

RAY M A M

POSTMORTEM

ORIGIN 2

THE LEADING GAME INDUSTRY MAGAZINE VOL 19 NO 10
OCTOBER 2012 INSIDE: HOW TO SHUT DOWN YOUR DEV STUDIO

gd

GAME DEVELOPER MAGAZINE



EASY 3D with
stereographic
REPROJECTION





NOW HIRING

ART

Senior Character TDs / VFX Artists / Technical Artists

DESIGN

Lead Level Designers / Senior Systems Designers

ENGINEERING

Senior Graphics Engineers / Senior Backend Engineers
Senior Core Engineers / Senior Gameplay Engineers

Apply online at jobs.lucasfilm.com



LUCASARTS

A LUCASFILM COMPANY

© 2012 Lucasfilm Entertainment Company Ltd.
or Lucasfilm Ltd. & ® or TM as indicated.
All rights reserved.

postmortem

26 RAYMAN ORIGINS

How do you take a time-tested favorite like RAYMAN and make it fresh and fun all over again? In this month's featured postmortem, Ubisoft designer Chris McEntee talks about how a small team building new, artist-friendly developer tools ended up turning into the triple-A team that put RAYMAN back on the map. *By Chris McEntee*

features

7 SEEING DOUBLE

Adding stereoscopic 3D support to your game isn't necessarily a difficult thing to do, but doing it without having to render any given scene twice—once for each eye—is a bit trickier. Insomniac Games's Jim Van Verth explains how the stereographic reprojection system behind RATCHET & CLANK: ALL 4 ONE produced an impressive 3D effect without breaking the framerate budget. *By Jim Van Verth*

17 SIGGRAPH 2012 HIGHLIGHTS

The 2012 Siggraph conference featured new technologies and trends for current-generation game devs and more affordable development tools, which could offer new opportunities for studios large and small. *Game Developer* correspondents Carey Chico and Mike de la Flor give us this year's show floor highlights—and how they may shape your work in the future. *By Carey Chico & Mike de la Flor*

21 GAME OVER

Sometimes game studios have to shut down. Carey Chico walks us through his own experience closing his own development studio (Globex LA) and explains how you can make sure your own closing process—knock on wood—is handled responsibly, respectfully, and as gracefully as possible. *By Carey Chico*

departments

2 GAMEPLAN *By Brandon Sheffield*
Spam Me Not

[EDITORIAL]

4 HEADS UP DISPLAY *By Staff*
Correlating curse words with code, and the beauty of the game art asset.

[NEWS]

32 TOOLBOX *By Carey Chico*
3Dconnexion SpaceMouse Pro

[REVIEW]

34 INNER PRODUCT *By Bruce Dawson*
Floating-Point Misses

[PROGRAMMING]

41 GDC NEWS *By Staff*
Game Developer's Choice Online Awards Honor Raph Koster, WORLD OF WARCRAFT

[NEWS]

42 PIXEL PUSHER *By Steve Theodore*
R.I.P. Paul Steed

[ART]

46 DESIGN OF THE TIMES *By Damion Schubert*
Depth vs. Breadth

[DESIGN]

48 GOOD JOB *By Alexandra Hall*
Q&A with Adam Orth, new studios, and who went where

[CAREER]

49 BUSINESS *By Kim Pallister*
Crowdfunding and Emotional Equity

[BUSINESS]

50 AURAL FIXATION *By Yann Seznec*
Procedural Music

[SOUND]

52 EDUCATED PLAY *By Alexandra Hall*
REFRACTION

[EDUCATION]

56 ARRESTED DEVELOPMENT *By Matthew Wasteland*
The PR Coach

[HUMOR]



gd

GAME DEVELOPER MAGAZINE

CONTENTS.1012
VOLUME 19 NUMBER 10



SPAM ME NOT

SHOUTING ABOVE THE NOISE IN THE GAME WORLD

Notifications and cross-promotions are getting out of control. Facebook wants my attention, to tell me someone I've never met wants to be my "friend." Twitter wants me to know that someone "favorited" my tweet, and then emails me about that once per week even though I've asked it not to. That's just life in the digital age. But games have gotten particularly bad about this, especially in the free-to-play space. On the simplest level of spamming, you'll have to click through an ad for a game to get to the one you want to play. Sometimes the ad takes up most of the screen, but this is easy enough to ignore. Worse, I think, is when your iPad or iPhone wakes up to tell you it's time to restock TINY TOWER'S Luxury Cruise item in the travel shop. Four hours later, you'll do it again, not to mention all the quicker items, so maybe you need to actually keep those notifications on if you want to play the game properly. It quickly begins to feel like work—the mechanic of "waiting to click," and paying to avoid the wait, is pretty much the antithesis of fun. It's a compulsion more than a fun loop, and that's why it makes money.

Some of these games don't ever let you go. There's that little number notification on your smartphone app, telling you how many "things to click" you have waiting for you upon your return—but some games don't ever get rid of the "1," even if you've clicked everything. There's always at least one thing that needs clicking, so the game can constantly shout at you to return, because otherwise, why would you go back and click? How would you remember that clicking is a fun thing to do? The game needs to constantly be in your face, or else you'll forget how "fun" it is to touch a field of corn to harvest it. That is still the current face of free-to-play on smartphones and on the web.

Even inside our games we're getting an incredibly high ratio of noise to content. Why is this?

Obviously companies need to get the attention of players, and the pop-up ad is still in effective use across the Internet, compelling confused parents and homemakers to give their bank information to fake Nigerian princes. If, perchance, you do forget to click on a game for a while, the below image is the sort of thing you might see in-game:



The free-to-play business model was honed in Korea. I was just in Seoul last week pitching projects, and I noticed something interesting. I've become so conditioned to ignore advertising that it took me a few days to see it, but normal, everyday people in Seoul are basically living inside of a real-life pop-up ad.

There are some advertising-light areas, but anywhere you might pause, like subway stations, coffee shops, or gas stations, is a wall of advertisements. Even apartment complexes are branded by some business or other. For this Seoul trip, I stayed with some friends to save money. This is the first building you see upon leaving their subway exit.



And this is just the suburbs—45 minutes by train

away from the Gangnam district, the financial hub which most Korean game companies call home! People have to be so desensitized to advertising and visual noise just to live their normal lives, it's no wonder these game notification tactics I consider aggressive are par for the course in the country where the model was perfected.

The more desensitized one gets, the louder advertising has to get to grab your attention.

Many have said that the trend toward overnotification and underhanded tactics like always leaving a "1" hovering over the app is dissipating. I disagree. The more we see of these tactics, the more we come to get used to them. And that may very well mean the tactics will get even more nefarious. I believe we can do better, and high-end PC free-to-play is getting there. But there's a long road ahead, and as companies like Zynga move into gambling, I foresee even more clever uses of overnotification in our future.

As a final note, I'd like to thank everyone for reading *Game Developer* over the last several years. This is my 100th issue as an editor of this magazine, and I will now be officially stepping down as editor-in-chief, letting my able protégé Patrick Miller take up the reins of the magazine and this column. But I'll still have my own monthly column, fret not! It will be called Insert Credit—you can expect to read it next month!

—Brandon Sheffield
twitter: @necrossofty



UBM LLC.
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
www.gdmag.com/contactus

EDITORIAL

PUBLISHER

Simon Carless e: scarless@gdmag.com

EDITOR-IN-CHIEF

Brandon Sheffield e: bshffield@gdmag.com

EDITOR

Patrick Miller e: pmiller@gdmag.com

MANAGER, PRODUCTION

Dan Mallory e: dmallory@gdmag.com

ART DIRECTOR

Joseph Mitch e: jmitch@gdmag.com

CONTRIBUTING WRITERS

Jim Van Verth, Carey Chico, Mike De La Flor, Chris McEntee, Bruce Dawson, Steve Theodore, Damion Schubert, Alexandra Hall, Yann Seznec, Kim Pallister, Matthew Wasteland

ADVISORY BOARD

Mick West Independent
Brad Bulkley Microsoft
Clinton Keith Independent
Brenda Brathwaite Loot Drop
Bijan Forutanpour Sony Online Entertainment
Mark DeLoura THQ
Carey Chico Globex Studios
Mike Acton Insomniac

ADVERTISING SALES

GLOBAL SALES DIRECTOR

Aaron Murawski e: amurawski@ubm.com
t: 415.947.6227

MEDIA ACCOUNT MANAGER

Jennifer Sulik e: jennifer.sulik@ubm.com
t: 415.947.6227

GLOBAL ACCOUNT MANAGER, RECRUITMENT

Gina Gross e: gina.gross@ubm.com
t: 415.947.6241

GLOBAL ACCOUNT MANAGER, EDUCATION

Rafael Vallin e: rafael.vallin@ubm.com
t: 415.947.6223

ADVERTISING PRODUCTION

PRODUCTION MANAGER

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

REPRINTS

WRIGHT'S MEDIA

Jason Pampell e: jpampell@wrightsmedia.com
t: 877-652-5295

AUDIENCE DEVELOPMENT

AUDIENCE DEVELOPMENT MANAGER

Nancy Grant e: nancy.grant@ubm.com

LIST RENTAL

Peter Candito
Specialist Marketing Services
t: 631-787-3008 x 3020
e: petercan@SMS-Inc.com
ubm.sms-inc.com



UBM

by 2016 smart phone penetration in the UAE will be up **by 71%***



that's twice the growth of Europe and the USA.

twofour54° Abu Dhabi – the tax-free gateway to new mobile opportunities.

The MENA region is one of the world's fastest growing media and entertainment markets. And with 80% of under 25s owning mobile phones; mobile content is a prime opportunity for digital businesses. Over 100 leading media companies are already capitalising on the opportunity at **twofour54° Abu Dhabi**.

- 100% company ownership in a stable, tax-free environment
- Guidance and liaison with UAE content regulatory bodies
- Easy licensing and business set-up services
- Dedicated fund for mobile apps development via Apps Arabia™
- Unique campus environment with facilitated business networking

twofour54.com/mobile
+971 2 401 2454



twofour54
Abu Dhabi

media & entertainment hub

*Sources: Arab Media Outlook 2010. Media on the Move 2009. A.T. Kearney. Introduction to Gaming. Michael Moore. Screen Digest. IDC. Arab Advisors Group. ComScore 2011. Informa Telecoms and Media 2011

twofour54° is an initiative of the Abu Dhabi Government.



bad language

correlating curse words with code

Every programmer has gotten caught cursing at their code (or maybe a fellow programmer). But which languages inspire the most profanity? U.K.-based web developer Andrew Vos decided to settle the question by pulling just under one million commit messages from GitHub and correlating the presence of certain choice words to the corresponding project's programming language.

SHIT



F*CK



PISS



MOTHERF*CKER



Patrick Miller: Why'd you go through the Git commit messages and correlate profanity to programming language? Lazy Friday afternoon?

Andrew Vos: I hadn't written any code in about a week, and I really felt like writing something. I decided to rip a huge chunk of commit messages and maybe do some sort of statistical analysis on them. I was hanging around in the IRC channel #ruby-lang and someone may have mentioned

finding out how many swear words I could find.

PM: Can you elaborate on your methodology?

AV: I wrote a Ruby script to search for "shit," "f*ck," "piss," and "motherf*cker." For the stats to be valid, I had to rip an equal number of commit messages from each language.

PM: Out of almost one million commit messages, you only pulled about 250 or so with curse

words in them. Did that surprise you? (For comparison's sake, my emails have a much higher rate of profanity.)

AV: Yes, that was very surprising to me, but I think that because all these projects are open source, people tend to be more civilized. Some of the larger companies I have worked for have way more swear words in commits!

PM: Your results put C++, Ruby, and JavaScript on top, then Perl and C, then C# and Java, with Python and PHP at the bottom. Have you tried to draw any conclusions with the data? Does this jibe with your understanding of the respective languages (and perhaps the people who use them)?

AV: Seeing Ruby at the top was a huge surprise for me, because it's by far my favorite language. I can't believe there's so much hate for it, but I also think that Ruby developers tend to care less about being professional and are generally more fun. One of my most hated languages, PHP, has the least swear words! I definitely expected it to be at the top. I think this is because larger companies tend to use PHP more.

PM: You also posted the list of commit messages with swear words in them. Got any favorites

you'd like to share?

AV: These were my picks:

- *f*ck all the tests, you're going to have to live without them*
- *f*ck git*
- *chunk of shit i wrote over whiskey*
- *f*ck you and your rubygems*
- *Leave it to Chelimsky to f*ck up a simple name like Marston*

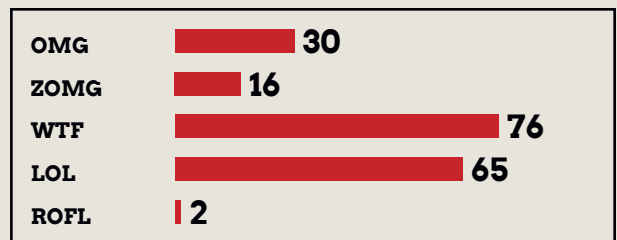
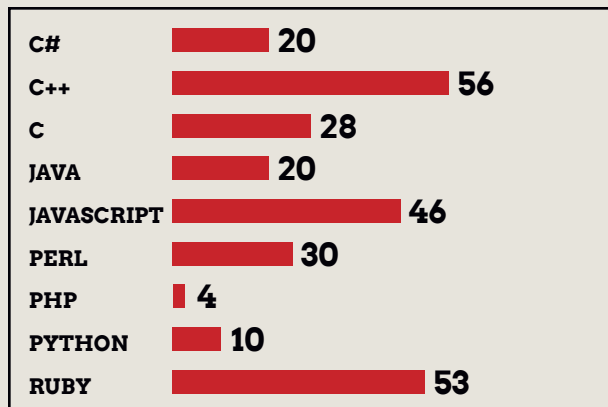
PM: "Zomg"? Weren't we done with "zomg" in 2005?

AV: Probably got to do with me hanging around Ruby developers too much.

PM: If you do the study again, is there anything you'd like to change about it?

AV: I would have loved to have access to all the commit messages, because the results do seem strange. Network speed was an issue when downloading all the commits. I would most probably have to spend a few weeks downloading commit messages if I wanted them all. It would be great if GitHub could give me access to their servers for a day!

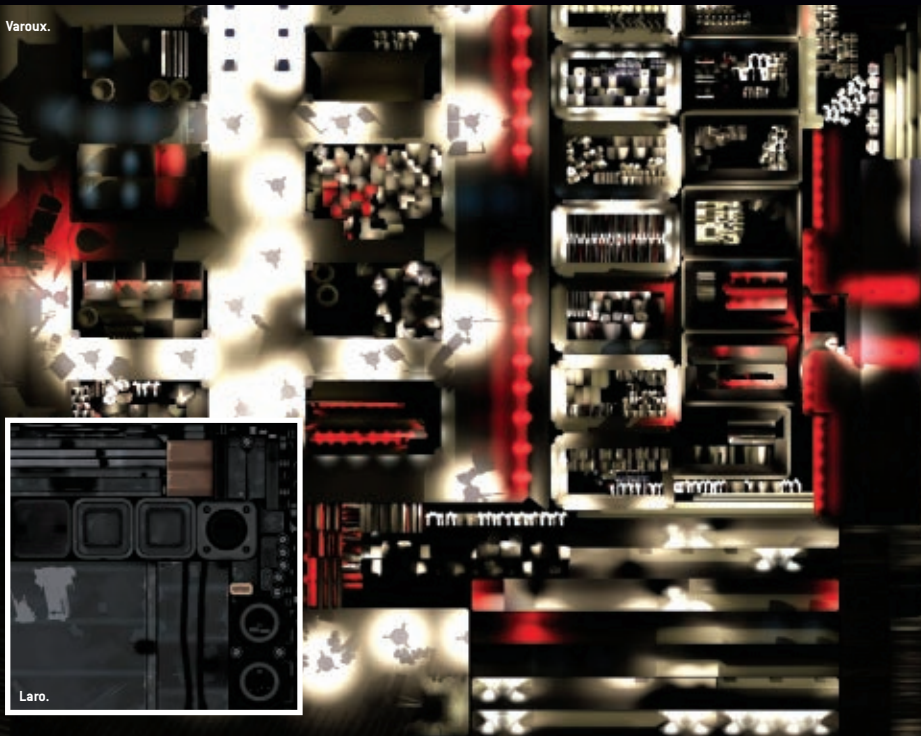
—Patrick Miller



Loving the bones

the beauty of the game asset

Varoux.



Laro.

we peel skyscrapers like apples.

We dice light into slivers and arrange those golden shards into vast mosaics; we transform time and shape into swirling color and use it to map river flows. When we make people, we chop off their faces, flatten and bake these human pelts, then tattoo makeup onto their unblinking eyes.

As game asset artists, we perform these countless miracles and atrocities every day. But when it comes time to curate the public face of game art, we hide that which imbues our real-time art with drama and magic. We're content to tape a dozen landscape paintings and portraits onto a wall, then point to that concept art and say that is game art, as we see in public exhibitions like *Into the Pixel*.

It's great to celebrate the visual arts in video games—but concept art is not what defines game art. It's not the actual thing that goes into the game. We tell people it's game art anyway, because concept art conforms to society's existing notions of paintings as art. This is the aesthetic equivalent of easy mode; this is lazy.

Let's try a higher difficulty setting, a more playful and complex mode of thought. Let's try to understand and promote the real-time art asset as its own medium of visual art. I will focus on the game texture—specifically, these three masterworks, coming soon to a museum near you.

If you've ever found a new hat or weapon in *TEAM FORTRESS 2*, there's a good chance Rob Laro painted it. Note the very

efficient pack ratio on his texture flat for the Black Box tank-buster rocket launcher, how very little texture space is wasted, yet the overall texel resolution is evenly distributed. Here, Laro has skillfully painted a fantastic gunmetal, a very narrow range of matte mid-gray values that still provides a great sense of contrast in the material. This piece is a classic in the way most industry game artists might appreciate a flat surface; it is topologically elegant and demonstrates masterful technique.

We can also think of textures as more conceptual pieces. What if you produced flat paintings without actually texturing anything, or what if you never saw the corresponding model? Conceptually, then, is

it still a texture? Consider this lightmap by Thomas Varoux, with its countless halos. You might imagine it lighting a space station at twilight, or a vast Himalayan palace at dusk. You imagine invisible models—impossible models with non-manifold geometries. When you can see only the shadow of a world, its shape is infinite.

If you did not know what a lightmap was, then it would cease to represent light. This character sprite sheet by Anna Anthropy, viewed in a similar way, might cease to represent a walk cycle. Andy Warhol famously painted a grid of Campbell's soup cans, reducing their individual characteristics to the sum of an abstract pattern; the sprite sheet, then, could reduce each frame's differences and emphasize the common iconography. Does each frame depict the same character in different poses and perspectives, or is each frame a separate character?

Or maybe Warhol just liked soup, and we just make assets that appear in entertainment products and that's all there is to it. You can think like that, and you wouldn't be wrong. But you might end up bored and stuck on easy mode, and being bored is worse than being wrong.

—Robert Yang



Anthropy.



BILZARD

ENTERTAINMENT
UNIVERSITY RELATIONS

DO YOU HAVE THE PASSION TO CREATE
AND THE WILL TO FORGE GREAT GAMES?

If you have a passion for games and strive for excellence, we encourage you to check out these opportunities to join Blizzard Entertainment and contribute to the most epic entertainment experiences... ever!

INTERNSHIPS

Continuing students from all over the country are welcome to apply for our Summer Internship program! Work directly with development teams and business operations departments to gain hands-on experience!

For more information, visit
ur.blizzard.com

FULLTIME OPPORTUNITIES

We're always looking for top talent! If you're a recent graduate, have a passion for games, and aspire to work for Blizzard Entertainment, we may have the right opportunity for you!

