
- 2002 TONY HAWK'S PRO SKATER 4
- 2003 TONY HAWK'S UNDERGROUND
- 2004 TONY HAWK'S UNDERGROUND 2
- 2005 TONY HAWK'S AMERICAN WASTELAND
- 2005 GUN
- 2006 TONY HAWK'S PROJECT 8
- 2007 TONY HAWK'S PROVING GROUND



- 2001 TONY HAWK'S PRO SKATER 3
- 2002 TONY HAWK'S PRO SKATER 4
- 2003 TONY HAWK'S UNDERGROUND
- 2004 TONY HAWK'S UNDERGROUND 2
- 2005 TONY HAWK'S AMERICAN WASTELAND
- 2005 GUN
- 2006 TONY HAWK'S PROJECT 8
- 2007 TONY HAWK'S PROVING GROUND
- 2007 GUITAR HERO III: LEGENDS OF ROCK
- 2008 GUITAR HERO: AEROSMITH
- 2008 GUITAR HERO: WORLD TOUR
- 2009 GUITAR HERO: METALLICA
- 2009 GUITAR HERO 5
- 2009 BAND HERO
- 1996 SKELETON WARRIORS
- 1998 APOCALYPSE
- 1999 TONY HAWK'S PRO SKATER
- 2000 SPIDER-MAN
- 2000 TONY HAWK'S PRO SKATER 2
- 2001 TONY HAWK'S PRO SKATER 3
- 2002 TONY HAWK'S PRO SKATER 4
- 2003 TONY HAWK'S UNDERGROUND
- 2004 TONY HAWK'S UNDERGROUND 2
- 2005 TONY HAWK'S AMERICAN WASTELAND
- 2005 GUN
- 2006 TONY HAWK'S PROJECT 8
- 2007 TONY HAWK'S PROVING GROUND
- 2008 GUITAR HERO: AEROSMITH
- 2008 GUITAR HERO: WORLD TOUR
- 2009 GUITAR HERO: METALLICA
- 2009 GUITAR HERO 5
- 2009 BAND HERO
- 1996 SKELETON WARRIORS
- 1998 APOCALYPSE
- 1999 TONY HAWK'S PRO SKATER
- 2000 SPIDER-MAN
- 2000 TONY HAWK'S PRO SKATER 2
- 2001 TONY HAWK'S PRO SKATER 3
- 2002 TONY HAWK'S PRO SKATER 4
- 2003 TONY HAWK'S UNDERGROUND
- 2004 TONY HAWK'S UNDERGROUND 2
- 2005 TONY HAWK'S AMERICAN WASTELAND
- 2005 GUN
- 2006 TONY HAWK'S PROJECT 8
- 2007 TONY HAWK'S PROVING GROUND

**LEAD MULTIPLAYER DESIGNER • MULTIPLAYER DESIGN SCRIPTER**  
**SR. LEVEL DESIGNER • LEVEL BUILDERS • CONCEPT ARTIST**  
**SR. ENVIRONMENT ARTIST • SR. TECHNICAL ARTIST**  
**LIGHTING ARTIST • PC PLATFORM LEAD ENGINEER**

**APPLY @ [HTTP://WWW.NEVERSOFT.COM/SITE/HIRING.HTML](http://www.neversoft.com/site/hiring.html)**





# SOLACE

<http://solacegame.com>

A BULLET HELL SHOOTER IS AN UNLIKELY FORUM FOR EXPLORING KÜBLER-ROSS' FIVE STAGES OF GRIEF, BUT IN THEIR GAME SOLACE, THE ONE MAN DOWN TEAM AT DIGIPEN HAS FOUND NEW EXPRESSIVE POSSIBILITIES IN THE GENRE. WE SPOKE WITH SOLACE PRODUCER JORDAN HEMENWAY TO FIND OUT HOW THE 2010 PAX 10 AND 2011 IGF STUDENT SHOWCASE WINNER CAME TOGETHER.

**JEFFREY FLEMING:** *The bullet patterns in SOLACE are really lovely. Were you able to use any libraries, or were they coded from scratch?*

**JORDAN HEMENWAY:** At the beginning of the project, we sought help from a DigiPen alumnus who had worked with bullet hell-styled games in the past. With help and suggestions, Dan Rosas wrote a library of behaviors that could be chained together using an action-list method. Most of the patterns resulted from a visual idea that Dan would think up, like the raindrops falling. He would then tweak the calculations and timing until we had something visually interesting while still playable in-game.

**JF:** *In SOLACE, music is generated from the action on the screen. What methods did you use to ensure that the result was musical rather than cacophonous?*

**JH:** One of the most challenging things for us in developing SOLACE was finding ways to keep the music sounding pleasant while using a bit of randomness and repetition.