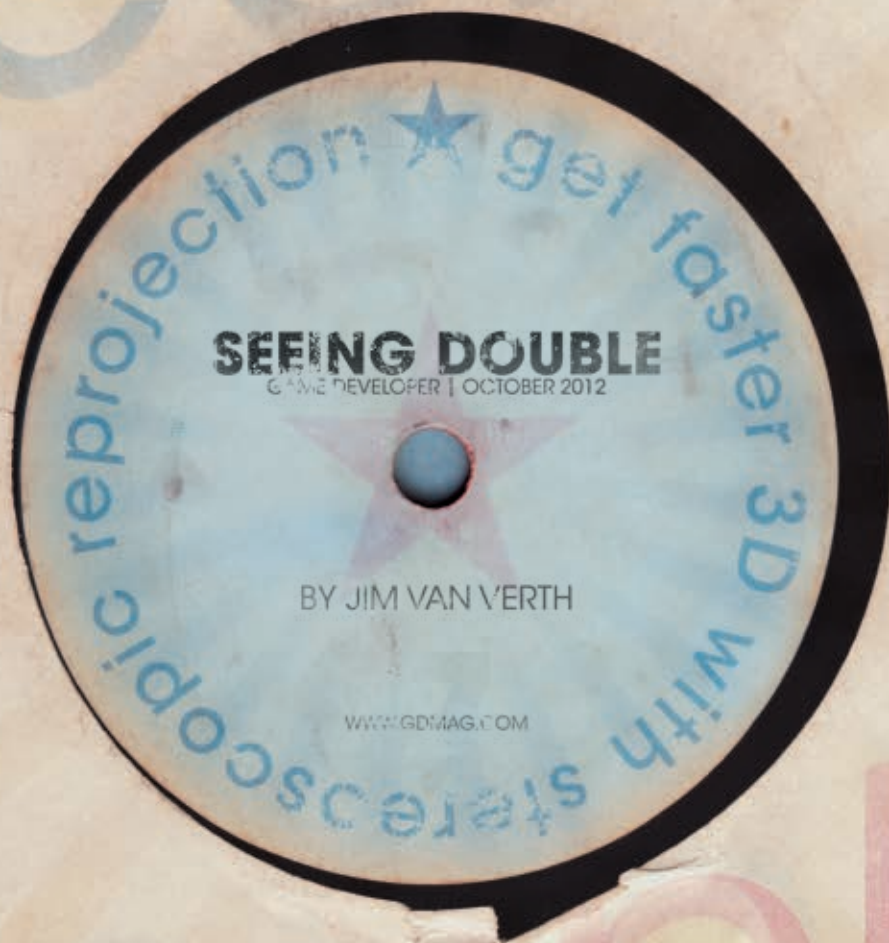
For more information, visit
jobs.blizzard.com

Dedicated to creating the most epic entertainment experiences... ever.

DIABLO

WORLD
WARCRAFT

STARCRRAFT



☐ There are plenty of ways to add stereoscopic 3D to your game. The hard part is doing it without having to fully render any given scene twice and killing your frame rate. In this article, I'll explain how we designed RATCHET & CLANK: ALL 4 ONE's stereo reprojection system to produce attractive 3D visuals without sacrificing performance. I first became aware of stereo reprojection when Crytek announced stereo support for CRYISIS 2. At the time, we at Insomniac were just beginning to plan stereo support for RATCHET & CLANK: ALL 4 ONE, and the prospect of having a stereo solution that added very little to frame time was quite appealing. The frustrating bit for us was that Crytek gave only the slightest of hints as to how its implementation worked. Clearly it used a pixel shader to gather pixels, and there was a tantalizing mention of bilinear filtering, but not a lot of details. Thus began our quest to make our own version of stereo reprojection. While I did not quite get the performance Crytek described, we delivered what we needed to ship the game, we added some new features, and our results look quite good. >>>

WHY WE COULDN'T USE STANDARD STEREOGRAPHIC PROJECTION

Before looking at our stereo reprojection solution, let's quickly review how the standard stereographic method works. The idea is that we are trying to provide to each eye (via shutter glasses, or polarization, anaglyph color, or head-mounted display) a view from that particular eye's perspective.

There are two pieces here: First, each eye is displaced from the center by a certain distance, called the interocular distance. It's this displacement that allows stereo viewing in the first place. Secondly, depending on what we're focusing on, the eyes will toe in or out when we look at a near or far object, respectively (see **Figure 1**).

The interocular distance is about 5–8cm, measured from the center of each pupil. When working with virtual cameras, we often refer to the interaxial distance instead. Because of toe-in, there will be a plane where points on that plane will project to the same screen position in both views. We call this the convergence plane, and the distance to that plane is known as the convergence distance. Points between the view points and the convergence plane will have negative parallax (and will appear to stick out of the screen), and points behind the convergence plane will have positive parallax. Note that the units for these distances could change, depending on whether you are measuring in

view space or in NDC (Normalized Device Coordinates) space.

When projecting for stereo, we add two additional transformations to represent the interaxial and convergence: an **xz**-shear **S** and an **x**-translation **T**. Some implementations incorporate the translation into the view matrix and the shear into the projection matrix, but I find it more compact to add both to the projection matrix. The end result can be seen in **Figure 2**, showing both standard D3D and OpenGL perspective projection matrices.

$$M_{D3D} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & da & 0 & 0 \\ \pm S & 0 & Q & 1 \\ \pm dT & 0 & -Qn & 0 \end{bmatrix}$$

$$M_{GL} = \begin{bmatrix} d & 0 & \pm S & \pm dT \\ 0 & da & 0 & 0 \\ 0 & 0 & R & (R-1)n \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

where

$$Q = \frac{f}{c}$$

$$R = \frac{a}{c}$$

$$d = \cot(\text{fov}/2)$$

and

- a – horizontal aspect ratio
- fov – horizontal field of view
- n – near-plane distance
- f – far-plane distance

FIGURE 2: Standard stereoscopic projection matrices.

Note that the shear and translation have opposite signs (they end up with the same sign in the OpenGL projection matrix because the shear gets multiplied by the —1

in the bottom row), and each term has equal magnitudes but opposite signs for each view. The left view will have negative shear and positive translation, while the right view has positive shear and negative translation.

So what do we choose for the magnitude of **S** and **T**? They need to be related; otherwise you will end up with bad convergence and headaches for your viewers. Other things that need to be taken into consideration are your monitor/television size and your horizontal field of view. A larger monitor implies that you will be sitting farther from it, which implies that less shear is required. And the convergence distance in turn is dependent on the field of view.

The set of equations that we used in ALL 4 ONE can be found in **Figure 3**, partially derived from Samuel Gateau's GDC 2009 talk on stereoscopic 3D [see **Reference 1**] and the Sony documentation. As we can see, **S** is dependent on the interocular distance and monitor width, and **T** is dependent on **S**, convergence distance, and field of view. This is a bit counterintuitive, but it works quite well.

$$\|S\| = l/w$$

$$\|T\| = \|S\|c \tan(\text{fov}/2)$$

where

- l – interocular distance
- w – real-world monitor width
- c – convergence distance
- fov – horizontal field of view

FIGURE 3: Stereoscopic parameters.

The process at this point is quite simple: Render two views, one for each eye, and send each image to the display. Other than setting up the projection matrix, this doesn't take much work, and in fact some stereo drivers take advantage of this simplicity to automatically provide stereo for games. However, it does have one disadvantage: You do have to render the scene twice. We could optimize this a bit, of course; we could do all nonrendered processing first, cull for the convex hull of both frustums rather than one at a time, or generate a single pushbuffer and only modify the projection matrices—but in the end

you still need to pass data through the GPU twice, which takes time. In our case, we were already up against the 30 fps boundary, so we had very little extra rendering time.

OUR SOLUTION: STEREO REPROJECTION

The alternative to rendering everything twice is to render color and depth information for a single view, then use the information for that view to generate a stereo pair—rendering the left view and using it to create the right view, for example. Because the original image will not have the correct projection for at least one of the views, we will have to remap the color data to match the new view, or perform reprojection. For the sake of the rest of the article we'll assume that unless stated otherwise, we'll be reprojecting from a left view to a right one.

We will need to have access to the color and depth images to do this. If you're rendering into a backbuffer (or don't have access to a console that lets you just dip into memory and point a new texture to it), you'll need to change your rendering approach. Fortunately, if you're using something like deferred rendering or deferred lighting, you'll have already done this and will have textures available with this information. Otherwise, you'll have to use render targets in Direct3D or framebuffer objects in OpenGL for both color and depth.

While CRYISIS 2 was the first notable use of stereo reprojection in games, it is not a new technology. Erik Bernald described a similar approach on his blog in 2008 [see **Reference 2**]. Other reprojection approaches have used scatter-based splatting or image warping using meshes [see **Reference 3**]. The technique I'm presenting here is a gather approach derived from parallax occlusion mapping, which takes a color image, a depth image, and uses raycasting to create the appearance of geometry displacement within a shader.

A full description of parallax mapping is outside the purview of this article. You can find more

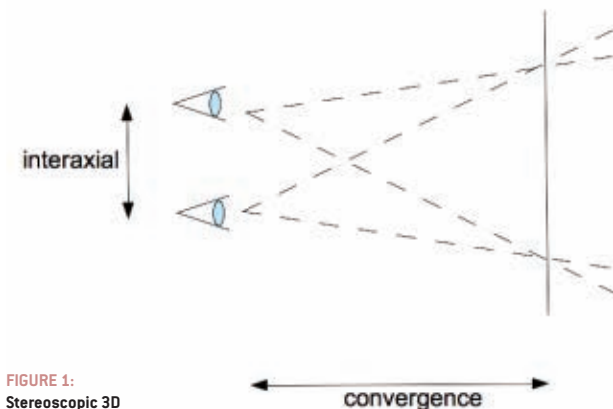


FIGURE 1: Stereoscopic 3D frustum setup.

LISTING 1

The core of the fragment shader (using GLSL).

```

UNIFORM SAMPLER2D COLOR_TEXTURE;
UNIFORM SAMPLER2D DEPTH_TEXTURE;
UNIFORM VEC4 STEREO_PARAMS;
VARYING VEC2 UV;
VOID MAIN()
{
    VEC2 OUT_TEX_COORDS = UV - VEC2(STEREO_PARAMS.X, 0.0);
    VEC2 REPROJ_TEX_COORDS = UV;
    FLOAT DIR = SIGN(STEREO_PARAMS.Y);
    FLOAT ORIG_X = UV.X;
    FLOAT SHIFT = STEREO_PARAMS.Z;
    FLOAT STEP = 0.0625*STEREO_PARAMS.W;
    INT INDEX = 0;
    WHILE (INDEX <= 16)
    {
        REPROJ_TEX_COORDS.X = ORIG_X + SHIFT;
        FLOAT L = GETLINEARDEPTH(DEPTH_TEXTURE, REPROJ_TEX_
COORDS);
        FLOAT DEPTH_ADJUST = STEREO_PARAMS.X + STEREO_PARAMS.Y
/ L;
        FLOAT TEST = SHIFT + DEPTH_ADJUST;
        IF (DIR*TEST >= 0.0)
        {
            OUT_TEX_COORDS.X = ORIG_X - DEPTH_ADJUST;
            BREAK;
        }
        SHIFT = SHIFT + STEP; INDEX = INDEX + 1;
    }
    GL_FRAGCOLOR = TEXTURE2D(COLOR_TEXTURE, OUT_TEX_COORDS);
}

```

information in Brawley and Tatarchuk's original ShaderX article [see [Reference 4](#)]. However, the general idea is that when you render the surface you wish to apply parallax mapping to, for each pixel you cast a line segment along the view direction at progressive depths into the depth map. When the line segment has penetrated

the depth map, you use that penetration point to determine the color seen from that view direction.

Stereo reprojection, at its heart, is essentially the same process. We have a color image and its corresponding depth information [albeit using a perspective projection rather than an orthographic one], and

we have a new view of the same scene we wish to render. We could just use the same algorithm by progressively casting line segments per pixel to generate the new view. However, because of the nature of stereo reprojection, we can simplify things greatly. **Figure 4** shows the situation for a desired location in the right eye view (in green). We create sample points along a ray from the right viewpoint through the view plane (in this case, the convergence plane) location for our current pixel. Since the view separation is only in the x -direction, we end up only casting the ray along x and z . The y -value of the ray will always be 0.

For the sake of clarity, we start our sampling in **Figure 4** at the convergence plane, though we usually will want to start between the eye point and the convergence plane to get some negative parallax. For each sample point (in yellow), we cast a new ray from the left eye, determine the view plane position for that ray, and look up the depth by mapping that view plane position to texture space. If this depth is less than the depth of the sample point, we stop and use an interpolation of the sample point's depth and left eye depth to get our final depth value, which we use to reproject from the right viewpoint to get our final color.

However, because we are only searching in the xz directions, we can think of this in a different way—see **Figure 5**. In this case, we know the corresponding view plane position in the left view will have the

same y value as our desired view plane location, so we'll just search a range of candidate screen positions P^* along a line of constant y . In this case we start with our pixel's position from the right eye viewpoint (in green) and iterate in the $-x$ direction (to yellow and orange). So, rather than casting a ray and then computing a view plane position to look up a depth value, we'll just use the depth at the current P^* to reproject into the right eye view. If its position is close to our desired location then we'll stop. We'll use that depth as our final depth to look up the color value for our pixel.

This raises the question of how we perform our reprojection—namely, how do we use a view plane position P_{left} in the left view and the depth to determine the corresponding view plane position P_{right} in the right view? We can simplify the problem down to one of finding the displacement along the convergence plane, between the left view and the right view (see **Figure 6**). This is a case of computing a ratio of similar triangles: The distance between P_{left} and P_{right} is to the distance between the two view positions as the distance from the convergence plane is to the distance from the view positions. This gives us the final equations in **Figure 6**.

Since this is using a ratio of distances, this will work just as well whether we're in NDC space or texture space, but it assumes that the depth we have is linear. However, if we use standard perspective projection, the depth

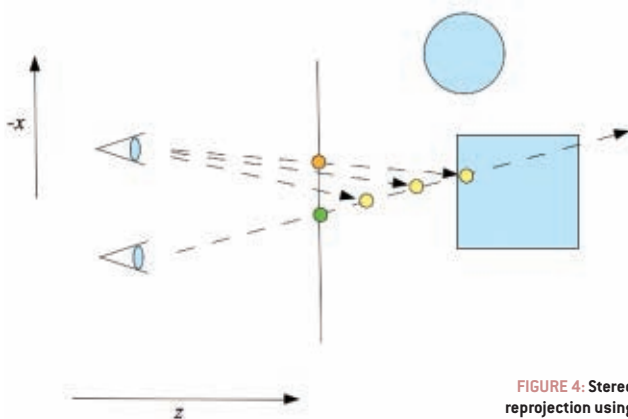


FIGURE 4: Stereo reprojection using ray-casting.

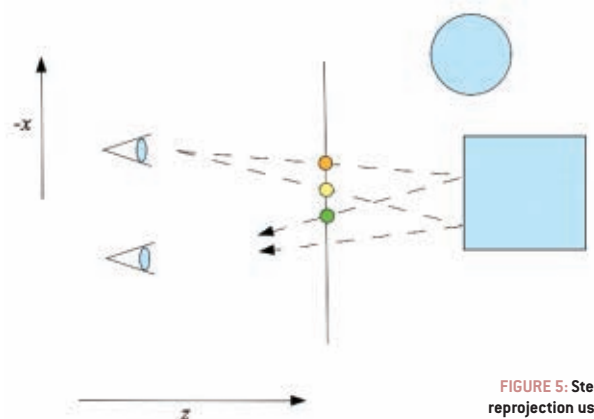


FIGURE 5: Stereo reprojection using pixel match.



GAME DEVELOPERS CONFERENCE™ CHINA

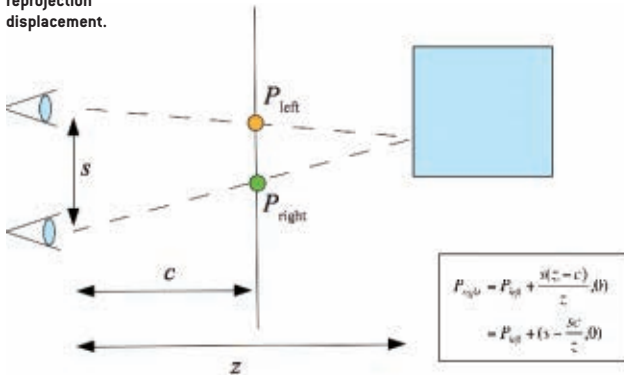
SHANGHAI, CHINA
SHANGHAI INTERNATIONAL CONVENTION CENTER
NOVEMBER 17-19, 2012

2012

www.GDCCHINA.com



FIGURE 6: Computing reprojection displacement.



values in the buffer are nonlinear. Under normal circumstances, ALL 4 ONE'S engine provides a second floating-point depth buffer which is linear (it's used for computing SSAO, among other things), but because the stereo buffers require more VRAM, activating stereo disables SSAO and removes this extra depth buffer. We solved this by using a scratch area of VRAM to compute the floating-point buffer just before applying the reprojection. Note that you do need a certain amount of resolution in your depth, otherwise you may get strange striations across your stereo images as you suddenly step from one depth range to another.

Summarizing the algorithm:

- ☞ For each **P_{right}** [i.e., the pixel we're considering in the fragment shader]:
- ☞ Iterate along values of **P*** for the left view
- ☞ Reproject to get a candidate location
- ☞ If it is past **P_{right}**, we have a hit
- ☞ Use the color values for our current and previous **P*** to set our final color

This search strategy raises a few questions. First, what is the range of the search for **P***, and what step size should we use? As **z** approaches infinity, the magnitude of the displacement between **P_{left}** and **P_{right}** approaches **s**, which means the maximum possible value of separation for positive parallax will be **s**. So for my initial algorithm I chose a range of texels

in the source view from **P_{right}-s** to **P_{right}+s**, and stepped one texel at a time. This provided a certain amount of negative parallax without unduly increasing the search. However, by stepping one texel at a time, the number of texels we search will increase as **s** increases. Since **s** is one of the knobs we can turn to improve our stereo result, we will either have poor performance, or are limited to fairly flat-looking images.

We derived a better approach from parallax mapping. In my final revision, I chose a constant number of iterations, and split the search space into a fixed number of samples. In the parallax mapping case, the samples are along the search ray; in our case, along the scanline. One approach (see **Reference 5**) uses a single sample at the farthest depth, but I didn't get very good results with it. Instead, 16 samples presented the best trade-off between quality and speed. Reducing the amount of negative parallax by only searching between **P_{right}-s/2** to **P_{right}+s** also helped the quality.

As far as the final color value, for the initial algorithm I used a linear interpolation of the current and previous texel colors, based on the previous and current screen space positions. This made for muddy edges, particularly when an edge switches from a near object to a far-distance object, or vice versa. In parallax mapping, as mentioned, you would normally take a blend of the ray sample point's depth and the depth from depth map lookup, and then use that to reproject



FIGURE 7: Right-eye views of Ratchet, showing a) ideal stereo, b) reprojecting with the wrong search direction, and c) reprojecting with the correct direction.

from the right eye view position. This final reprojection point is used to look up the color value, in this case, in the left eye image. However, I discovered a paper by van de Hoef and Zalmstra (see **Reference 5**), two students from

the University of Utrecht, who were also trying to create fast reprojection. They skipped the blend and just used the depth map lookup alone for the final reprojection. This is faster than the lerp, matches reasonably well,

and gets rid of the smeared, blurry effect on near-to-far edges.

The final issue is to decide which direction to search along the scanline—the positive x direction or the negative x direction? I've given away the answer already, but in my initial algorithm I unthinkingly searched in the positive x direction, which had some unexpected effects

The core of the fragment shader (using GLSL in this case, but the Cg or HLSL shader would be equivalent) is in **Listing 1**.

In this case, we've replaced the P_{right} end condition with a check against zero to avoid an unnecessary addition, but the same principle applies. The constant stereo_params contains $-/+s$, $+/-sc$, the

egregious artifacts from the previous method, but it still has a few issues.

The first issue is that we are limited in the amount of negative parallax we can represent. Recall that negative parallax occurs when objects are closer to the viewpoint than the convergence plane, and it shows up visually as objects

▼▼ In our case we simply repeat the foreground or background information and hope that the area is small enough that the players don't notice. Some ways of reducing the area are to again bring in the convergence plane (which does reduce the stereo effect, but helps with the errors) or to render the scene from a camera with a standard frustum, and then reproject twice to generate the left and right view from that. ▼▼

on the result. **Figure 7a** shows the standard stereo projection, while **Figure 7b** shows the result of this reprojection algorithm, and the smeared ruin of poor Ratchet's face. Tragedy and despair.

This is a problem because by iterating in the positive x direction when doing left-to-right reprojection, we're moving in the wrong direction along the view ray. Rather than searching deeper and deeper into the screen, we were starting with the maximum depth value for our ray, and searching more and more shallowly. This leads to an early exit from the search, and Ratchet's poor deformed face. Fortunately the solution is simple: Reverse the search direction! **Figure 7c** shows this far better result.

starting offset for the search, and the search range. **GetLinearDepth()** converts the nonlinear depth value from the texture into a single floating-point linear value.

This shader has the advantage in that it works both for reprojecting the left view to the right one, and the right view to the left one, simply by changing the stereo parameter constant. However, if you only want to reproject from left to right, for example, you can simplify it by removing the "dir" term and adjusting signs accordingly.

PROBLEMS WITH REPROJECTION

☞ Overall, our approach works quite well and removes the more

popping out of the screen. For aesthetic purposes, we usually don't want a lot of negative parallax because if it occurs near the edge of the screen it can destroy the 3D illusion (because one eye can see the object, while in the other it appears chopped off). But it's difficult to avoid entirely, so ideally our algorithm needs to handle it.

While we know that the maximum displacement due to positive parallax is s , for negative parallax there is no practical maximum. Let's look at **Figure 8** to see why this is the case.

Here we see an object with negative parallax. The distance between the left and right view ray intersections is clearly larger than s , so we could broaden our search to include that. However, as objects get closer and closer to the eye points, that displacement gets larger and larger, approaching infinity. Clearly capturing all negative parallax is not practical, so the only thing we can do is set a maximum value, and try to keep our objects within that amount of negative parallax. If we end up with a situation where objects might fall beyond that range, we can bring in the convergence plane to reduce the parallax.

Another set of issues crops up because the new viewpoint can see parts of objects that the original can't. These are known as reconstruction issues. The original problems with Ratchet's face were an extreme example of this, with some very bad reconstruction. However, even with an improved

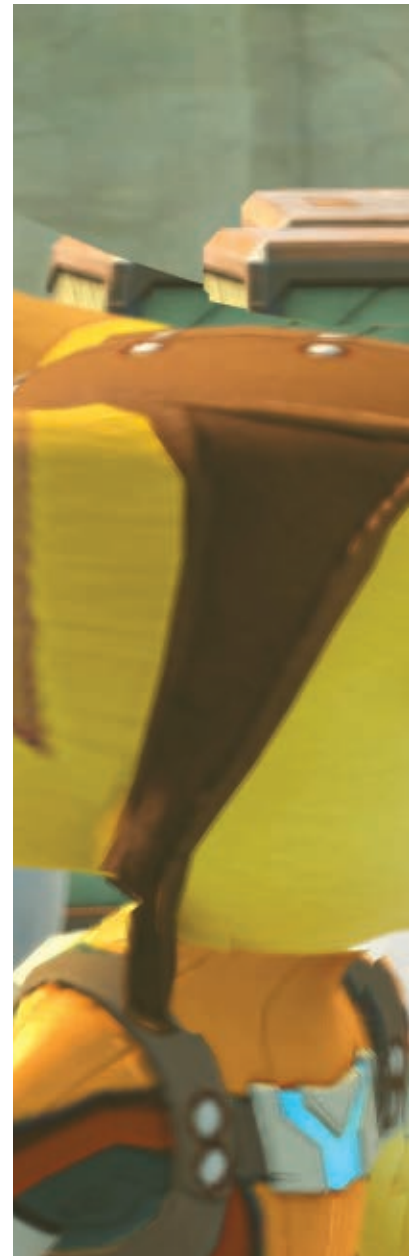
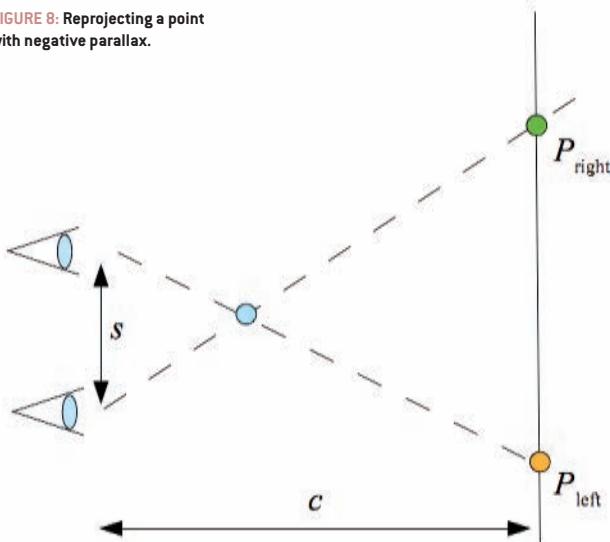


FIGURE 8: Reprojecting a point with negative parallax.



algorithm it is impossible to perfectly reconstruct such areas, because the information simply isn't there (which we can see in **Figure 7c**, to the right of Ratchet's face). In our case we simply repeat the foreground or background information and hope that the area is small enough that the players don't notice. Some ways of reducing the area are to again bring in the convergence plane (which does reduce the stereo effect, but helps with the errors) or to



render the scene from a camera with a standard frustum, and then reproject twice to generate the left and right view from that. The reprojection step does take twice as long when we do that, but the results look significantly better, and it's still faster than rendering the scene twice. **Figure 9** shows the result of this, with the left side rendered with left-to-right reprojection, and the right side with a standard non-stereo view reprojected to both left and right.

A third issue is related to reconstruction. If we look at **Figure 1** again, we can see that there are regions in space that the left eye can see that the right one can't, and vice versa. So we have no information on how to reconstruct that region from the other view. One possibility is to place black bars to cover that area of the viewport, but that's not very appealing. Our final solution was to expand the horizontal field of view so that region is outside of the desired viewport, then after reprojection

crop down to the new shared viewport area.

The final set of issues is due to how we handle alpha-blended objects. In the ALL 4 ONE engine, we have a separate pass for such objects, sorted by depth, and we do not write out depth. This produces good results for transparency but has the slight problem that we don't have the depth information to correctly reproject that object. The end result is that alpha-blended geometry looks like it's painted onto

whatever is behind it, which made all our effects look quite strange. We tried writing depth with alpha, but that introduced a new set of problems. In the end, the best way to handle both the alpha-blending and the 3D stereo was to use reprojection to render the opaque passes, and then render the alpha passes using a standard stereo projection.

In order to take that approach, though, we have to do two things. First, we have to reproject depth values as well as color values, so

Using the D3D-style stereo projection matrix and projecting a point into the left view gives us:

$$\begin{bmatrix} d & 0 & 0 & 0 \\ 0 & ad & 0 & 0 \\ S & 0 & Q & 1 \\ dT & 0 & Qn & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} dx Sz + dT & ady & Qz & Qn \\ & & & z \end{bmatrix}$$

After the reciprocal divide and considering only the x values, we end up with:

$$\begin{aligned} x_{left} &= \frac{dx}{z} - S + \frac{dT}{z} \\ &= x_{center} - S + \frac{dT}{z} \end{aligned}$$

Where x_{center} is the result of non-stereo projection. Expanding and simplifying, we end up with:

$$\begin{aligned} x_{left} &= x_{center} - S + \frac{dT}{z} \\ &= x_{center} - S + \frac{dSc \cdot \tan(fov / 2)}{z} \\ &= x_{center} - S + \frac{Sc}{z} \end{aligned}$$

(Recall that d is $\cot(fov/2)$). Similarly, for x_{right} :

$$x_{right} = x_{center} + S - \frac{Sc}{z}$$

Subtracting these gives the displacement between the left and right views in NDC space:

$$\begin{aligned} x_{right} - x_{left} &= x_{center} + S - \frac{Sc}{z} - x_{center} + S - \frac{Sc}{z} \\ &= 2S - \frac{2Sc}{z} \end{aligned}$$

However, NDC space has a range from -1 to 1. To convert this to a displacement for texture coordinates, which range from 0 to 1, we halve the distance, which gives us:

$$x_{right} = x_{left} + S - \frac{Sc}{z}$$

If we replace S with s , this is clearly the same as the equation in Figure 6, so s is just S when reprojecting from a left eye view to a right eye view. Note that for center reprojection, we would split the difference going from center to left and center to right, and so halve the displacement.

Again, signs are important for computing this: for reprojecting and rendering the left eye, we'd use $-s$, $+sc/z$, $-S$, and $+T$. For the right eye, we invert the signs and so use $+s$, $-sc/z$, $+S$, and $-T$.

The end result is quite nice. We get most of the speed benefits of reprojection, with the good-looking alpha of standard stereo, and all it took was a little bit of math and some extra setup.

that the alpha-blended objects will have the correct depth occlusion. Secondly, we must compute standard stereo projection matrices that will match up with our reprojected result.

MATCHING STANDARD AND REPROJECTION STEREO

Remember that we have two variables for our standard stereo: S and T , which are

primarily dependent on our convergence plane distance c and our interocular distance i . In reprojection we have the same convergence plane distance, but we have a different value s that probably bears some relation to the interocular distance. However, it's unclear what that would be to match the standard stereo projection. To determine that, let's project a point into left and right NDC space and measure the delta between them in the x direction.

DIY 3D

That's the whirlwind tour of our implementation of 3D stereoscopy in RATCHET & CLANK: ALL 4 ONE. I hope this gives you some direction in writing your own version of 3D stereoscopic reprojection, no matter the platform. To give you a starting point, I've created an OpenGL demo demonstrating the basic techniques, which you can find at <http://gdmag.com/resources/>

references

- [1] Gateau, Samuel. "The In and Out: Making Games Play Right with Stereoscopic 3D Technologies." GDC 2009, <http://developer.download.nvidia.com/presentations/2009/GDC/GDC09-3DVision-The In and Out.pdf>
- [2] Benerdal, Erik. "Generating stereoscopic images with parallax occlusion mapping," www.scalarinet.net/2008/04/22/generating-stereoscopic-images-with-parallax-occlusion-mapping
- [3] Didyk, Peter, Tobias Ritschel, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. "Adaptive Image-space Stereo View Synthesis." In *Vision, Modeling, and Visualization* (2010), Reinhard Koch, Andreas Kolb, Christof Rezk-Salama (Eds.)
- [4] Brawley, Z., and Tatarchuk, N. 2004. "Parallax Occlusion Mapping: Self-Shadowing, Perspective-Correct Bump Mapping Using Reverse Height Map Tracing." In *ShaderX3: Advanced Rendering with DirectX and OpenGL*, Engel, W., Ed., Charles River Media, pp. 135-154.
- [5] van de Hoef, Marries, and Bas Zalmstra. "Fast Gather-based Construction of Stereoscopic Images Using Reprojection." Utrecht University, 2011.



code.php. I'm looking forward to seeing how you improve upon it in your future games! ☺

JIM VAN VERTH has been worked in the games industry since 1996, and is currently a senior engine programmer at Insomniac Games. He's also co-author of the book *Essential Mathematics for Games and Interactive Applications*. You can email him at jim@essentialmath.com.

RATCHET AND CLANK images courtesy/ copyright SCEA 2012



UNREAL ENGINE NEWS

Arkane Studios Reimagines First-Person With *Dishonored*

Arkane Studios has been utilizing Unreal Engine 3 (UE3) to let their imaginations run wild in crafting a world that looks unlike any game ever seen before. *Dishonored*, the new first-person perspective action adventure game, is set in the industrial whaling city of Dunwall. The world has a painterly style with a 19th century London feel blended with steampunk, fantasy and science fiction influences. The teams in Austin, Texas and Lyon, France have been using UE3 technology to further push the envelope of non-linear exploration.

"We love first-person action games with aspects of RPGs or adventure games added in," said Raphael Colantonio, co-creative director on *Dishonored* at Arkane Studios. "There's a particular type of game that is a blend of genres that we've always strived to create: immersive, first-person, with resource economics; a cohesive, plausible sense of place; variable approaches and outcomes that are presented in a consistent way, often relying on simulation, so the player can formulate and execute strategic plans in how they undertake encounters."

Dishonored gameplay blends combat, stealth, and role-playing features. Gamers take on the role of a supernatural assassin in a retro-future world. The game mechanics can be applied creatively, as the developer is encouraging gamers to play their way.

According to Harvey Smith, co-creative director on *Dishonored* at Arkane Studios, there remains confusion over the types of games Arkane has

worked on in the past such as *Arx Fatalis*, *Dark Messiah of Might & Magic* and *Deus Ex: Invisible War*.

"We've never worked on open-world games, but the confusion exists because our games – while mission-based – feel very nonlinear and open-ended in very specific ways," explained Smith. "There are multiple pathways through the environment. Sometimes these are just realistic routes, like rooftops and windows, and at other times these pathways are enabled by the player's choice of powers, play style and morality. We want to allow the player many choices so they can author their own experiences."

Arkane had a choice in technology when it came time to bring their vision to life. Colantonio said that barring the additional stealth and AI code that the team wrote, UE3 enabled their artists and level designers to start immediately prototyping layouts and mission concepts.

"Our game is special, blending first-person melee combat, stealth and physical mobility," said Colantonio. "So our tech director and lead gameplay engineer – Hugues Tardif and Stevan Hird, respectively – have led their teams in the creation of a variety of features and code extensions."

"We've used Unreal Kismet, Unreal Lightmass, the new navigation mesh system and most of the standard tools. We've also added our own features, such as audio propagation and a dialog editor, and we've replaced 'Matinee' with 'Soiree' because we needed to have a way for our NPCs to constantly switch between AI and scripted scenes," added Smith. "We're proud of the amount of things we got the engine to do at the same time for performance."

UE3, along with advances in technology, have helped the team implement exceptional lighting, develop better physics and gather more characters on screen with higher polygon counts. As a studio, there's a certain level of graphical fidelity that the team loves. Smith and Colantonio like the abstract game mechanics, but they also really like creating rich, interesting environments.

"We're using a lot of the advances in technology for a very analog AI system," said Smith. "The AI is not just a magic circle that knows where you're at when you're inside the circle. It has a set of view cones. It's used in the dark and with distance, where the vision falls off. The AI goes into an indeterminate state when it's not sure if it's seen you and will give up if it hasn't found you. It's a very warm, perception-based AI."

The end result is a non-linear adventure that begs for replay given the open gameplay choices. This rich virtual world, ravaged by a rat plague and filled with unique characters, shows yet another way that UE3 is being used to enhance the creative visions of game developers around the globe. *Dishonored* stands out from the crowd, such as UE3 games *BioShock* and *Mass Effect* before it, as an innovative new interactive adventure.

Thanks to Arkane for speaking with freelance reporter John Gaudiosi for this feature.

UPCOMING EPIC ATTENDED EVENTS

GDC Online
Austin, TX
October 9-11, 2012

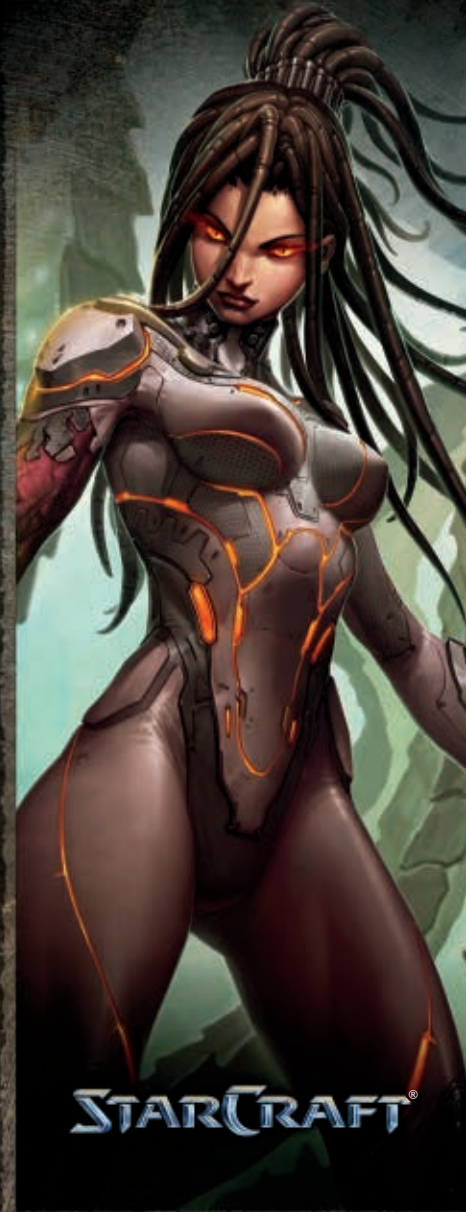
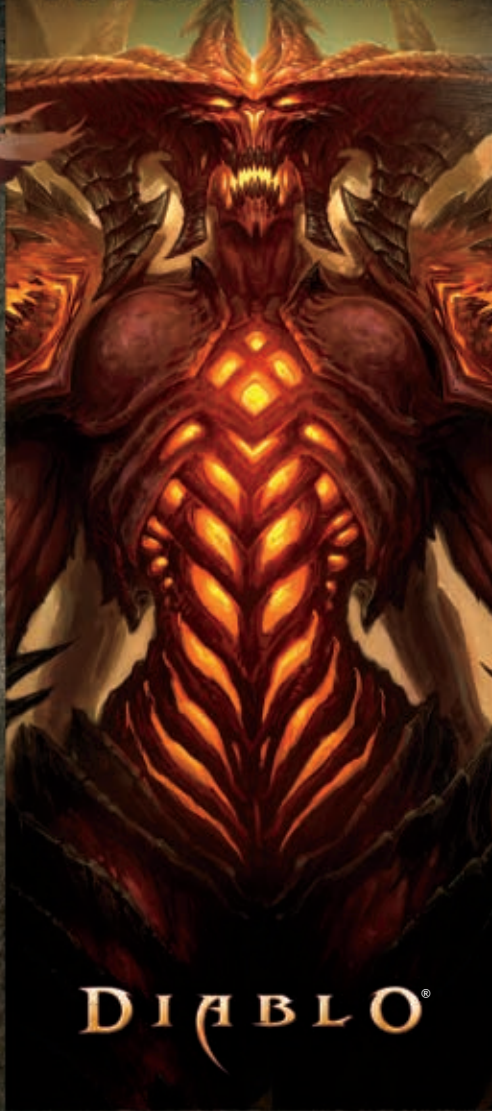
MIGS
Montreal, Canada
November 13-14, 2012

Please email licensing@epicgames.com for appointments



BLIZZARD®

ENTERTAINMENT



BLIZZARD IS HIRING

We are actively recruiting for the following key positions across our game and online technology teams:

SENIOR SERVER ENGINEER | SENIOR RELIABILITY ENGINEER | SENIOR TOOLS ENGINEER
SENIOR CONSOLE ENGINEER | LEAD 3D ENVIRONMENT ARTIST
SENIOR 3D ENVIRONMENT ARTIST | SENIOR 3D CHARACTER ARTIST | FX ARTIST
LEAD BATTLE.NET DESIGNER | LEAD CAMPAIGN DESIGNER | SENIOR LEVEL DESIGNER
PRODUCTION DIRECTOR | BUSINESS OPERATIONS DIRECTOR

jobs.blizzard.com

Follow us on Twitter: [@blizzardcareers](https://twitter.com/blizzardcareers)

SIGGRAPH 2012 HIGHLIGHTS

THE STARS OF THE SIGGRAPH 2012 SHOW FLOOR


By Carey Chico & Mike De La Flor

Every year, graphics gurus and engineering experts congregate in a convention center to talk about new technologies, techniques, and trends at the annual Siggraph conference. Since we're nearing the end of this console cycle, this year's show focused mostly on maximizing performance and quality on existing platforms—and making high-end development gear more accessible to studios with smaller budgets. From the latest on voxels and new cutting-edge GPUs, to what's new in the next generation of 3D art software and digital-art peripherals, these are the trends at this year's Siggraph.



Unreal Engine 4.

VOXELS

 Voxels are three-dimensional pixels. They're nothing new—MINECRAFT is the most recent high-profile game to use voxels—but they're making a comeback for a very particular use in games. Because producing indirect lighting

effects (such as global illumination) is often computationally expensive, these effects have often been precomputed and baked into the scene. But what if your lighting and world are incorporating more and more dynamic elements? Enter Voxel Cone Tracing. Martin Mittring from Epic Games spoke about the increasingly used sparse voxel octree global illumination in his "The Technology Behind the Unreal Engine 4 Elemental Demo" talk, specifically focusing on how they used

a technique called voxel cone tracing (originally developed by Cyril Crassin: www.icare3D.org/research/GTC2012_Voxelization_public.pptx) to compute global illumination in real time.