We started off by keeping all sounds triggered by the player and enemies on beat, and we used basic music principles like picking a specific scale and selecting certain notes from it. For many instruments, like the piano in Denial or the synth instrument in Bargaining, picking random notes from a bank worked well at any time with background layers.

However, other instruments, like the guitar in Anger, didn't sound correct since guitars don't usually lend themselves well to random notes. Instead, for those cases we used a straightforward

progression that was broken up into several tiny pieces for each bullet. It took a lot of iteration to figure out what made sense to the ear. For example, we ended up going through at least seven versions of the Anger level to get the music into its current state.

**JF:** *SOLACE was your sophomore project at DigiPen. Does the One Man Down team stick together for the next year's project or do you all join new teams?*

**JH:** Yes, it is true that SOLACE was a sophomore project, but that is a bit misleading. The game was a project started by three sophomore programmers, but we were able to snag our artist, Jami Lukins, to do all the art for the game despite her being a senior at the time. She is now happily graduated and employed.

As far as the programmers are concerned, I joined a team with other friends. We are currently developing an open-world exploration game. Robert Francis and Dan made a team together and have been working on a music-based platformer along with their team of artists and other programmers. Shortly into the spring semester, Dan took an internship at the Seattle-based game studio Fuelcell. So sadly, the team is split in four directions working on their own separate projects, but who knows what the future might bring?

**JF:** *Where did the idea to incorporate the five stages of grief into SOLACE come from? It's an odd but interesting framework for a shooter-style game.*

**JH:** During the early stages of the production of SOLACE, Robert Francis, our technical director, lost his twin brother. The team name "One Man Down"

came from the fact Robert was missing from the Engine Proof milestone presentation due to this tragedy. Robert's loss had affected the whole team, and he says he wouldn't have been able to continue that school year if it hadn't been for the support from his teammates, and the game giving him something to focus on.

We entered the second semester and Dan came up with the dynamic music mechanic we now have in the game, and when we were deciding on the artistic direction, overall theme, and scope of the game, the five stages of grief just seemed to be the perfect match. If you wait until the end of the credits, there's a dedication to Robert's twin brother, Nathan.



**JF:** *Bullet hell shooters are typically extremely challenging and for the hardcore only. What did you do in SOLACE's design to make it more accessible for players but still retain the visual intensity of bullet hell?*

**JH:** While we were planning SOLACE, we talked about difficulty and how the general style of a bullet hell shooter can lend itself to difficult gameplay. After looking at our options we decided that we would try lightening the overall difficulty so that more casual players could

survive long enough to see the beautiful bullet patterns.

We attempted to keep things simple, eliminating the HUD and displaying all important information on the player. We were also much more forgiving to our players, allowing them to take multiple hits before losing the level. In keeping with our simple design, we decided not to add bombs or other power-ups during gameplay. Doing so would have required some form of HUD, taking away from the elegant design.

**JF:** *What was the biggest challenge in developing SOLACE?*

**JH:** Overall, the hardest part about making SOLACE was trying to express an abstract emotional

component during each stage through the gameplay.

Some stages lent themselves to fairly straightforward interpretation, like Anger and Depression, making it easier to represent visually and audibly. Stages like Denial and Bargaining however, left us thinking how a musical instrument, bullet patterns, or even color could represent such a concept or feeling. In the end it was definitely our most interesting challenge, and one we have learned a lot from.

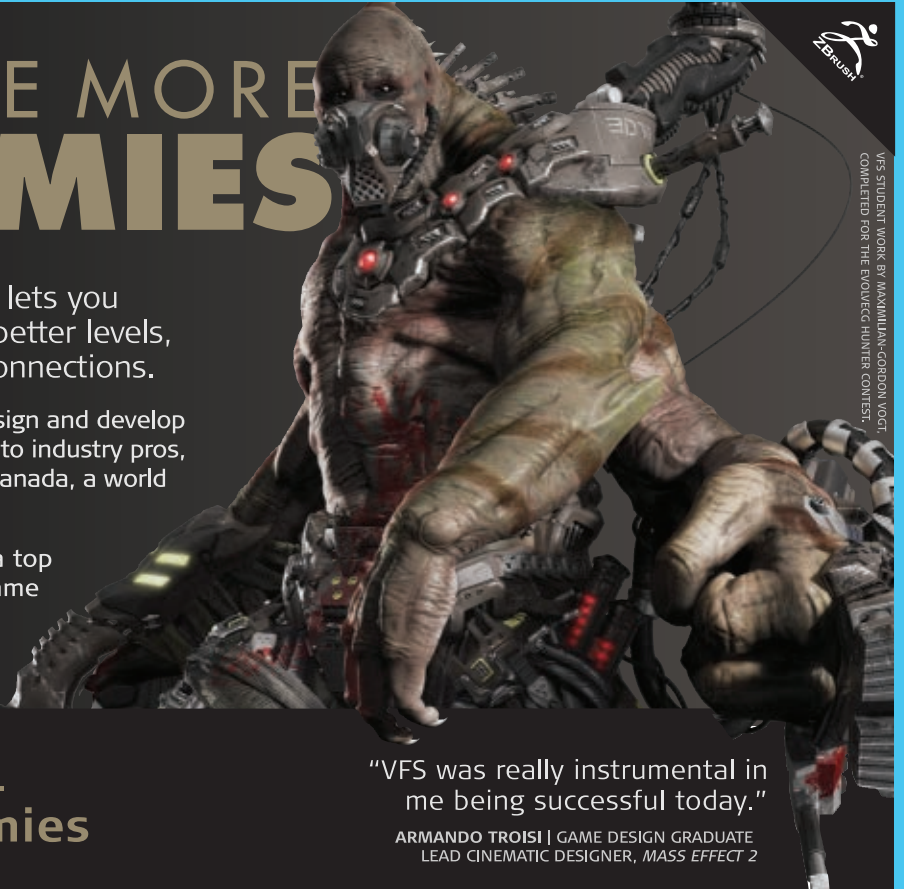
—Jeffrey Fleming

# MAKE MORE ENEMIES

Game Design at VFS lets you 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, Canada, a world hub of game development.

The LA Times named VFS a top school most favored by game industry recruiters.



VFS STUDENT WORK BY MAMMILIAN-GORDON VOIGT. COMPLETED FOR THE EVOLVEG HUNTER CONTEST.

VFS

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

"VFS was really instrumental in me being successful today."

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

## INVEST IN YOURSELF. BECOME A MASTER OF DIGITAL MEDIA.

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

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

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

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

CENTRE FOR  
DIGITAL MEDIA  
Great Northern Way Campus





# TURN YOUR PASSION FOR GAMING INTO A CAREER

## Game Art

Bachelor's Degree Program  
Campus & Online

## Game Design

Master's Degree Program  
Campus

## Game Development

Bachelor's Degree Program  
Campus

## Game Design

Bachelor's Degree Program  
Online



### Campus Degrees

#### Master's

Entertainment Business  
▶ Game Design

#### Bachelor's

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

#### Associate's

Graphic Design  
Recording Engineering

### Online Degrees

#### Master's

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

#### Bachelor's

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



**FULL SAIL**  
UNIVERSITY

[fullsail.edu](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

## ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
Epic Games	6	Ontario Media Development Corporation	3
Full Sail Real World Education	47	Neversoft Entertainment	44
IDG World Expo	40	Rad Game Tools	C4
IGDA	20	Unity Workshop	31
Masters of Digital Media	46	Vancouver Film School	46
NaturalMotion	C2		

*gd Game Developer* (ISSN 1073-922X) is published monthly by United Business Media LLC, 303 Second Street, Suite 900 South, South Tower, 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 *gd Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 838-7566. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *gd Game Developer* on any correspondence. All content, copyright *gd Game Developer* magazine/United Business Media LLC, unless otherwise indicated. Don't steal any of it.



## How to ANNOY YOUR PRODUCERS! *the complete guide!*

**TIRED OF THOSE PESKY PRODUCERS BOSSING YOU AROUND AND TRYING TO MAKE YOU MORE "EFFICIENT?" HATE MEETINGS? THIS SIMPLE GUIDE WILL KEEP YOUR PRODUCERS JUST AS UNHAPPY AS YOU ARE!**

### PHASE ONE

» Say, "More producers, huh?" every time a new producer is hired. Joke about whether they can get you a coffee. Ask them to send out a meeting request for you. Tell the long, convoluted story about the producer you had at your last job who was useless, and an alcoholic to boot.

Question the value of production methods to the game development process. Question the value of schedules. Question the value of producers. Talk about how Valve and Insomniac have no

producers, and don't they do pretty well? Talk about how back in the day teams used to be smaller, not like these huge teams we have today, and how those were the good old days. Back when there were no producers. Talk about how everything is better at the game studio across town.

### PHASE TWO

» Don't do your work. Do your work, but don't tell them you've done it. Do work that's not the work you agreed to do in the meeting. Get blocked and don't tell anyone. Get blocked

and tell the programmer, the tech artist, the animator, the environment artist, and the community manager, but not the producer. Do your work but don't check it in. When asked about it say, "Oh, that? It's been done for ages. I just haven't checked it in." Then continue to not check it in. When you do finally check it in, make passive-aggressive check-in comments like "not my idea," "whee!," or "asdfasdf;,"

Come into work late every day. Come into work sporadically. Work from home and promise you'll "be on e-mail all day," then don't answer e-mails. Go to lunch with the guys at noon and come back drunk at 3 PM. Estimate that you need just a couple more days every couple days. Look unconcerned about deadlines. Ask, "When's the next milestone again?" Then ask, "And what was I supposed to deliver for that milestone again?" Tell the producers not to worry and that everything will be fine.