In the Unreal 4 engine, voxel cone tracing casts rays off of each screen pixel, which travel through the scene volume and which dynamically computes global illumination. Where ray tracing uses infinitely thin lines, voxel cone tracing uses "cone traces" as an intersection primitive, which generates

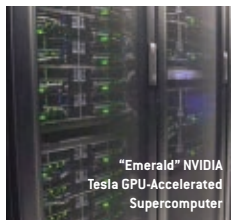
many texture lookups into a multiresolution volume texture that stores directionally encoded voxel data. With this approach, they can use the cone's opening angle to gather light from more directions, and they can efficiently approximate distant geometry through filtered or mip-mapped pixel data (the far scene gives a larger sample area with less accuracy), which can then be used as an LOD system for lighting detail. The voxels deliver an alternative representation of the level

geometry, and are capable of approximating complex geometry with fewer voxels. Because they are being cast off of screen pixels, the limitation is the resolution of the image. This significantly limits the time needed to produce results, and hence can be done in real time.


There are a number of benefits to the method that Epic employs here that help produce the best results. First, the voxels support fractional occlusion and therefore produce smooth cone-trace results. Second, glossy reflections can

SIGGRAPH 2012 HIGHLIGHTS

be implemented with a single cone trace, while ray tracing often needs hundreds of rays. Third, since opacity and color accumulate during the trace, pixels hidden behind other opaque pixels can be cast aside for optimization. Finally, by only updating the parts of the scene that have changed, Epic can optimize the calculations even further. Martin showed off a lit scene from the Unreal demo which demonstrated indirect lighting and ambient occlusion. If you'd like to know more, you can find Mittring's slides and some sample video here: <http://advances.realtimerendering.com/s2012/index.html>



GPU COMPUTING

 GPU computing—the practice of using your powerful GPU along with your workstation's CPU to speed up calculation-heavy tasks—has gained significant momentum in computer graphics lately, and NVIDIA, AMD, and ARM all showed off their new gear for the next year.

NVIDIA announced the second generation of the Maximus workstation platform, which combines its Quadro and Tesla cards into one system so artists can use the same system to model, texture, light, and animate heavy datasets, then render in real-time or even run simulations while designing. Maximus is a scalable system, so artists can choose to focus processing power


either in the visualization (such as real-time effects) or computing (real-time rendering) end of the spectrum as necessary.

AMD launched four new FirePro graphics cards, each of which use AMD's Tahiti GPU. The flagship card is the FirePro W9000, which has 6GB of GDDR5 memory and can hit processing rates of up to four teraflops, as well as lower-end W8000, W7000, and W5000 models. Like NVIDIA's Maximus, the new FirePro cards are designed to integrate heavy visualization tasks such as design, and heavy computing tasks like rendering, into one system.

Meanwhile, ARM had its new quad-core Mali-T604 GPU on display. Several applications were running on the new GPU, each focused on specific features. The bottom line is that the Mali-T604 delivers significant GPU processing power for real-time physics, particle simulations, and advanced lighting, which will give mobile developers more power to play with in their games. (Samsung mobile devices have already begun to employ the Mali GPU.)



CHEAP 3D PRINTING


 3D printing was at Siggraph in a big way this year—not because it's a new topic per se, but because it's getting cheaper (read: within the four-figure limit of a typical credit card). Considering that we're seeing more crossover between digital and physical

goods in both directions (see: TEAM FORTRESS 2 hats and SKYLANDERS figures, for example), creative game developers will no doubt want to keep their eyes and ears on the democratization of 3D printing tech.

3D Systems (<http://cubify.com>) practically launched the 3D printing industry in 1986, and the company's new Cubify (\$1,300) is a consumer-oriented 3D printer with a small (5.5-inch) printing platform. Combine this with 3D3 Solutions's (www.3D3solutions.com) Kinect-based 3D scanning system (\$599)—which lets you capture a 3D object and use its software to produce an exportable mesh, complete with diffuse texture information—and you might just have what it takes to kickstart the consumer 3D printing market.

Acute3D (www.acute3d.com) takes a different approach with its Smart3DCapture software suite, which builds 3D models from simple photographs—no scanning hardware required. Smart3DCapture consists of two programs: S3C Scanner scans individual objects, and S3C Surveyor produces 3D surveys of physical locations. In one demo, Acute3D representatives showed how Surveyor could capture entire cities with satellite photo and street-based panoramic data—handy, perhaps, for game devs looking to quickly prototype a large city level.

CHARACTER ANIMATION AIDS

 Character animation is a tedious process, so we weren't surprised to see plenty of tools and services on display that promised to streamline the animation workflow.



Qumaron human input device.

CyberGlove Systems (www.cyberglovesystems.com) typically develops technology for military, scientific, and engineering applications, but the new CyberGlove II system was demoed with a rigged character in MotionBuilder. Eighteen sensors lining the hand and fingers of the glove are mapped to the joints of a rigged digital character; slip the glove on and move your hands and fingers, and you will pose and animate the figure accordingly. You can use one glove to drive more than one character at a time, or you can use a pair of gloves to animate several characters. The connection between your hand movements and the corresponding animation doesn't feel particularly intuitive at first, but it doesn't take long to adapt to it.


SoftEther (www.softether.co.jp/en) was on hand to show off Quma, a jointed, anthropomorphic, USB-based input device that looks like a small toy robot. Quma is essentially a miniature, real-time, motion-capture system; it has 32 sensors in its joints that capture its position and send the data to a rigged digital character in its proprietary

software (though Maya and 3Ds Max plug-ins are available, too). It's a direct and intuitive way to animate a character, and you get immediate results. Right now, the Quma can only animate bipeds and is only shipping in Japan, though support for quadrupeds (and international sales) is in the works.

iPi Soft showed off a markerless motion-capture software system called iPi Motion Capture 2.0, which uses your PC and one or two Kinect sensors (or even webcams/digital cameras) to let you create your own mocap studio, use the proprietary software to clean up and edit the animation post-capture, and convert it into commonly used file formats (including GBX, BVH, and Collada). The capture area is around 7 by 7 feet, but can expand to 20 by 20 if you implement more cameras. There is a trial version of the iPi Mocap Studio software on iPi Soft's web site, and the iPi Recorder software is free. Pricing for the iPi Motion Capture software line is \$295 for the Express version (supports one Kinect, no multicamera support); \$595 for the Basic version (supports up to two Kinects and three or four video cameras); and \$1,195 for the Standard version (two Kinects, three to six video cameras, and can track multiple people).



ASSET DEVELOPMENT APPLICATIONS

 From models and textures to animation and special effects, you can't make games without art

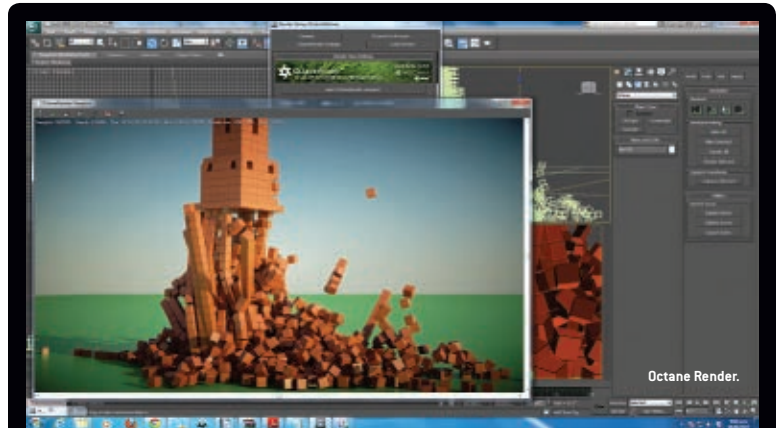
assets—or the applications artists use to build them. Autodesk, Pixologic, NewTek, and other developers were on hand at Siggraph 2012 to show their newest wares.

Autodesk's 2013 Entertainment Suite (3ds Max, Maya, SoftImage, Mudbox, Motionbuilder, and SketchBook Designer Pro) should be intimately familiar to any reader of *Game Developer*. Beyond the usual application-specific incremental updates, the overall theme is improved interoperability between Autodesk products and external applications. For example, Maya features native support for Alembic, making it easier to transfer scene and animation data between applications; Media Sync in 3ds Max can transfer scene and animation data to and from Adobe After Effects; Mudbox's compatibility with Adobe Photoshop continues to improve the process of exchanging paint layers and data between the two applications.

Pixologic's ZBrush 4R4 was released a few weeks before Siggraph, and the hot topic at the booth was the new QRemesher Alpha, which automatically retopologizes any mesh, and the QRemesher Guide Brush, which lets artists direct polygon and edge flow for cleaner, more predictable quad meshes. Also new to version 4R4 is the Virtual Machine, which lets the artist look through thousands of undos to review a model's history over time, and the new Curve engine, which works with various brushes (such as the Insert Multi Mesh brush or Tube brush) to help you lay down complicated organic or mechanical details.

NewTek showed off LightWave 11.5, the latest version of its workhorse 3D modeling and animation suite. This release introduces Genoma, a modular rigging system built into Modeler that lets artists quickly rig biped or quadruped models, and includes several preset rigs for common parts like spines, arms, legs, and even wings. Also, LightWave's core modeling toolset has been catching up to its competitors with new tools like Edit Edges, Thicken, and UV Unwrap, and its animation and special effects features have added a new specialized particle system called Flocking, which simulates more organic-looking flock movement, and improved soft-body bullet dynamics. NewTek also spotlighted LightWave's Virtual Studio Tools, which provide artists with real-time performance capture for virtual cameras and lights and character puppeteering via game controllers. For example, a development team could move through and interact with a game level inside LightWave to work out animation and design.

Lightbrush (\$1,500) is an intriguing new shadow and lighting manipulation tool by Tandent Vision Science that was recently released for the Mac (<http://tandent.com/lightbrush.html>). In the demo, Tandent representatives showed how artists looking to use photos as sources for textures could take a photo of a brick wall that had tree shadows splayed all over the surface, use Lightbrush to indicate an area of diffuse color and an area of shadow, and let Lightbrush separate the shadows out into a separate layer. A Windows version is in the works.



emerging trends and products

CLOUD RENDERING: Cloud rendering is a growing trend for smaller development houses that do not have their own data centers or render farms, and we saw plenty of them on the show floor, including Keyshot (www.luxion.com), Fox Renderfarm (www.foxrenderfarm.com), XL render (www.xlrender.com), and Rendercloud (www.scalar.ca/rendercloud).


OPTIS MATERIAL SCANNER (OMS2): The OMS2 (www.optis-world.com/products/hardware/OMS2.html) is a handheld device that essentially lets you scan textures into your CAD application, which could be handy for artists looking to make their materials and textures more realistic. The OMS2's sensor picks up surface and light information and dumps it to your laptop so you can quickly work with your new material. You can also pair the OMS2 with Theia-RT, Optis's real-time raytracer, to scan real-world objects and update Theia-RT immediately.

PATH TRACING: Jules Erbach showed two interesting tech demos that aren't quite ready for prime time yet: OTIO's Path Tracing renderer, called Octane Render (<http://render.otio.com>), and a real-time path tracing tech demo called Brigade. Octane Render is a path-tracing renderer that is so fast that users can change their material settings and receive near-instant feedback. Brigade (<http://igad.nhtv.nl/#bikker>) is a research project attempting to provide a real-time path-tracing solution. Brigade's path-tracing implementation tracing uses both the GPU and CPU, and balances the use of both depending on the time to render each frame. The renderer can support an unlimited number of light sources.

Z-SPACE: Z-Space (<http://zspace.com>) is a 3D-viewing station that features a 24-inch passive stereoscopic 3D display fixed with tracing sensors. With a pair of polarized 3D glasses and the included stylus, you can use the display to view and manipulate objects in 3D. This wasn't the only 3D-viewing system shown on the convention floor, but it was certainly one of the most impressive systems, and we can expect to see many more 3D systems appearing in future shows. — *Carey Chico*

DIGITIZING TABLETS

Wacom's booth was constantly crowded with attendees waiting to try out Wacom's latest line of Cintiq drawing tablets. The main attraction was Wacom's new flagship panel, the Cintiq 24HD Touch, which lets the artist directly manipulate documents and content with hand gestures, so you can rotate, move, and scale with one hand while holding your pen in the other. It does take a little getting used to, and right now it's only supported in Corel Painter and a handful of Autodesk productions, but we think it'll catch on quickly.

Also, Wacom has discontinued its popular Cintiq 20UX model in favor of the new Cintiq 22HD, which features a high-definition, widescreen LCD panel that is brighter and has higher contrast levels than the 20UX, and a stylus built on the Intuos 4's design, with 2,048 pressure levels and 60 degrees of tilt sensitivity. Unlike the much larger Cintiq 24HD (see our review in the September 2012 issue of *Game Developer*) the 22HD retains the standard popular Cintiq stand. 

CAREY CHICO is a 16-year veteran of the game industry.

Having worked on all platforms, from PC to console to mobile, he has carefully nurtured his talents to expand his skill sets driving products and managing studios from creative, production, and business mindsets. He has always been an active member of the industry at-large, formerly as a GDC Advisory Board member, a freelance writer for Gamasutra and Game Developer magazine, and a frequent speaker at industry trade shows and events.

MIKE DE LA FLOR is a freelance medical illustrator, animator, instructor and writer. He's the author of *The Digital Biomedical Illustration Handbook* and other CG books.

save the date

GDC 13

GAME DEVELOPERS CONFERENCE®

SAN FRANCISCO, CA /// MARCH 25-29, 2013

EXPO DATES: MARCH 27-29, 2013

www.GDCONF.com



Disclaimer: This article does not provide legal advice and should not be interpreted as such. If you have any legal questions, please consult with an attorney licensed in your state.



"game"

[how to shut down a small game studio]

"people"

I pressed send, and one by one, the termination notices flew off to their unlucky recipients. That was the simple part. I was the only one in the office at that point, surrounded by several lined-up computers, assorted cables, fancy wire garbage cans, and cubicle walls for 10 workstations. The story behind the closure of my studio, Globex LA, was simple: The latest bridge loan targeted our publishing and marketing effort in China, which meant that I suddenly had no funds to make payroll. Given the suddenness of the situation, the only thing I could do was soften the blow and work to dissolve the studio in a way that took care of our staff in the best way possible. Closing down a studio is not easy (and it never should be), but one can enter into it prepped and aware of the details that will help make the process easier on both you and your employees. Hopefully, you'll never have to use it. >>>

By Carey Chico



exiting gracefully

☑ Not every studio closure has to be a 38 Studios-esque gossip-strewn affair. I was with Pandemic Studios when it closed in 2009, and its closure was as well thought out and orderly a process as one could expect. A huge group of HR agents and escorts were on hand to manage the process, group meetings were delivered on each floor to make the announcement and outline the procedures for termination, and support programs to help people write résumés and find new jobs were discussed. Surprisingly, there was laughter and smiles, unsurprisingly mixed with melancholy and sadness. In the end, Pandemic's closure was executed in a manner meant not just to care for the employees, but to engender a little bit of hope in them as they left the office for the last time. If you have to close, that's how you want to do it. Given that I was also one of those employees who received my layoff notice that day, I can tell you that I'm not speaking from the perspective of those who were in charge that day.

1. If you have no money and you cannot guarantee salary—and you know it—don't let your employees work another day.
2. If you terminate your employees and still haven't paid their wages, then the clock starts ticking.

In the situation where you have unpaid wages, the following rules then go into effect:

1. Each day that goes unpaid is charged a full day's wages.
2. This goes on for 30 days.

After this period, the laws regarding unpaid wages are different in every state. In California, we could have incurred penalties from the state; in other states, it could have been a misdemeanor or even a felony.

judgment calls

☑ The bottom line is that closing a studio is a horrible experience no matter what you (or anyone else) do. People in charge never relish the opportunity to do layoffs or mass firings. The best thing to do is not lose focus of the people during the process. Focus on closing your studio in the fairest way possible. This starts with being honest not only to your employees, but to yourself. It's very easy to justify keeping the team in the dark, telling yourself that you are shielding them from harm, but most of the time this is just nonsense. The selfish part of you is saying "If I tell them what's going on, they'll leave," but the compassionate part of you is saying, "I need to let them know what's going on."

If your conscience still hasn't triumphed, you might be more persuaded by the law: The simple fact is that you cannot let people work for free, knowing that you cannot pay them. (See the Fair Labor Standards Act in the Resources section.) Two things are important here:

prepping the employees

☑ Preparing the employees for the studio shutdown can be the difference between a chaotic closure and an orderly, responsible one. Here are some suggestions and guidelines on how to bring the house down while limiting collateral damage.

Close your studio earlier than you want to. In an ideal scenario, it would be best to prep for a shutdown before you have run out of money. It's not easy, of course; often, many founders are extremely driven individuals who may have misplaced optimism regarding the state of the company, and can end up being their own worst enemy. But if the reality is that you have only one month of funds left with no potential financial solutions on the horizon, you might be better off laying off your staff with at least one month of severance and outstanding vacation pay.

Tell everyone together. Ultimately, it's best to perform a mass layoff together, as a company, in one room. If you are a larger organization, you



ILLUSTRATIONS BY JUAN RAMIREZ



will have to bring your leadership team on board early and employ them in the actual termination process, since they will be needed to help deliver the information. Most everyone should hear about it at the same time. In my case, at my studio, I had my COO prepped for the event, but we both sat in the same room and let everyone know the situation together.

Unpaid leave can buy you time. After notifying the team about the funding problem, the next thing I did was put them on unpaid leave. Given the extremely surprising nature of our shutdown, this allowed the studio to wind down with a softer landing. The nature of unpaid leave is that it retains the hope of resuming work within a short time period. This is useful if you know you can't pay people now, but you have high hopes that they can come back to work within a short period of time. In our case, we had high hopes that this would happen. We had 15 days remaining in March—which meant 15 days of active benefits before I had to pay new insurance premiums for the staff. This may be dependent on your state's labor laws, though, so you'll have to figure out what your specific situation requires of you.

Work-sharing programs may soften the blow. In my case, California had a statewide work-sharing program that can help employees on unpaid leave; if a company implements a reduction in wages or hours greater than 10% each week, then the company can apply for work sharing on behalf of its employees. The state then will pay partial unemployment benefits relative to the percentage of wages that have been reduced.

While this doesn't equal a 1:1 replacement for wages lost, it does allow employees to receive unemployment benefits services before they are unemployed. It also allows a company to try to retain employees rather than do immediate layoffs to save money. The work sharing program even allows a slow ramp down of wages/hours to permit a full 100 percent benefit while on work sharing.

It's important to note that while the goal of this was to supplement the lost wages, it also was a strategy to improve team morale. To make it even more fair, we also initiated four day work-weeks. This was a blessing and a curse in some ways. The goal was to again work to the employees' benefit, but in the

end this may have actually slowed down the team's momentum and limited progress on game projects that they were working on. An important note to clarify is that I implemented work sharing in January, two months before the studio reached the point of impending closure. At that point, the goal was to retain employees as long as was necessary until the company could reach revenue generation.

termination

Leading up to the final date of March 31, I notified the parent company of our impending closure, and prepared paperwork for the team in the event that I would have to terminate them. The key reason I could not keep them on unpaid leave past March 31 was because I would have to pay the premiums on all benefits for that next month, so I had no choice but to terminate them if we could not find the funds to move forward. As it turned out, that is exactly what happened, so I issued termination notices to the team via email (we didn't need to call them back to the offices for that). The termination process requires the employee to fill out a few different forms, which help the employee apply for unemployment (and help the company cover its butt).

winding it all down

At this point, I was the only employee left. I had deferred my resignation in order to take care of all of the remaining issues of a closure. Despite having issued the termination notices, I wasn't done yet. I still had to take care of the bills, rent, benefits, and assets.





Craigslist is your friend. While selling assets to employees is a logical solution, it's not the best option. There are certain liability risks and potential wage claims and lawsuits that make selling assets to ex-employees a risky one. (Also, the concept of a newly unemployed person spending their last paycheck by putting it back into the company by which they were just terminated just doesn't seem fair.) To sell the assets, I posted ads on Craigslist and called other potential buyers, such as an auction house and a used-furniture reseller. I ended up selling most of the assets via Craigslist and managed to sell almost everything within two weeks. On the last day, I had arranged for a secondhand office-supply company to come by and take the remaining desks and cubicle walls out of the office for free, which wasn't ideal but let me close the doors on a vacant office. One might think that the office management may have wanted the furniture for the next tenant, but this wasn't the case with my office building.