Over-participate in networking events in your area, even the one the community college is holding about Web 2.0 startups. Over-participate on forums about game development, or games, or geek culture. Over-participate on the team spam alias. Send funny .gifs and cat memes in response to every e-mail you get, ever. Send spam-style e-mails to the full team list. Argue about politics and religion in long threads with your co-workers. Look up various historical events on Wikipedia to bolster those arguments. Tab out of WORLD OF WARCRAFT anytime someone walks behind you. Have BitTorrent running behind your main Maya window. Run an mp3 server on the lighting farm. File share your porn directory.

### PHASE THREE

» Be on a different page. Drill up. Get mushy. Talk about the beginning of the day. Say, "You know what? These Gantt charts are actually completely worthless." Say, "Well, you're the producer, so why don't you figure it out? I'm going back to my desk." Estimate in 20-day chunks. Estimate in 10-minute chunks. Say you can't estimate anything because the future is impossible to know and all predictions are doomed to failure.

Complain about the crunch food supplier of the evening. Say "Woo, pizza again!" in as sarcastic a voice as possible. After the food arrives, mention you are vegetarian. After the vegetarian food arrives, mention you are vegan. Ask for a lactose-free, gluten-free crunch meal option. Initiate crunch food eating contests. Accidentally drop the quart-sized styrofoam container of guacamole on the break room floor. Drop the quart-sized styrofoam container of mustard on the carpet.

Tell your producers your back hurts because of your chair and ask whether you can get a different one. Ask for one of those kneeling chairs. Ask for one of those standing desks. Say it's too cold because you're next to a vent. Say it's too hot because of your computers and dev kits. Say you are chilly when the person who shares the desk with you is sweating uncomfortably. Ask for different lighting conditions around your desk than everyone else's. Ask for a different gym membership deal than the one you currently don't use, and then never use the new one when you get it.

### FINAL PHASE

» Brainstorm new ideas in the scoping meeting. Don't make decisions. Change things back to the way they were after changing them away from what they were in order to "test out a theory." Bring up the idea that you "reserve the right" to reverse it the other way again later. Say it doesn't matter what we pick anyway, because the fans will eat it up no matter what we do.

Question choices that were made a long time ago. Bring up old points of contention and mention you were never "fully on board" with what was decided back then. Keep trying to steer the game back toward the direction everyone agreed they weren't going to follow. Two years into an action game project, say "Seriously, though, does it have to be an action game? I've got some great ideas for some turn-based mechanics ..."

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

ILLUSTRATION BY JUAN RAMIREZ





# GDC Europe

## Game Developers Conference™ Europe

August 15-17, 2011 | Cologne Congress-Centrum Ost | Cologne, Germany

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

Supported by



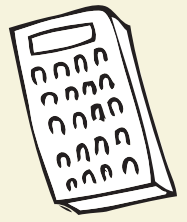
European  
Games Developer  
Federation



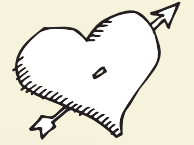
IF YOU WANT TO BE AN EVEN MORE

# AWESOME

I ♥



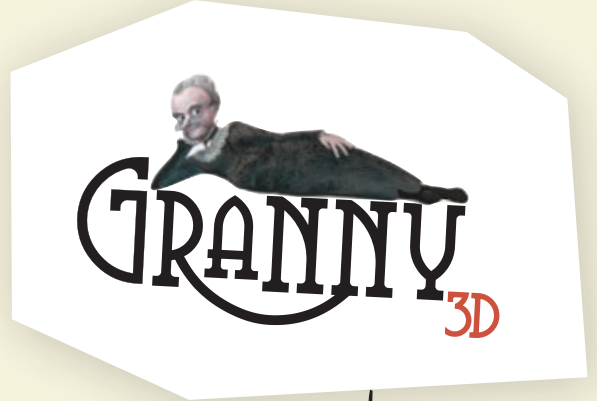
# GAME DEVELOPER



THEN YOU NEED TO BE USING

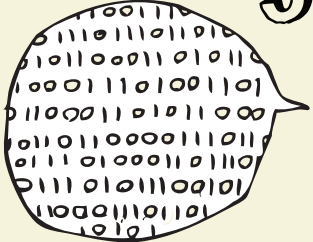
# Granny 3D

.....



You get a run-time dynamic

## 3D ANIMATION SYSTEM with



AMAZING content exporters

EASY data manipulation

and a brand **NEW** blend graph editor.

.....



# MUSTACHES



USING OUR TOOLS NOT ONLY MAKES YOU

MORE awesome, THEY MAKE YOU rad!



[www.radgametools.com](http://www.radgametools.com)  
(425) 893-4300