Be persistent with lease negotiations. Our studio still had one year left on the lease, and the office management wanted us to pay a percentage of the final year for ending the lease early. However, with persistence you should be able to exit paying only for the time used. In my case, after much discussion and pleading, the management finally agreed to terminate by taking the deposit and remaining month of use.

Check your benefit payment timing. With most benefits providers (including ours), you pay premiums in advance of the month that they are used. I had managed to pay our health, vision, and dental benefits through the end of March, so the employees would remain covered until the end of that month. However, at the beginning of April, I would owe the next set of premiums, which I couldn't pay. The process of shutting down the benefits programs was fairly simple, but a few programs such as our VOIP service, 401k plan, and workers' compensation insurance plan caused me long-lived problems which remain outstanding as I write this. Be sure to check the status of your benefit programs and try to reserve enough to ensure a proper roll-off of services to employees.

Budget for insurance audits. This was one of our sore points—when you pay workers' compensation insurance premiums, you prepay for the year based on total wages for the office. At the end of the year, the insurance agency audits your studio to determine the difference between predicted wages and actual wages. If you are like any other studio, you'll be hiring or giving raises throughout that year, and you'll find that you owe more for workers' comp insurance than you previously paid. Of course, if you are terminating a studio and you have no cash, you'll find that you owe money for workers who are no longer working at the company.

Service contracts are a pain to terminate. Those service contracts you signed to minimize your monthly expenses will come back to bite you. We had a hard time dealing with our service contracts (such as our VOIP service) because if you are on contract, you may owe a large early termination fee. You will have a hard time convincing these companies that you don't owe the money—because technically you do—and cannot pay the termination fee because your company is being shut down. There isn't a whole lot you can do; you'll end up receiving collection notices for these outstanding amounts and you'll have to work out solutions to these issues. In our case, I've discovered that these companies have no empathy at all to the issues plaguing a company shutting down, and generally don't believe you when you say you have no more money to pay them.

for example, you may find out later that you are the guarantor for the card if the company fails to pay it. This comes as a surprise to most everyone who has a card for their business.

Generally, when you sign up for a business bank account, you get a business credit card along with it. What isn't so clear is that, even though you may be incorporated—even though you may have a company tax ID number attached to that account—your signature affirms your personal guarantee for the balance of that card. So, after the company has shuttered and you are long gone, you may discover an old balance from that card that is now your responsibility to pay.

the kindness of vendors

☑ If you're lucky, not all the people you work with while closing down your studio will be so diligent in seeking final payment. I was lucky enough to have several vendors who continued to provide services for me despite delayed payments. My IT consultant let me use my outstanding paid hours even after I could no longer make the monthly payments. My CPA graciously worked on profit-and-loss reports in spite of delayed payments for her services. My office furniture agent helped provide me with contacts for resellers who would help me remove the furniture from the office. My HR consultant granted me zero-fee consulting calls as I worked out the details of the termination process. Thankfully, with these selfless good-willed supporters at hand, the process went much more smoothly than I could have hoped.

moving on

☑ I gave my resignation two weeks after I issued termination notices to the employees, but this wasn't the end of my work shutting down the studio. Even after resigning, I kept in contact with all the employees to give them guidance. In the end, my personal belief is that the most important aspect to shutting down a studio is how you deal with the employees who put their trust in you and in the company. While it may seem that you are only dealing with X number of employees, the repercussions of the event will affect spouses, children, and extended family in ways that are not initially obvious. Do what's right and fair. Shield them from pain, not the truth, and always remember that after this experience, there will be another one—and it's a small industry indeed. In some ways, we are remembered for not only how we ran the studio and for what products we made, but also how we cared for our employees when the business ended. 🙏

resources

- Fair Labor Standards Act: www.dol.gov/whd/regs/compliance/fairpay/fs17g_salary.pdf
- California Total and Partial Unemployment documentation: [www.edd.ca.gov/uibdg/Total and Partial Unemployment_TPU_8005.htm](http://www.edd.ca.gov/uibdg/Total_and_Partial_Unemployment_TPU_8005.htm)
- California's Work Sharing Unemployment Insurance Program: www.edd.ca.gov/pdf/pub_ctr/de23229.pdf

CAREY CHICO is a 16-year veteran of the game industry. Having worked on all platforms, from PC to console to mobile, he has carefully nurtured his talents to expand his skill sets driving products and managing studios from creative, production, and business mindsets. He has always been an active member of the industry at large, formerly as a GDC Advisory Board member, a freelance writer for *Gamasutra* and *Game Developer* magazine, and a frequent speaker at industry trade shows and events.

Special thanks to Helen Parker for HR consulting and Robert Wynner for legal counsel throughout this ordeal.

personal liability

☑ When you start a company, even a legal corporation, you might think that you are shielded from personal liability. Surprisingly, this isn't always the case; if you have a business credit card,



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 NIKOLAS LAZAR

VFS

Find out more.
vfs.com/enemies



INFORMING, ENGAGING, AND EMPOWERING THE INDUSTRY

gamasutra.com
the art and business of making games







RAY MAY ORIGINS 2

By Chris McEntee

With RAYMAN ORIGINS, Ubisoft Montpellier sought to bring RAYMAN back to his 2D roots with a fresh coat of HD paint and gameplay that felt both new and nostalgic. But no team has an easy time building a brand-new engine and using that work-in-progress tech to make a game—especially when that game is a reboot of one of the company's oldest and most beloved franchises. RAYMAN ORIGINS has had an amazing critical reception since its release in late 2011, and the game remains one of the most creative, chaotic, and valuable game experiences most of them have had to date. The following postmortem is a culmination of what I have heard and discussed with numerous teammates, as well as a personal reflection on what I experienced and witnessed while working as a designer on RAYMAN ORIGINS.

RAYMAN

origins

ORIGINS 2



WIZARDY VISIONS

1. START SMALL AND EXPAND WHEN NECESSARY

RAYMAN ORIGINS began as a small production of only five individuals with a vision of creating a creative and artist-friendly engine and using it to produce a quality game. They worked in the intimate Ubisoft Montpellier Villa under RAYMAN creator Michel Ancel, and at first they didn't even know their project would end up as a RAYMAN game. This small

team, working under rather calm circumstances, was able to explore and experiment with this new technology before the pressures of production and release dates sunk in, which let them establish RAYMAN ORIGINS'S identity and creative vision early on.

The real pre-production phase came about after several months of experimentation and engine building, and an additional team of people began to trickle in to help cover all the creative bases to

make sure that the pre-production process would build a solid base for the production team to tackle when it was their turn to come on board.

Eventually the team became about 75 at the Ubisoft Villa, and around 15 in Casablanca, making a team of more or less 90 pushing through a tough and short production cycle to get the game out on time, and even artists from *Ubisoft Reflections* in Newcastle were brought on to pick up some of the slack in the last stretch.

This is one of the major things that was done right on RAYMAN ORIGINS; the team was small when there was time to breathe, but when it was time to get the game out there, the team was able to expand with talent just in time to go full steam ahead and build a RAYMAN worthy of the franchise.

2. THE UBIART FRAMEWORK

A few years ago, Ubisoft Montpellier set out to build tools that would encourage artists to experiment freely, by themselves or in small groups, regardless of their experience or artistic domain. With these new tools, each of the artists can create a unique and interactive prototype with ease; seeing your artwork in action is as easy as dragging and dropping it into the scene. This was the concept behind the UBlart framework, and it is one of the main factors that contributed to RAYMAN ORIGINS'S success. With no in-engine asset browsers, the artist simply has to browse in Windows for the desired asset and drop it into the editor window, which makes it much easier to add assets into the game. The engine might not be user-friendly enough to be released to the public in its current state, but it is very streamlined so that it only has what is necessary for the development team.

While building UBlart, the team developed an integrated animation tool called GenAnim, which lets





animators combine hand-drawn animation with modern bone-driven techniques. This was integral to building RAYMAN ORIGINS'S unique animation style, which is reminiscent of Saturday morning cartoons with a hint of a French touch. In fact, a handful of animators on the RAYMAN team came not from the video game industry, but straight from animated film; with GenAnim, they can be easily integrated into the development team.

The UBlart framework is handy for designers, too, because it's easy to prototype with. Its intuitive skeletal binding system makes it easy for designers to use the game's existing characters and objects to set up complex ideas in simple ways. With UBlart, it's fairly easy to create a basic level topology, define collision, and apply textures to the level's "friezes" (blockout geometry), and the engine will make sure it has proper artwork that scales and tiles accordingly no matter what configuration or number of vertices the designer uses. The most important aspect of the UBlart framework to a level designer, however, is that the game is always running in the editor, so as the designer pulls and stretches the geometry, or drops in a series of platforms, RAYMAN continues to animate and move with the geometry, just waiting for a controller input to take him through the environment. Someone can even be playing while another person is editing, which makes the process of refining and iterating on a level much more efficient.

3. FOCUS ON THE THREE CS

At Ubisoft, the three Cs of a game (camera, character, and controls) are the most important thing to define and maintain because they determine the player's experience throughout the entire game. Obviously, when the control of the main character feels wrong, when the camera just doesn't cut it, or when the character animations don't provide good feedback to player inputs and in-game actions, the overall experience in the game world feels awful, even if the game and level design is outstanding. This

is especially true for platforming games, as they are entirely based around the player's ability to use the controls to deftly maneuver the player character through troublesome terrain. The better this character feels for players, the more quickly they will feel in control of the avatar. With RAYMAN, we combined hand-drawn animations with fluid movement metrics, which results in control that few players have ever experienced in a game before.

The UBlart framework was not originally developed specifically to create a RAYMAN game, and it wasn't until the basics of the engine were up and running that Ubisoft decided to create a RAYMAN reboot. From there, the artists defined his animation style, and when they felt that his movements looked fluid and fun, the programmers ventured to craft the in-game movement metrics to match the fluidity and timing of the animation. The player mechanics were designed and implemented iteratively, and the animations were changed and tweaked and thrown away until the three Cs felt perfect. Only once the main character was finished and comfortably fun to control did we begin working on level and gameplay element design.

The reason this worked out so well is that rather than making levels and designing the character mechanics concurrently and having both influence each other (which would potentially water down the three Cs to compensate for the level-design metrics), the team knew the character felt great and just needed to build levels that he could traverse with the use of his skill set and metrics. In this way, the team was guaranteed that no matter what, playing through the levels was going to feel fluid and fun. While the team did end up making changes to the three Cs later in the development process to fine-tune them, the fact remains that the RAYMAN that was functional before game production began was at least 90% of the RAYMAN that is playable in the finished product.

4. CREATIVITY VS. RATIONALITY

Creativity is a hallmark of Ubisoft Montpellier, and much of that comes

from creative director Michel Ancel. While creativity is obviously an important aspect of game creation, and had a very large hand in the critical success of RAYMAN ORIGINS, pure uncontrolled creativity can be dangerous to game development. A highly creative individual or team needs very rational designers to balance them.

In RAYMAN ORIGINS, the game director, Sébastien Morin (who has a strong background as a programmer), and the level design director, Olivier Palmieri (who is highly skilled in the field of rational design—I'll talk more about this later), helped contain the outpouring of creative ideas. Every day they would filter them into something equally charming but far more manageable for the team to build (and more easily understood by the player).

This process of rationalizing creative output for implementation didn't happen until pre-production had been finished for a while; for the year of pre-production, where the small core team was creating the UBlart framework and developing the three Cs, we didn't need to worry about it. Pre-production is the perfect time for creativity to run rampant and see where it takes you, because you are still working hard to define the identity of your game.

Once production started, however, the design directors stepped into their roles more firmly and began to tackle the sea of interesting concepts and mechanics and rationally construct a game system out of it, and Michel Ancel stayed involved in order to preserve the game's creative charm. It was this fine balancing act between two great forces that kept the project on an even keel and ultimately produced a game that was both creative and playable.

5. RATIONAL LEVEL DESIGN

In March 2012, I wrote a lengthy article on Gamasutra about Ubisoft's internal rational design methodology, which was conceived by Lionel Raynaud, Ubisoft worldwide content director, and Eric Couzian, Ubisoft game design conception director, and was taught throughout Ubisoft

by Olivier Palmieri. (You can read the article here: http://gamasutra.com/view/news/167322/Rayman_Origins_designer_Chris_McEntees_rational_approach_to_game_design.php.) Here is a quick explanation of rational design from that article:

"Rational design is all about eliminating unnecessary information, making things inherently readable, understandable, and apparent, introducing mechanics in an orderly and easily digestible fashion, and preserving the learning and difficulty curves of a game, known as macro flow. In principle, it is best to provide a player with significantly interesting and deep mechanics that are well explored and exploited through clever rationalized level design, rather than injecting the game full of one-shot gameplay mechanics to feign depth. A good mechanic, such as the portal gun in the Valve game Portal, can carry an entire game by itself with the addition of proper gameplay elements to help emphasize the usefulness and depth of the mechanic."

When this Design Academy was finished, Palmieri came to the RAYMAN ORIGINS team as level design director to teach the design team how to properly rationalize the game. Before this, the team had some idea of what they wanted for the levels in the game, but once they began the rationalization process, the design became far more consistent. The level design team employed rational design to understand how difficult a gameplay sequence was, why it was so difficult, and where in the macro structure of the game such a sequence would fit. Rational design helped define what color a gameplay element should be to greater clarify its function in the game, and it even denoted when some exotic gameplay was required to break up the repetition of the core gameplay.

Quite a large handful of reviews of RAYMAN ORIGINS commented on

RAYMAN

origins

ORIGINS?

its learning and difficulty curves, saying that they are as close to perfect as a game can get. While the team understands that it isn't perfect and there is always room for improvement, the fact that the players felt the need to comment positively on the difficulty of the game is testament enough to the implementation of rational design methodology. Through its use the designers were able to anticipate what a player should be learning at any given point in the game, as well as understand where it was appropriate to challenge the use of certain in-game powers.

Not only was the final game better thanks to the rationalization process, it helped push out the 60+ levels that were required for the release of the game. When we were getting close to the end of the production phase, we still had a lot of levels left to build, and we were able to speed up that process by creating rational design tables and level design briefs for gameplay element distribution, which gave the team a clear direction. Production near the end was lightning fast compared to the beginning, and the game made it out the door on time with the scope that was promised.

gameplay element, they should have a sense of instant familiarity with it, and an inherent understanding of what it is and what it will do. In this way, the designer makes an agreement with the player to provide clear signs and forms, and in return the player cannot feel cheated when they fail; if all the information is provided clearly, then it is only the player's lack of skill that is to blame for their failure.

In a creative project like RAYMAN, however, the development was very much driven by artwork, style, and silliness in character, environment, and object designs. Many of the gameplay elements that can be seen throughout the game were built backward, where form defined function. Rather than asking, "We need a platform for this world, so what can we make it look like that will clearly signify its function as a platform?" the team ended up with something more along the lines of "So in this concept for the current world we have these silly-looking birds. I suppose because their beaks are flat they could be platforms, right?" This doesn't sound so bad at first, but in this example, birds were used as platforms, enemies, and spiky traps. By using birds for

witnessed nearly every time during playtesting. Ultimately the birds remained birds, but their original red coloring (which was foolishly the exact same color the enemy birds were sporting) was changed to green to match the green color of the platforms in the previous world.

This was not an isolated case of form defining function in the development of RAYMAN ORIGINS, and as a result the game simply isn't as easy to read and understand as it could have been.

2. OUTSOURCING

Outsourcing has become a standard in game production; all the biggest and best projects outsource to other studios to simply get the massive volume of work done within the publisher's unrealistic timeframe. While RAYMAN was not exactly a massive production, it was still a triple-A game that needed enough content to justify the price tag, and since Ubisoft Montpellier's Villa was a fairly small team, the studio looked to its partner studio Ubisoft Casablanca.

Ubisoft Casablanca is a very solid studio. They have produced a number of Ubisoft titles, including a few platformers, so it made sense to work with them on ORIGINS. The issue with outsourcing is that the management of the studio and the communication between studios needs to be top-notch for it to work, especially when the core team is so small. When a change is made to the level-design guidelines, for example, virtually every involved person knows before the end of the afternoon at the Villa. For the outsourced studio, we have to make a phone call to relay this information. If it comes too late, the team there will have continued working under outdated guidelines, making some of their work obsolete even before it is created.

This is obviously something that can be avoided simply by having a proper chain of communication, but because of the fluid way that the RAYMAN team at the Villa worked, it became a problem quite quickly. At the studio, the team rarely had a fully fleshed-out set of rules and guidelines for level creation and implementation of gameplay elements; the creatives in the team

were constantly changing their minds because only when they saw the elements in action could they really know if they were what they wanted. This meant that rules changed almost every day, and it was not possible to constantly brief the Casablanca team on what changed twice a day. This meant that when it came time to review the outsourced content, it was unsurprisingly "sloppy" by definition, simply because it had been created under an outdated rule set. Ultimately the responsibilities of the Casablanca team were adapted to fit into an area of production more easily managed by an outsourcing coordinator, since the rules could be set in stone and easily overseen constantly from abroad.

In the end, the outsourcing really helped in the creation of the sheer volume of content required to have a complete game, and there is no doubt that the game would not have been finished had it not been for the collaboration with the Casablanca studio. It is simply important to realize the realistic scope of what can be handled from such a distance and how to best make use of your outsourcing partners to maximize the amount of usable content they can produce.

3. LACK OF PROPER TOOLS

The UBIart framework was, as previously stated, a godsend for the production of a modern two-dimensional platformer. The core of the framework was perfect for the task, but the rest of the content-development tools weren't properly integrated with Ubiart, which made it hard for the dev team to create levels or build out other aspects of the game in the engine.

Sure, there was GenAnim, which was a very big part of creating RAYMAN ORIGINS, but even GenAnim has its shortcomings. For an outside animator coming onto the project, GenAnim couldn't be any less user-friendly; there are no menus, hardly any buttons or tooltips, and even the animation timeline is as barebones as can be. One wonders how the animators get anything done at all in such a tool, but when you talk to one of the animators on the team it becomes quite clear: Everything is



Caption

WHAT WENT WRONG?

1. FORM DEFINED FUNCTION

Architect Louis Sullivan famously said, "Form follows function." In games, this means that when the player sets their eyes on the

platforms, the team was giving the player the initial impression that they were actually enemies or traps to be avoided. Unsurprisingly, the player's first instinct was to attack the platform rather than to jump onto it, which is something that was



done using keyboard shortcuts. In order to use GenAnim, you have to memorize the keyboard shortcuts.

While the lack of proper tools did not necessarily make the quality of the game any lower, what it did do is make life more difficult for the team, meaning production was not as efficient as it could have been with the proper tools in place.

This point was not exactly clearly recognized during actual production, as the team had been working in the basic engine since the beginning. Only when we started developing the follow-up game *RAYMAN LEGENDS* was it increasingly clear that the team needed better tools. There were many small issues in the engine that forced the level designers to find their own workarounds just to get the functionality they needed in place, but with the simple addition of proper tools and engine improvements the work was clean and finished five times faster than before.

4. LIMITED PRE-PRODUCTION SCOPE

Earlier, I mentioned that *RAYMAN ORIGINS* spent a significant time in pre-production, producing a very solid base for the game [the UBart framework, the three Cs, GenAnim]. Since it took so much work to build a solid core for *RAYMAN*, we had to choose what we wanted to focus on. During pre-production we had built the beginnings of a jungle world, complete with gameplay elements for the characters to interact with, but beyond that the team decided that the engine and characters were the most important areas to take care of before production started. This meant that we couldn't start investigating gameplay elements for the other worlds until the production phase began.

When the production phase started, we had concept art for the other five or so worlds in *RAYMAN ORIGINS*, since we needed that to define the final art style of the game. But concept art alone does not satisfy the requirements for pre-production on a world. We needed to know if the powers the player unlocks in each world are strong enough to define the gameplay experience and flexible enough to be

used in a wide variety of gameplay situations. Without this research, the design team was practically flying blind until enough content had been created to justify the decision or, worse, prove it wrong and send things back to square one.

Some of the gameplay elements that were created ended up being far less reusable than others, which resulted in a large time investment on the art side, the programming side, and even the level-design side, since we needed to experiment with each element before drawing conclusions on how useful it was. This is not necessarily anything new or surprising for game developers, and considering the sheer amount of different gameplay elements in the final game it would have been impossible to experiment with all of them back in pre-production. However, this doesn't change the fact that had the team focused on a broader range of different gameplay elements from all the proposed worlds in the game rather than focusing on every element in the first world, we would have been able to make the rest of the game more smoothly and throw less work away.

5. BOSS FIGHTS

It is more or less safe to say that out of all the content that *RAYMAN ORIGINS* boasts, the boss fights are far from the best part of the experience. Apart from the odd chase sequence or Mosquito shooter boss fight, the rest are quite lackluster and require little skill to defeat; the player simply has to meticulously memorize an otherwise unpredictable pattern, and then hope that there isn't another phase afterward.

The other issue with the boss fights is that there is a lack of clarity not only in the patterns, but in the creatures themselves. For example, each boss has its own weak spot that appears from time to time. This weak spot is always placed on top of the creature, whose entire body is otherwise deadly upon contact. What this means is that even if the boss is in a temporarily stunned state, if the player misses the weak spot and touches the boss, they are hit and possibly killed. This is very unclear for a player, and something that should and could have been addressed



when the bosses were being created.

A boss should also be there to challenge a player's understanding of the gameplay mechanics they have been taught during the previous world. Let's take the punch power *RAYMAN* learns in the Jungle world of *RAYMAN ORIGINS*. The first boss of the game, Poor Little Daisy, should have been designed to challenge players to master the punch ability in high-tension situations. That way, players can walk away feeling like they have mastered the punch ability, and no matter what is thrown at them in future levels, it will not be their lack of ability to use the punch that will slow them down. Instead, Poor Little Daisy simply required the player to employ the wall-run ability learned in the sixth world of the game to avoid Daisy's patterns until she hurt herself and revealed a weak spot, which could also be jumped on, rendering the punch entirely optional.

The bosses were developed by a more isolated space of the dev team, where some consistency issues were able to go by the wayside. Unfortunately, this left the player with subpar boss fights

which, through a few simple design choices, could have ended up far more understandable, relevant, and, ultimately, a lot more fun.

FROM ORIGIN TO LEGEND

Many things went right with *RAYMAN ORIGINS*, but there was plenty of room for improvement, too. Luckily, the team has the opportunity to create a follow-up title, *RAYMAN LEGENDS*, on the Nintendo Wii U. We've been given the opportunity to not just improve on the base of *ORIGINS*, but instead reflect on our shortcomings and craft a better, cleaner, and ultimately better *RAYMAN* title that is new and creatively refreshing for both the team and our players.

CHRIS MCENTEE was a level and boss designer on the production of *RAYMAN ORIGINS*, having previously attended college in the Netherlands (NHTV Breda University of Applied Sciences) where he discovered a strong passion for game design. He continues to work at the Ubisoft Montpellier villa, currently as a game and level designer on the upcoming Wii U title *RAYMAN LEGENDS*.



3DCONNEXION SPACEMOUSE PRO

BY CAREY CHICO

Let's face it: We're lazy. We made computers so we wouldn't have to compute by ourselves, and we made graphical user interfaces and mice so we wouldn't have to type anything. I'm just waiting for the day I can wear a semicircular, mind-melding device on my head to control my computer. This is kind of the appeal of 3Dconnexion's SpaceMouse Pro. The idea that movement in 3D space can be simplified to one single controller appeals to my inner couch potato. Rather than holding down some keyboard keys while simultaneously dragging an oblong device—which I'm no doubt also holding down keys on—I simply hold and push one round controller and I'm flying in 3D space. Easy.

FLOATING OVER THE WORLD IN BLISS

» The casual user can get the best feel for the SpaceMouse Pro in Google Earth: One minute, I'm smoothly gliding over the Earth looking down like an astronaut. The next minute, I'm controlling my descent through the atmosphere so smoothly that I feel like a feather. I found the movement so Zen-like that I spent an inordinate amount of time exploring random geological wonders of the world. I probably spent a whole hour touring Disney Paris with my son. As a promotional controller for Google Earth, SpaceMouse Pro shines.

I also gave the SpaceMouse Pro a shot in MICROSOFT FLIGHT to test how the controller worked as a replacement joystick. The software

detected and used the SpaceMouse Pro without any issues, but I was confused initially by how it was mapped. Once I got past the logic of shifting the mouse left-to-right in order to steer the plane, I was gracefully gliding over Hawaii. With MICROSOFT FLIGHT, the default settings for the 3D controller didn't use a return-to-zero position. In other words, if you pushed the controller left to turn left, and then released the controller, the movement of the plane remained fixed to that controller position. You had to push it back to the right in order to stabilize the plane again. Ideally, this would be an exposed setting that I could adjust to my liking.

Of course, you probably aren't planning on buying a \$300 mouse for flight sims and Google Earth.

Let's see how it fares in actual 3D modeling packages.

WORKING IN 3D

» The first work application I tried out with the SpaceMouse Pro was Autodesk Softimage. It was at this point that I experienced some of the chronic issues that continually plague these types of unique control devices. Ultimately, when you try to employ the mouse for things other than simple movement through 3D space, you discover it isn't the best method for getting actual work done. For programs in which you want to move through commands at lightning speed, manipulate window functions, and work in 3D space, the SpaceMouse Pro becomes something of a handicap. When my hand is resting on the 3D movement

3DCONNEXION SpaceMouse Pro

www.3dconnexion.com

PRICE

> \$300

SYSTEM REQUIREMENTS

> Windows 7, Vista, or XP Mac OS 10.4.6 (or later) USB port

PROS

- 1 Beautifully smooth operation in Google Earth and other flying programs
- 2 Useful for zoom and pan in Photoshop when used with Wacom Pen
- 3 Helps get the user out of the way for asset review in 3D packages

CONS

- 1 Not enough options for user-preference setting
- 2 Can't fully replace a mouse
- 3 Needs to support more general program usage in Windows

▼▼ The SpaceMouse Pro is meant for movement. When I used the device with its factory settings, I could glide through my model just as with Google Earth. It's perfect for demos, and it makes it easy to sit back and move through a 3D environment or review a model without my big head getting in the way. But when it comes to actually doing modeling work, it isn't quite so useful. ▼▼

knob, my mind is thinking "movement." When I'm doing everything else in a 3D package, my mind is thinking "everything else." There just aren't enough keys on this thing to go around.

I focused my efforts on configuring the device (which, thankfully, includes every key command in Softimage), so that I could do some modeling. As I began to pare down my list, I tried using the SpaceMouse Pro in actual modeling scenarios and quickly found that no matter what I set up, there was always some key combination that I couldn't perform without having to take my hand off the SpaceMouse and return it to the keyboard. There just aren't enough shortcut keys on the SpaceMouse Pro to replace a keyboard, which means you need to choose between resting your hand on the mouse or the keyboard—you can't do both.

The SpaceMouse Pro is meant for movement. When I used the device with its factory settings, I could glide through my model just as with Google Earth. It's perfect for demos, and it makes it easy to sit back and move through a 3D environment or review a model without my big head getting in the way. But when it comes to actually doing modeling work, it isn't quite so useful.

(Note: I should point out that at this point in my review, the device failed. The pan/zoom/tilt controls stopped operating correctly, and in some cases didn't operate at all. I have to thank the 3Dconnexion team for working very hard to debug the problems and then shipping a new unit to me that worked just fine.)

2D WORK WITH A 3D MOUSE

» Even though Adobe Photoshop is a 2D application, using a 3D mouse with it is surprisingly handy. I used the SpaceMouse Pro to smoothly scroll and zoom in and out of images, quickly tab through images, and instantly switch among my four favorite tools. Panning and zooming in on

images worked like a charm, and I immediately preferred using the SpaceMouse Pro to the dozens of keyboard shortcuts I'm used to. I mapped the other keys to quickly fit images to the screen and undo mistakes, which was handy.

I did notice that as I panned an image, it would jitter slightly, which was at most a visual distraction. Also, I could not paint while panning, though I think this is an issue with Photoshop, not the mouse. All in all, I could see the benefits of having the SpaceMouse Pro by my side in addition to my Wacom pen.


EVERYDAY PC APPS NEED NOT APPLY

» After trying out the SpaceMouse Pro with Photoshop, I wanted to know whether it would change the way I used my normal PC applications—Internet Explorer, Microsoft Word, and so on. While some of the shortcut keys did work with these applications, the coolest part of the mouse—the 3D controller—didn't work. The mouse currently doesn't ship with any presets for these standard PC apps, nor could you really manually configure it to work. I think that's a bit of a miss at launch for 3Dconnexion—I really wanted to have more use for the device beyond my art packages and Google Earth. The additional keys are a perk to a very sleek and cool control device, but for \$300 I want the mouse to seamlessly handle volume controls, playback controls, browser scrolling, and other application-specific tasks.

A PROMISING START

» I really wanted more from the SpaceMouse Pro. I liked it, to be sure, but ultimately, the best uses I found for it were in Photoshop (a 2D program), Google Earth, and MICROSOFT FLIGHT. For me, anyway, the idea of keeping my hand on both a mouse and a SpaceMouse makes it harder for me to work, and in most cases my keyboard connects me far more contextually to what I'm doing than a 3D mouse does. I believe that the SpaceMouse Pro could

even be an ergonomic replacement for a regular mouse—if the drivers existed, which we will have to wait and see whether the version 10 driver set supports. During my review process, the SpaceMouse Pro earned a place to the left of my keyboard and caused me to enjoy using Google Earth on almost a daily basis; kids especially love it. Perhaps if the support of the device extended beyond the programs tested above, the SpaceMouse could

have taken over the right side of my keyboard—a place of prestige for the universal controller, the mouse. 

CAREY CHICO is a 16-year veteran of the game industry. He has always been an active member of the industry at-large, formerly as a GDC Advisory Board member, a freelance writer for *Gamasutra* and *Game Developer* magazine, and a frequent speaker at industry trade shows and events.





FLOATING-POINT MISSES

DEALING WITH THE DANGERS OF FLOATING-POINT MATH

Floating-point numbers are ubiquitous in games because they are a convenient way to handle the necessary math. Floats gracefully handle overflow and underflow, and they have enough range and precision that we can sometimes forget how limited and dangerous they are. Developers naturally want floats to be the same as the real numbers we studied in school, and floats maintain the illusion quite well. But sometimes the illusion comes crashing down—and brings your game with it. In this article, we'll take a good long look at how floating-point math can make your life miserable—and some strategies for minimizing that misery.

32 BITS, REINTERPRETED

» If you're reading this, you should already have a solid understanding of the float format, so we'll keep the overview brief: A standard IEEE float consists of a sign bit, 8-bit exponent, and 24-bit mantissa (see **Figure 1**). Yes, this adds up to 33 bits, but all that magically fits into a 32-bit package, because the leading one of the mantissa is, for numbers above FLT_MIN, implied instead of being explicitly stored.

A handy feature of the floating-point format is that if you increment the 32-bit representation of a float, then you move to the next float away from zero. Adjacent floats (of the same sign) have adjacent integer representations. Incrementing the integer representation normally just increments the mantissa, but if the mantissa is all ones, incrementing the 32-bit integer instead overflows the mantissa to zero and increments the exponent field. Due to the magic of the implied leading one, this still gives you the next float. This technique works all the way from zero to infinity, and from negative zero to negative infinity, and we'll discuss its application later.

The range of a float is generally plenty large enough, but the precision is a bit weak. A 24-bit mantissa means (for instance) that for numbers above about 16 million, a float actually has less precision than a 32-bit integer. The rule of thumb is that, over virtually the entire range, the precision of



a float is between one part in 8 million and one part in 16 million.

DON'T TORTURE THE MATH GODS

» 24 bits of precision is enough for a lot of purposes, but if you're not careful you may inadvertently throw away most of that precision. It turns out that subtraction and addition are the most dangerous operations for losing precision.

Specifically, subtraction of numbers with similar magnitude (or, equivalently, addition of opposite signed numbers with similar magnitude) loses a lot of precision, and so does adding or subtracting a small number to a large number.

SUBTRACTION OF SIMILAR

» Imagine that you want to measure someone's height with a very long tape measure that

is accurate to about one part in a thousand. If you measure someone's height directly, then your answer will be accurate to within about two millimeters. Now imagine that instead you decided to stand the person on top of the Empire State Building, measure their height from the ground, measure the height of the building, and then subtract the two numbers. Instead of measuring 1.80 m directly, you are now measuring 382.8 m and then subtracting 381.0 m, with both measurements having an error of about .38 m. Your answer will be the person's height, plus or minus about .76 m. The loss of most of the top digits when subtracting similar numbers is known as catastrophic cancellation.

ADDITION OF DISSIMILAR

» Now imagine that you have two numbers with dissimilar ranges—perhaps the height of the Empire State Building and the height of a person—and both of these numbers are accurate to about one part in a thousand. If you add them and store the result in a number that is accurate to one part in a thousand, then most of the digits of the small number will be lost. If you do this repeatedly, the cumulative loss can grow arbitrarily big.

IT'S THE SIG-FIGS, BABY

» It surprises a lot of people that multiplication and division aren't the source of more math errors—probably because those

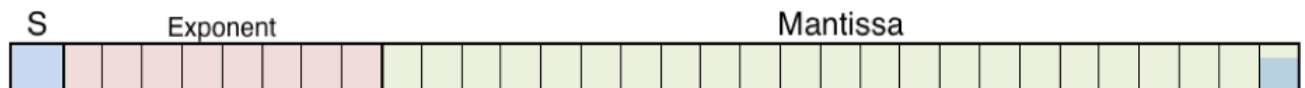


FIGURE 1: The anatomy of a standard IEEE float.

listing 1

A fragment of code used for dealing with particles—that unfortunately magnifies floating-point math inaccuracy.

```
float CalcTBad(float intervalEnd, float intervalLength, float time)
{
    float intervalStart = intervalEnd - intervalLength;
    float t = (time - intervalStart) / intervalLength;
    return t;
}
```

listing 2

A modified, more accurate version of Listing 1, reworked to be more accurate.

```
float CalcTGood(float intervalEnd, float intervalLength, float time)
{
    float timeToIntervalEnd = intervalEnd - time;
    float t = (intervalLength - timeToIntervalEnd) /
intervalLength;
    return t;
}
```

listing 3

Sample relative float comparison code.

```
bool AlmostEqualRelative(float A, float B, float maxRelDiff =
FLT_EPSILON)
{
    // Calculate the difference.
    float diff = fabs(A - B);
    A = fabs(A);
    B = fabs(B);
    // Find the largest
    float largest = (B > A) ? B : A;

    if (diff <= largest * maxRelDiff)
        return true;
    return false;
}
```

operations are so much more difficult than addition to do by hand. But despite our intuition's misgivings, floating-point multiplication and division are deliciously safe and stable, and cause virtually no precision loss. Even though we have to toss away half of the digits of the result, the relative error after multiplication is increased very little, if at all (except in the case of overflow or underflow).

It turns out that IEEE addition, subtraction, multiplication, division, and square-root operations are

all equally accurate. They are all guaranteed to be correctly rounded (by default this is round-to-nearest, tie goes to even). So why is subtraction of same-signed numbers so problematic?

If the values of the floats being subtracted are within a factor of two of each other, then the result is exact. It's pretty cool to have such a concrete guarantee about the chaotic world of floats. And yet you may have zero significant digits left, because most or all of the digits may have been canceled out. In this case, the absolute

precision of your input numbers is perfectly maintained, but the relative precision can get arbitrarily bad. In the example of measuring a person's height when they are on a skyscraper, the real problem is the initial numbers, whose absolute error (about .38 m) is insufficient for the ultimate goal of measuring a person's height. This inadequacy becomes manifest when the values are subtracted, as the subtraction makes the relative error about 200 times worse.

The other problem with addition and subtraction occurs when you are adding numbers of dissimilar magnitudes. Unlike with multiplication and division, the digits of the smaller number may in this case be completely ignored.

TWO WRONGS DON'T MAKE A RIGHT

» The worst-case scenario is if you add a small number to a big number (thus losing the least-significant digits of the small number) and then subtract a similarly valued big number (thus losing the most-significant digits of the big number). In this case you end up with very few significant digits—perhaps none. In other words, this is really bad:

Result = (Big1 + Small) - - Big2

Understanding these behaviors sometimes lets you significantly improve your code. As an example, the function in Listing 1 is representative of real code from a game I was working on. It takes in two numbers that represent an interval of time, and another number that represents a time within that time interval. The function's job is to return a value **t** from 0.0 to 1.0 representing how far through the interval **time** is. This isn't a great design for multiple reasons, but I'm going to ignore that for now and focus on the implementation.

This code was used for dealing with particles, so after the game has been running for a while it is safe to say that the end of the interval will have a value in the hundreds or thousands, while the interval length will have a

range of just a few seconds. We can therefore assume that **time** and **intervalEnd** are relatively close to each other, whereas **intervalLength** is much smaller. In Listing 1, the first subtraction is of a small number from a large number, so most of the digits of **intervalLength** will be lost—they won't affect the result. Then we subtract **intervalStart** from **time**, and we cancel out most of their most significant digits. The first subtraction tosses away low-order digits, and the second subtraction cancels out high-order digits. That leaves very few digits, and even when **time** is clearly in range, the rounding means that the result sometimes ends up being outside of the 0.0 to 1.0 range!

A better way to express this algorithm is to start by subtracting the two times. In most cases these two times will be within a factor of two of each other so this will be perfectly accurate—we can't do better than that. We then subtract the result from **intervalLength**, and since these values are likely to have similar values, we don't lose much accuracy. In many cases, we'll only need to round in the final division, and the result has many more digits of precision than the original calculation. See the improved code in Listing 2; the net result is that the fixed code is perfectly stable, and the client code no longer needs to clamp the result into the 0.0 to 1.0 range.

COMPARING FLOATS

» Most developers know that comparing floats for equality is not a good idea. With some exceptions, floating-point math operations are not exact, error accumulates over multiple steps, and you probably won't get exactly the result you wanted.

It's easy to say that you shouldn't do an equality test, but saying what you should do instead is much trickier. There are many examples of floating-point equality tests with epsilons (comparison tolerances) so small that they are just fancy-looking, more-expensive equality tests, or epsilons so large that that



you can drive a virtual truck through them.

The best starting point for comparing two floats to see if they are “close enough” is to take the larger of the two floats, multiply it by some small number, and use the result as the epsilon. The smallest number you should multiply a float by to get your epsilon value is `powf(2, -23)`, because that is the minimum guaranteed precision for a normalized float. Using a number much smaller than that is, for floats, just an expensive equality test. Larger epsilons are fine, but if you find that your relative epsilon needs to be thousands of times larger, then you might want to investigate to find out what is going on, since that level of instability can be hard to work with.

It turns out that `powf(2, -23)` has a standard name: `FLT_EPSILON`. The constant `FLT_EPSILON` is the smallest amount that you can increment `1.0f` by, and its value is also the smallest relative epsilon you should use with floats. Small multiples of `FLT_EPSILON` give progressively greater tolerance in the comparison, so you can pass, say, `10 * FLT_EPSILON` to allow for more error. See **Listing 3** for some sample comparison code:

When doing floating-point comparisons, you might find it handy to make use of the integer representation of floats. Since adjacent same-signed floats have adjacent integer representations, if you subtract the integer representation of two [same-signed] floats, your result is their distance from each other—the number of floats in between plus one. If two floats are adjacent, we say that they differ by one Unit in the Last Place [ULP]. By making use of a union of an `int32_t` and a float we can implement this technique in a way that stays within the aliasing rules of all the compilers I am familiar with (see **Listing 4** for an example). The check for the signs being different is important because subtracting the integer representations of opposite-signed floats is problematic when the signs don't match.

Unfortunately, the ULPs-based relative comparison technique can sometimes be quite expensive. Transferring data from the float unit to the integer unit can break pipelining and cause other significant performance costs. However, on processors with good store forwarding, this technique may perform well, and you can implement it in vector units that do float and vector operations in the same registers.

I like this technique when doing accuracy investigations because it lets me make definitive statements such as “The answer was only off by one ULP” or, equivalently, “The floats were adjacent.” These statements have more intuitive meaning than error ratios have. The semantics of the two techniques are similar, but not identical. When your numbers are near a power of two the meaning of an ULP suddenly changes—the distance between floats suddenly doubles (see **Figure 2**).

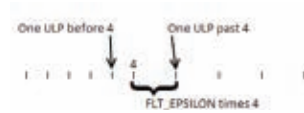


FIGURE 2: Measuring inaccuracy in ULPs.

THE MASK OF ZERO

» Maybe the Babylonians should have not invented zero. It just causes problems for floats. Relative error works incredibly badly when the numbers being compared are near zero. It isn't even well defined. If one of the numbers is zero and the other isn't, then the relative error is either infinite or 100%, depending how you measure it. If the two numbers straddle zero, then the relative error is even worse. (It's negative, I guess—whatever that means.)

Since relative epsilon doesn't work around zero, you need to use an absolute epsilon to handle this case instead. This is simply a small

```
listing 4
Using Units in the Last Place (ULPs) to compare floats Relative float
comparison code, using the integer representation of floats.

union Float_t
{
    Float_t(float num = 0.0f) : f(num) {}
    bool Negative() const { return i < 0; }

    int32_t i;
    float f;
};

bool AlmostEqualULPs(float A, float B, int maxULPsDiff = 1)
{
    Float_t uA(A);
    Float_t uB(B);

    // Different signs means they do not match.
    if (uA.Negative() != uB.Negative())
    {
        // Check for equality to make sure +0==−0
        if (A == B)
            return true;
        return false;
    }

    // Find the difference in ULPs.
    int ulpsDiff = abs(uA.i - uB.i);
    if (ulpsDiff <= maxULPsDiff)
        return true;

    return false;
}
```

number such that if the difference between the two floats is less than this number then you say that they are equal.

Finding the right value can be quite tricky, to say the least. There is no natural, obvious default value. If you do a bunch of test calculations in your game world, then you may find that an absolute epsilon of `0.00001` is appropriate. However if you were to change your units from, say, meters to millimeters, then all your numbers would get a thousand times larger and you would need to change your absolute epsilon to `0.01`. It's slightly terrifying that

this possibly essential value will vary depending on what units you have chosen.

I hate to say “Experiment and see what works for your particular domain,” but I don't have any better advice. You can do a complicated mathematical analysis of your algorithms in order to evaluate their stability, but I don't think that's particularly practical.

PI, PLEASE

» One fascinating example of the problem of comparisons around zero is for `sin(pi)`. Mathematics teaches us that the result of this

<code>float(pi)</code>	<code>+3.141592741012573</code>
<code>+ sin(float(pi))</code>	<code>-0.000000087422776</code>
<code>= a better approximation of pi</code>	<code>+3.141592653589797</code>

GDC ONLINE

YEARS

GAME DEVELOPERS CONFERENCE® ONLINE

AUSTIN, TEXAS | OCTOBER 9-11, 2012 | EXPO DATES: OCTOBER 9-10



Join us at GDC Online, October 9-11, for three days of world-class online and connected games content led by top industry experts.

SESSIONS

TUESDAY-THURSDAY

- Business & Marketing
- Customer Experience
- Design
- Production
- Programming

SUMMITS

TUESDAY-THURSDAY

- Game Narrative Summit
- Game Dev Start-Up Summit
- Gamification Day
- Smartphone & Tablet Games Summit

EXPO FLOOR

TUESDAY-WEDNESDAY

Explore the latest connected game technologies and innovations, and connect with product experts.

FEATURING

TUESDAY-WEDNESDAY



GDC
Play

VISIT GDCONLINE.COM FOR MORE INFORMATION



listing 5

Relative and absolute comparison routine.

```
bool AlmostEqualRelativeAndAbs(float A, float B,
    float maxDiff, float maxRelDiff = FLT_EPSILON)
{
    // Check if the numbers are really close -- needed
    // when comparing numbers near zero.
    float diff = fabs(A - B);
    if (diff <= maxDiff)
        return true;

    A = fabs(A);
    B = fabs(B);
    float largest = (B > A) ? B : A;

    if (diff <= largest * maxRelDiff)
        return true;
    return false;
}
```

listing 6

Relative and absolute comparison routine using ULPs.

```
bool AlmostEqualULPsAndAbs(float A, float B,
    float maxDiff, int maxULPsDiff = 1)
{
    // Check if the numbers are really close -- needed
    // when comparing numbers near zero.
    float absDiff = fabs(A - B);
    if (absDiff <= maxDiff)
        return true;

    Float_t uA(A);
    Float_t uB(B);

    // Different signs means they do not match.
    if (uA.Negative() != uB.Negative())
        return false;

    // Find the difference in ULPs.
    int ulpsDiff = abs(uA.i - uB.i);
    if (ulpsDiff <= maxULPsDiff)
        return true;

    return false;
}
```

calculation should be zero, but float math stubbornly refuses to give this answer. Typical results for `sin(float{pi})` are `-0.000000087422776`, which is 867,941,678 ULPs away from zero. That's as many ULPs as `1.5e31` is from `1.0`. Which is not close.

Most people are inclined to blame the implementation of the `sin` function, but that is inappropriate—the problem isn't with the `sin` function (which is extremely accurate) but with the value that is passed to it. It is important to understand that `float{pi}` is not equal to `pi`, and

listing 7

Classes for conveniently disabling and re-enabling (or vice-versa) floating-point exceptions.

```
// Declare an object of this type in a scope in order to enable a
// specified set of floating-point exceptions temporarily. The old
// exception state will be reset at the end.
// This class can be nested.
class FPEXceptionEnabler
{
public:
    // Overflow, divide-by-zero, and invalid-operation are the FP
    // exceptions most frequently associated with bugs.
    FPEXceptionEnabler(unsigned int enableBits = _EM_OVERFLOW |
        _EM_ZERODIVIDE | _EM_INVALIDID)
    {
        // Retrieve the current state of the exception flags. This
        // must be done before changing them. _MCW_EM is a bit
        // mask representing all available exception masks.
        _controlfp_s(&mOldValues, _MCW_EM, _MCW_EM);

        // Make sure no non-exception flags have been specified,
        // to avoid accidental changing of rounding modes, etc.
        enableBits &= _MCW_EM;

        // Clear any pending FP exceptions. This must be done
        // prior to enabling FP exceptions since otherwise there
        // may be a "deferred crash" as soon the exceptions are
        // enabled.
        _clearfp();

        // Zero out the specified bits, leaving other bits alone.
        _controlfp_s(0, ~enableBits, enableBits);
    }
    ~FPEXceptionEnabler()
    {
        // Reset the exception state.
        _controlfp_s(0, mOldValues, _MCW_EM);
    }
private:
    unsigned int mOldValues;

    // Make the copy constructor and assignment operator private
    // and unimplemented to prohibit copying.
    FPEXceptionEnabler(const FPEXceptionEnabler&);
    FPEXceptionEnabler& operator=(const FPEXceptionEnabler&);
};
```

for values of `x` near `pi`, `sin(x)` gives us the distance between `x` and `pi`. In other words, when we calculate `sin(float{pi})`, we are actually calculating `pi - float{pi}`. The result is an extremely accurate measure of the error in `float{pi}`. A bit of calculus shows us that the expected maximum

error in `float{pi}` (and hence in `sin(float{pi})`) is roughly `pi * FLT_EPSILON / 2`, assuming that `float{pi}` has an error of no more than half of one ULP. If we use this as the absolute epsilon, then the result suddenly seems totally reasonable, and is "equal" to zero. The table below shows how

`sin(float(pi))` is a very accurate measure of the error in `float(pi)`.

With all that background in place, the two generic comparison routines I recommend are in **Listing 5** (relative and absolute) and **Listing 6** (ULPs and absolute).

In both cases, if you know that you will not be dealing with numbers near zero, then you can use the simpler versions that omit the `maxDiff` absolute epsilon.

FLOATS FOR TIME AND LOCATION

» Fixing rare or difficult-to-reproduce bugs is the most tedious part of software development, so any design patterns that lead to such bugs should be avoided at all costs. In game development, there are a couple of patterns that lead to bugs that only occur after many hours of play, or in distant regions of maps, and if you don't take steps to avoid these patterns you may have some long nights before certification.

Floating-point numbers are designed to have consistent relative precision across a wide range of magnitudes. Or, to put it another way, they are designed to have much better absolute precision near zero. If you use float or double when this variable precision is not desirable—or when it might even be counterproductive—then you can end up with lots of exciting bugs.

If you use a float to store elapsed time in your game, that means you have more precision at the beginning of a game than at the end. When your game has been running for 60 seconds, a float that holds the elapsed time will have .0038 milliseconds of precision. Once your game has been running for a day (86,400 seconds) that same float only has 7.8 milliseconds of precision. This is a big enough drop to make it quite likely that you will have timing precision bugs that will only show up after the game has been running for a long time.

The simple fix to the timing problem is to store time in a 64-bit number—either a double or a 64-bit integer (representing the elapsed count of microseconds or nanoseconds). These techniques both work, but they are not

intrinsically safe because it is very likely that, even though your `GetGameTime()` function returns a double, some junior programmer will store the result in a float and you're back to having bugs that only occur after hours of gameplay.

Luckily, there is an easy solution. The whole problem arises because float and double have more precision near zero. All we have to do is not start our timer at zero. If we use a double, and if we

as these times do not grow as the game progresses, there will be no problems with bugs that only occur after several days.

EXCEPTIONS

» All code has bugs. That is the depressing reality of programming. In many cases the difference between shipping on time and missing Christmas is a matter of how quickly bugs can be found and fixed. Since games contain huge

point exceptions for its scope. By dropping these into key places you can validate the integrity of the code that you control, while sidestepping issues in the code that you don't control.

The ideal strategy is to put a float-exception enabling object in the start function of each thread, and then disable exceptions as needed. Pragmatic constraints might mean that you just enable float-exceptions in your particle system, or just in

❗❗ Fixing rare or difficult-to-reproduce bugs is the most tedious part of software development, so any design patterns that lead to such bugs should be avoided at all costs. In game development, there are a couple of patterns that lead to bugs that only occur after many hours of play, or in distant regions of maps, and if you don't take steps to avoid these patterns you may have some long nights before certification. ❗❗

start our timer at 4294967296.0 (2 to the 32nd power), then our precision will be consistent until the timer reaches 8,589,934,592 seconds—which won't happen for more than a century. In addition to ensuring consistent precision, this technique guarantees that anybody trying to store `GetGameTime()` in a float will immediately hit precision problems, and the mistake will quickly be discovered. If you prefer to return a 64-bit fixed-point integer, then an extremely large number as the start point will catch developers who store the game time in a float or an integer.

You'll see similar problems occur with player positions and world geometry. Storing player positions in a float means that you have much more precision near the origin, and it means that when you transmit player positions across the network you are wasting precious bits on transmitting the exponent. Using floats to store position is probably unavoidable, but when transmitting positions you should consider using fixed-point.

It is fine to store time differences—the time between events—in a float, and that may be necessary for compactly storing particle lifetimes and other attributes. As long as these times are relatively small, and as long

amounts of floating-point math, you can improve your odds of finding quirky bugs if you enable floating-point exceptions. Enabling floating-point exceptions for divide-by-zero, overflow, and illegal-operation is like adding asserts before and after every floating-point operation. I've used this technique for 20 years and it continues to be useful. On a recent project that was not written to be float-exception clean, I was able to enable float-exceptions in a few key systems and then find an illegal operation that was making particles disappear, as well as some reading of uninitialized stack memory that was triggering various forms of undefined behavior. It works.

The NaNs and infinities that are created by exceptional situations can also cause performance problems, particularly on the x87 FPU—yet another reason to avoid them.

The complicating factor is that not all code runs float-exception clean. Some libraries trigger illegal operations because they were carelessly written, and others may make legitimate use of divide-by-zero semantics. Either way, any practical way of enabling floating-point exceptions needs to account for this reality.

The simple technique is to define two C++ classes: one that enables a set of floating-point exceptions for its scope, and another that disables all floating-

your AI. Baby steps are better than no steps at all. See **Listing 7** for the classes I use.

SAVING FLOATS AS TEXT

» Text-based file formats can be terribly convenient because they are human-readable and easily editable, but it is not obvious how to convert a float to text in a way that retains the exact valuesufficient precision.

Printing the exact value of a float can require over 100 decimal digits. Luckily, printing the exact value of a float is rarely necessary. What is more useful is to print enough digits so that the original value can be reconstructed when the text is converted back to a float. For this purpose it is sufficient to print nine digits of mantissa. Ninety-four percent of floats can round-trip to text and back with just eight digits, but all floats are guaranteed to round-trip to text and back with nine digits. One possible way to do this is to use `printf("%.18e", f)`; which puts eight digits after the mantissa's decimal point. Perfect.

If you're not sure that you trust your `printf` and `scanf` functions to round-trip floats, you can instead easily write code that will iterate through all of the floats, printing to text, scanning back to float, and verifying that nothing is lost. On a modern computer, a complete test takes less than 15 minutes. There's no point in wondering

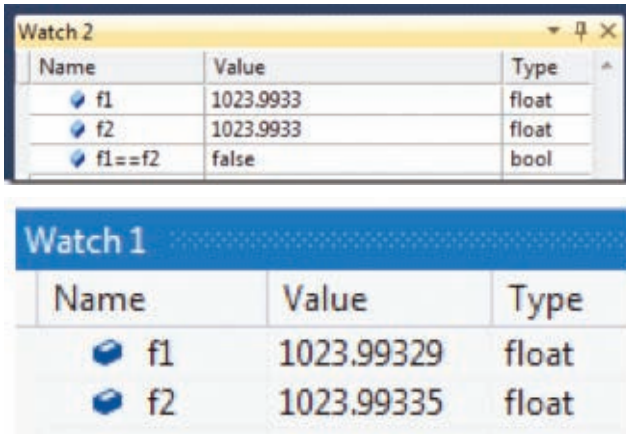


FIGURE 4: Testing the Visual Studio debugger for floating-point bugs.

whether this works when you can test it in the time it takes to get a cup of coffee.

While the standard requires that floats printed with nine digits of mantissa will round-trip, this is not quite guaranteed if you print floats with one implementation and then scan them from text in another (say, VC++ and gcc). If this is critical for your needs, you may need to test this.

LIES, DAMNED LIES, AND DEBUGGERS

>> The corollary of the above point is that your debugger's watch

```
float f1 = 1023.99328f;
float f2 = 1023.993347;
```

In versions of Visual Studio prior to VS 2012, these two numbers will both be displayed with eight significant digits, as 1023.9933. And yet, these two floats are actually distinct floats with different values. Figure 4 shows this issue at work in an earlier version of Visual Studio (top), though I am happy to note that I reported it to Microsoft and it has since been fixed in Visual Studio 2012 (bottom).

!! The simple fix to the timing problem is to store time in a 64-bit number—either a double or a 64-bit integer (representing the elapsed count of microseconds or nanoseconds). These techniques both work, but they are not intrinsically safe because it is very likely that, even though your GetGameTime() function returns a double, some junior programmer will store the result in a float and you're back to having bugs that only occur after hours of gameplay. !!

window (and memory window, tooltips, registers window, etc.) needs to display floats with at least nine digits of precision. Otherwise two floats that print to the same text value might actually be different floats—leading to obvious confusion. Visual Studio's debugger has for many years only printed numbers with eight digits of precision. (Oops.) To test this in your debugger of choice, create a program containing these statements and view the value of these variables:

PRECISION AND PERFORMANCE PITFALLS

>> All the modern CPUs I'm familiar with will perform float or double-precision math at the same speed. And yet, double-precision math can have significant performance costs that are often hard to see.

If we ask the compiler to add some floats together, it may decide, for complicated reasons, to do the calculation at double precision. This is the default on Visual Studio 2010 and earlier when compiling 32-bit code. When using the x87 floating-

further reading
This article is based on a series of thirteen13 blog posts written this year. To read more details, explanations, and code, and to discuss this article visit http://randomascii.wordpress.com/2012/09/09/game-developer-magazine-floating-point.
Years ago, I wrote an article on comparing floating-point numbers that became unexpectedly popular despite its numerous flaws. As an act of penance for its imperfect advice, I recently wrote a series of blog posts discussing floating-point math, most of which have been summarized in this article. You can find more of my work on floating-point math here; www.altdevblogaday.com/author/bruce-dawson/. Also, no floating-point article would be complete without a reference to David Goldberg's classic article "What Every Computer Scientist Should Know About Floating-Point Arithmetic" (http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html). It was written in a time when the IEEE floating-point math standard was not yet universal, but it still contains important insights.

point unit, there is no cost to this higher intermediate precision, but when using SSE/SSE2 these higher-precision intermediates are a problem. Each input float has to be explicitly converted to double precision, using cvtss2pd. If the result is stored in a float then it will have to be converted from double using cvtss2ps. The conversion instructions all have similar latency to a floating-point add or multiply. All these conversions can add a noticeable cost—I've seen more

want the calculation evaluated using double precision, and the compiler will dutifully add in three conversion operations.

The fix in this case is trivial—just add a trailing "f" to the floating-point constant. The trick is realizing when you have forgotten to do this. You could scan your source code for double-precision constants, or look in the assembly language for conversation instructions. Neither technique can easily distinguish between performance-critical code, and code where the conversions are harmless, but they are the best options I have to offer. If you find excessive conversions, check your compiler settings and your input values.

Let's end this article with a haiku:

Floating point game math
Is very fast and useful
But not accurate ☹

BRUCE DAWSON first worked professionally on games in 1987 at Distinctive Software Inc., where the coding standard mandated "colour" spelling and "zed-buffer" pronunciation. He has left the game industry twice, but always returns. Most recently he worked in the Xbox group at Microsoft for six years, then on Microsoft Windows, and is now a developer at Valve.

His parents did finally forgive him for dropping out of university.

GAME DEVELOPERS CHOICE ONLINE AWARDS HONOR RAPH KOSTER, WORLD OF WARCRAFT



Raph Koster.

» Raph Koster, a key industry veteran behind the prodigious MMORPGs *ULTIMA ONLINE* and *STAR WARS GALAXIES*, will be honored with the Online Game Legend Award at the third annual Game Developers Choice Online Awards. In addition, Blizzard Entertainment's influential MMORPG *WORLD OF WARCRAFT* will be inducted into the Choice Online Awards Hall of Fame during the

ceremony, which takes place Wednesday, October 10, at GDC Online in Austin, Texas.

These special award categories recognize developers and titles that have had significant impact in the world of online games. Honorees were selected through open nominations from the online game community and the distinguished GDC Online Advisory Board.

Koster, this year's Online Game Legend Award winner, has led a prolific career. As the lead designer on *ULTIMA ONLINE* and the creative director on *STAR WARS GALAXIES*, his contributions helped lay the foundation for the many massively multiplayer games that followed. Koster's professional credits span nearly every facet of game development, including writing, art, music, programming, and design.


Koster is considered a thought leader, and he is a frequent lecturer and published author on topics including game design, community management, storytelling, and ethics in game development. His book, *A Theory of Fun*, published in 2004, is considered seminal by educators and members of the art-game movement, and is one of the most popular books ever written about games.

This year's Game Developers Choice Online Awards Hall of Fame inductee, Blizzard Entertainment's *WORLD OF WARCRAFT*, was released in November 2004, 10 years after the launch of the company's venerable

WARCRAFT franchise. The game has earned its place in the Hall of Fame by dominating the MMO landscape, setting the standard for countless other MMOs that followed.

As the title nears its eighth anniversary, *WORLD OF WARCRAFT* remains one of the top-rated PC games of all time and continues to thrive globally, with a massive and passionate subscriber base. *WORLD OF WARCRAFT* is one of the most profitable and popular entertainment franchises in history, a bona fide pop-culture fixture. Members of the Blizzard Entertainment *WORLD OF WARCRAFT* team will be on stage to accept the Hall of Fame award.

Previous winners of the Online Game Legend award include MUD co-creator Richard Bartle and Kesmai founders Kelton Flinn and John Taylor, and Hall of Fame inductees for the awards include seminal titles such as *ULTIMA ONLINE* and *EVERQUEST*.

The 2012 Game Developers Choice Online Awards take place at the Austin Convention Center, October 9—11, 2012. More news and information about the show is available at its web site (www.gdconline.com) or via Facebook, Twitter, or RSS. GDC Online is owned and operated by UBM TechWeb, as is *Game Developer*. 

LEVEL DESIGN, DESIGNER MANAGEMENT, ART DIRECTION SESSIONS DEBUT AT GDC CHINA

» With GDC China 2012 just a few months away, show organizers have revealed the first sessions in its quickly expanding lineup. This first update introduces a robust level-design tutorial featuring developers from Bethesda, Valve, and The Fullbright Company, as well as game design talks from companies like EA Sports and Virtuos.

These debut sessions, along with the rest of GDC China, will take place November 17—19 at the Shanghai International Convention Center in Shanghai, China, and will bring together leading developers to share knowledge and ideas that will benefit the game industry in Asia and beyond.

Here are some details of GDC China's initial sessions:

First, the show will host the in-depth "Level Design Workshop," an interactive tutorial session examining the best practices for successful level design. Bethesda's Joel Burgess, Valve's Matthew Scott, and The Fullbright Company founder and *BIO SHOCK 2* designer Steve Gaynor will discuss fundamental principles such as layout, flow, pacing, and narrative, and will apply those concepts to a variety of genres.

In the Design Track's "The Business of Art Direction: 7 Critical Precepts," EA Sports executive art director Rick Stringfellow will discuss some essential guidelines that can help developers better manage their production pipelines. He'll examine everything from debugging difficult digital content to establishing new visual processes, and will give attendees the knowledge they'll need to streamline development for all types of games, platforms, and audiences.

Finally, Florian Dhesse of outsourcing house Virtuos will host "Improve Your Games' Quality by Managing Your Designers," offering tips to help Chinese studios lead and nurture their valuable game designers. Drawing from Virtuos's own experience working in China, Dhesse will outline the methods and tools studios will need to improve their products and better manage game design staff.

More information on these sessions is available on the Main Conference track pages on the official GDC China website (www.gdcchina.com). Those interested in attending this November's show can secure their pass now, as online registration is now open. (GDC China is owned and operated by UBM TechWeb, as is *Game Developer*.)



R.I.P. PAUL STEED

PORTRAIT OF A PIONEER

We like to tell ourselves that we're a young industry, but we're not so young anymore. This summer we've been confronted by our own mortality with the sudden death of Paul Steed, a pioneer of real-time 3D graphics who was an icon of the brash early days of the game business. He died on August 11, at the age of 48 (details have not been made public).

Unlike most of us, Steed didn't labor in obscurity. If you were involved in games during the '90s—whether as a professional or as a fan—it was hard not to pay attention to Paul Steed. He worked on some of the seminal titles of the decade, notably the *WING COMMANDER* series and the *QUAKE* series. He produced the first demo for Xbox 360, presented a Career Seminar keynote at the Game Developers Conference, and was a leading exponent of art outsourcing—proving that he could remain topical for nearly two decades. Always outspoken and always controversial, he was not a typical game artist—but he was the most public exemplar of what we do for people both inside and outside the business.

STEED'S START

» Steed joined Origin Systems in 1991 as a self-taught concept artist, fresh out of the Air Force. *WING COMMANDER* producer Chris Roberts recalled that he was hired as a “design assistant”—essentially, an intern—because the studio didn't have the budget for an established artist but liked his hustle. However, Steed found his calling as the studio transitioned to real-time 3D on *STRIKE COMMANDER*. The project was supposed to use sprite-based aircraft with a simple 3D backdrop, and Steed was tasked with creating simple props and buildings to test the infant 3D system. Instead, with his characteristic self-assurance, he modeled and textured complete flyable planes, which convinced the studio that the game could be fully 3D.

Looking at the visuals 20 years later, it's difficult to appreciate the implications of his humble triangular planes. 3D was hardly a novelty—this was, after all, a decade after the debut of *TRON* in 1982. CG-intensive movies like *The Abyss* and *Terminator 2* were everywhere in pop culture. For game developers though, 3D was an esoteric specialty dominated by a handful of academics, scientists, and engineers. The ticket to entry was a finicky UNIX workstation with custom software—an “entry-level workstation” with software in 1991 cost around \$50,000 in today's dollars; for the money, you'd get a machine with roughly the same performance as an iPad 2, but

without smooth skinning, inverse kinematics, ambient occlusion, or normal maps.

For the very few people who could get access to the tools, there were no traditions, no communities, and no shared experiences to learn from or to rebel against. There were no web pages to visit (the first graphical web browser didn't appear until 1993). If you could find a book on computer graphics, it was likely to be something like Van Dam's *Introduction to Computer Graphics*, 600 pages of math and diagrams. If you were wondering about the right way to model an airplane wing, or why a model always had a dark splotch around a particular vertex, the only option was to figure it out for yourself. 3D artists of this generation frequently had a reputation as nerds first and artists second (if at all), since their ability to manage the software came before their sense of style or artistic nuance.

Paul Steed strode into the geeky world of early '90s 3D art with the swagger of a hair-band front man at a *Dungeons & Dragons* convention. Jason “Loonyboi” Bergman, webmaster of the now-defunct LoonyGames web site, described him thusly (www.loonygames.com/content/1.5/feat/):

“On first glance, many thoughts enter your mind. This guy's a construction worker, you think. Chippendales dancer? Maybe military. Enlisted though, definitely not an officer. Or a Rambo-wannabe movie star. He probably spends his



days in the gym, or in the tanning salon. Totally full of himself, you can just see the arrogance in his demeanor. Whatever he does, it surely doesn't involve thinking.”

FLAME WAR VETERAN

» Steed's outsized persona became legendary (or infamous), and his willingness to be seen

and heard made him one of the most prominent voices in games and 3D art. His advice influenced a whole generation of young modelers at a time when there was almost no way to learn the craft. He later published two popular books (*Animating Real-Time Game Characters*, 2002 and *Modeling a Character in 3DS Max*, 2001) but

was best known for his active (and often controversial) engagement with the online community.

Steed was almost a prototype of the now-familiar Internet celebrity. He joined id Software in 1996, when it was the most closely watched company in games. In that year, QUAKE had shown the game industry that real-time 3D, action, and visual immersion were the future of interactive entertainment. Building off the success of DOOM, QUAKE became an online sensation; its online multiplayer spawned a huge, fractious community of fans, mod-makers, and would-be game journalists.

For better or worse, QUAKE fandom set the pattern for a lot of what we now know as Internet manners. The combination of useful information, endless factionalism, and sheer boorishness that we all presume to be part of Internet culture today was honed in arguments comparing QUAKE and Unreal, debates about the relative merits of OpenGL and DirectX, and, alas, in endless torrents of insults and abuse directed at rival players, fans of different games, or anybody unlucky enough to show up in the wrong forum on the wrong day. Somewhere in most of those flame wars you'd find Paul Steed.

The key source of tinder for this early generation of flame wars were the notorious ".plan files." Originally an innocent UNIX workstation feature, .plans published shared status information—so that, for example, a supervisor could remotely see what a group of engineers was working on without needing to walk the halls with his clipboard and pocket protector. However, .plans quickly evolved into a primitive form of Twitter—a medium for quickly sharing thoughts or opinions without considering the implications of making those thoughts or opinions public.

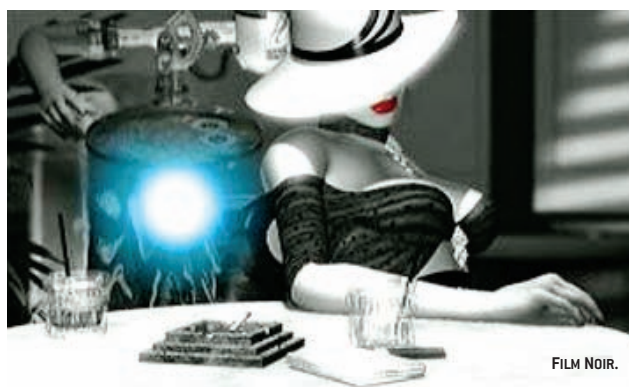
In the mid-'90s, this kind of immediate contact with public

figures was an intriguing novelty. Fans lapped it up, dissecting every update with Kremlinological zeal. Rival developers discussed each other's work, sometimes with admirable frankness and sometimes with unparalleled insensitivity. Fans repeated and amplified these comments, sometimes turning mild technical observations into grist for brutal takedowns.

Steed was a fearsome gladiator of the .plan battlefields, and he was always eager to share his opinions in the most direct way possible. He let readers in on his opinions about games, life as a developer, and life in general. He was always willing to engage with fans online or by email. He once told an interviewer from PlanetQuake (www.quakewiki.net/archives/legacy/interviews/steed.htm):

"...egocentric, self-centered bastard that I am, I keep all mail sent to me in a fan or flame vein... Flame me. I don't really give a shit. I'm confident that I can address or redress the nature of your problem with something I've posted and either pull you over to my side or simply ignore you. The way I feel about it is that if anyone takes the time to write me SOMETHING OF SUBSTANCE I have an obligation to answer them."

His opinions on life at id, on other games, and on life (which seemed to consist largely of alcohol and scantily clad women) were ubiquitous Internet fodder in the late '90s. His online feuds, most famously with Chet Faliszek and Eric Wolpaw—then of OldManMurray.com, now of Valve—were legendary in the gaming community. >>>



A MEMORIAL FUND FOR PAUL STEED'S FAMILY has been established at trentsteed36@gmail.com. His online column is available at www.loonygames.com/content/1.1/totb/. Along with his books, it provides a pretty characteristic look at his personal style as well as some old-school modeling techniques that give the flavor of what life was like when a triple-A character had 800 triangles.



The industry of the '90s was unimaginably open by today's standards [these days, it's hard to imagine many studios allowing employees to publicly criticize rival game studios, fan sites, or their own coworkers]. In 1997 the famous .plan file went dark with this parting comment:

“Restraint and temerity aren't really in my nature so rather than risk any further dishonor or bad press to this company due to yet another outburst from me, I want to go ahead and call it quits with the plannage.”

Two days later it started up again. Those were different times.

Steed's lack of self-censorship made him a PR nightmare, but it also made him a very influential figure in the rapidly growing community of game artists. His modeling tutorials were popular starting places for would-be artists, and he provided feedback and mentoring to many of the beginners who frequented his “Thinking Outside The Box” column on LoonyGames. He was an active participant in the early days of what is now Polycount.com, providing technical advice, critiques, and a little celebrity wattage to aspiring artists. One of the most telling online reactions to Steed's passing was a poster recalling fondly how Steed reviewed some of his models at an Australian trade show and called

them “decidedly mediocre”—a characteristically frank, but useful bit of feedback that inspired the young artist to take a more critical look at his craft.

Even as a mentor, Steed was controversial. He sponsored modeling contests on PolyCount for aspiring 3D artists. Mostly, these involved modeling buxom female characters in various states of undress [one of his online tutorials taught modeling using a Playboy centerfold for reference]. Perhaps his most infamous creation was a contest featuring a scantily clad female model known as the “Crackwhore.” In his defense, he created the model at the request of an all-female QUAKE clan of the same name, but the controversy made it into the mainstream games press, cementing his image as a frathouse bad boy.

STEED'S SECOND STAGE

» Steed's polarizing personality—public and private—became a problem for id. In 2000 he was fired over John Carmack's objections [publicized, of course, via .plan file]. He later called the firing his “exile from Valhalla.” However, he was a savvy careerist, always on the lookout for a new world to conquer. If the first decade of Steed's career encapsulated the growth of both real-time 3D and of Internet culture, the second decade seemed to be dominated by the major themes of the newly prosperous, and newly corporatized game business: the Internet, the next-generation consoles, and outsourcing.

In his second decade in the game industry, Steed pioneered another aspect of the game artist's life cycle: When you're too ambitious for the production line, the next frontier is entrepreneurship. Three milestones stand out in the last decade:

At Microsoft, he was the “creative director”—that is, the primary artist—for the Advanced Technology group that produced the Xbox 360. His 2004 demo “Film Noir” [featuring, naturally, a curvaceous femme fatale] introduced GDC audiences to what was still, in those days, known as “NextGen.”

Not long after that, Steed cofounded an outsourcing art house called Exigent. When a lot of American artists viewed outsourcing with a mixture of disdain and terror, Steed embraced the idea. He told anybody who would listen, including the artists he otherwise worked so hard to mentor, that production in the United States was “finished” and the future belonged to India and China. Like many other outsourcing proponents, he found that outsourcing did not provide a magical solution to development. One of his last jobs was working as a kind of outsourced studio head for UTV Ignition Florida—a project that collapsed in recriminations and bad blood in 2010.



Though he filed some of the rough edges off his public persona as the years went by, Steed remained a very public figure. He was a frequent speaker at GDC, and he became a member of the GDC advisory board,

where he worked to give visual arts more standing. He remained active in education and mentoring. In 2008 he gave the keynote for the GDC Game Careers Summit. The focus of the talk was how to get a job in games, but the advice was more like autobiography:

“Becoming a game industry ‘celebrity’ isn't rocket science—you have to have talent and a good game or three to show it off. Then you give talks, write books, and generally share your experiences with others. The real trick is staying known, staying relevant, and staying excited about what you do... Ambition, hard work, perseverance, luck, and shameless self-promotion—it's all part of the deal.”

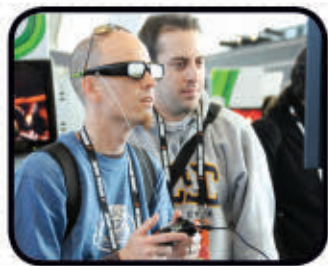
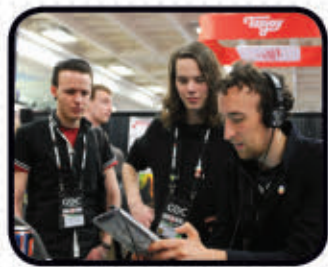
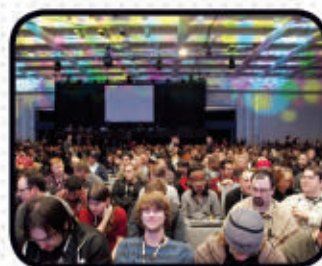
A COMPLICATED COMBINATION

» It would be comforting to have some dignified, impersonal traditions for gracefully summing up when one of our own passes on. Unfortunately, it's harder than trotting out a formula—especially when someone as complicated as Paul Steed is the subject.

For better or worse, Steed's uncensored, outspoken personality is as much a part of his legacy as his artwork—or even the many young artists he helped to inspire. His awkward mixture of crude fanboy culture, real artistic ambition, technical obsessions, and careerist hustle is, in many people's eyes, a central component of our industry's personality. Hopefully we'll be able to prune away the less-attractive parts of that legacy and still hold on to the drive, the passion, and the real generosity that he showed to his fellow artists. 🎮

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.

THE BEST ON-DEMAND CONTENT FROM THE GAME DEVELOPERS CONFERENCE SHOWS



GDC Vault

Streaming video, audio, and
PowerPoint presentations
from GDC 2012, GDC Europe,
GDC China, and GDC Online.

EDUCATION
GROUP RATES AVAILABLE!

For more information visit: WWW.GDCVAULT.COM



DEPTH VS. BREADTH

DOES YOUR GAME NEED TO GO DEEPER?



ULTIMA ONLINE and **EVERQUEST** represent two very different game philosophies. **ULTIMA ONLINE**'s creators tried very hard to create a virtual world with physics and interactions that mimicked the real world, so players could interact with each other in ways meant to model reality: You can chop down trees, dye clothes, build houses, attack almost anyone anywhere, and steal anything that isn't nailed down.

By comparison, **EVERQUEST** is a simple game, not much more than a combat simulator designed to mimic the basics of combat found in tabletop board games and old online Multiuser Dungeons (MUDs). Combat in **EVERQUEST** is very deep and intricate compared to that in **ULTIMA ONLINE**, with far more ways for players to attack and manipulate their enemies. However, combat aside, **EVERQUEST** was perceived to not be a very feature-rich game. Most of the world interactions in **ULTIMA ONLINE** aren't in **EVERQUEST**, and when they are, they aren't particularly deep or fleshed out—to the extent that many observers felt that **EVERQUEST** would

be too simple for the newly invented massively multiplayer genre.

As it turned out, **EVERQUEST** easily beat **ULTIMA ONLINE**'s numbers, and a few years later, a rematch of the two MMO design philosophies paired **STAR WARS GALAXIES** against **WORLD OF WARCRAFT**—with a repeat of the same end result. As it turned out, **ULTIMA ONLINE** has a lot of features, but many of those features don't have a lot of depth to them; it is broad, rather than deep. **EVERQUEST** has fewer features, but a combat model that is very deep (and became deeper as new boss mechanics were added to respond to an increasingly savvy

audience). **EVERQUEST** is a game about depth.

THE PITFALL OF BREADTH

» Most junior designers come into the industry favoring breadth. They want to design the perfect game, and they want to do so by throwing every possible feature under the sun into the design soup. This is especially true in massively multiplayer game design, where the possibilities of what you can do in a game is effectively unbounded—a virtual world can already incorporate almost any feature of the real world. Even worse, most game genre devotees imagine that the perfect

game in their genre is one that combines all of the best elements of other games, because they don't recognize the underlying costs of all of those systems.

Most triple-A games that actually ship (and most experienced designers and producers), however, favor depth over breadth for a few reasons. One is simply a matter of resources—it's hard enough to even do one game system extremely well. If you're trying to make a first-person shooter, for example, you are going to have a hard enough time getting the basics of making a deep and engaging FPS that can be on the same playing field as **CALL OF DUTY** without losing all the

resources and focus on building other systems.

This is especially important when you consider how your multiple game systems are supposed to interact with each other. For a game that is about breadth, the multitude of game systems can interact with each other in many often-unexpected ways. In some cases, this can be a good thing—something that designers like to call emergent behavior. Emergent behavior can be wonderful to behold, as the fans will undoubtedly surprise you with their ingenuity. However, the more systems you have, the more time you'll need to spend on QA and balance,

especially if you're making competitive online games, where that ingenuity can be used to win or to harass other players. The same game freedom that allows players to build amazing things like pianos by stacking items on top of each other in *ULTIMA ONLINE* also allows them to build staircases of spoons up to the top of other players' castles, enabling them to waltz in and rob them blind.

BREADTH CAN BE GOOD

» This is not to say that all games that are breadth-first are doomed to fail. *ULTIMA ONLINE* and *STAR WARS GALAXIES* both had sizable fan bases, largely because the two games were wild and unpredictable places where it felt like anything could happen. The idea that you can go anywhere and do anything is an attractive sales pitch, and it is one that two companies in particular have been remarkably effective at building toward—*Rockstar* (*GRAND THEFT AUTO*) and *Bethesda* (creator of *OBLIVION*, among others).

GRAND THEFT AUTO is the poster child of the breadth game, and it illustrates the

shooting that pales next to *CALL OF DUTY*, fighting that has nowhere near the tightness of control of *SOULCALIBUR*, and yet it is still widely considered one of the finest games ever made. Why? Because all these things blend into each other—you can crash your car, then get into a gunfight that ends in an epic fistfight. *GRAND THEFT AUTO* is, at its core, broad and open-ended wish fulfillment of the idea of a world without rules; you can go anywhere and do anything you want. The possibilities are endless, and it captures the imagination like few other games can.

Of course, *GRAND THEFT AUTO* also shows how expensive that kind of broad world interaction can be; development costs of *GRAND THEFT AUTO IV* exceeded \$100 million, according to Wikipedia. That said, the same source says *Take Two* earned five times that much in income in the first week the product was on sale. But few companies can afford that initial price tag.

SIMPLICITY IS KEY

» In the context of breadth vs. depth, the concept of simplicity is vastly

web site lists postman, kart driver, and soccer star among more classic combat roles like ninja and warrior. Each of these games has smaller minigames associated with them, with their own respective advancement tracks.

WIZARD 101, by contrast, was made by a smaller, scrappier studio (King's Isle). They didn't have the resources to make a broad game, and instead focused on making an intricate combat model reminiscent of *POKÉMON* and *FINAL FANTASY*. While the world itself is simple, the combat model is not—the game designers went out of their way to design a game with a long life, and with angles of expansion so they could continue to put out content that would be in demand on the microtransaction store. Their tactic appears to have been successful—*WIZARD 101* got its 20 millionth user in 2011, two years and two months after the game was launched, while *FREE REALMS* took four months longer to hit the same milestone.

PLAN TO EXPAND

» One of the interesting angles of the depth vs. breadth debate, especially in terms of online games with ongoing support and development, is the appetite for expansion. If you have one central game activity, then it is very easy to focus your development on that game activity—adding more creatures to summon in *WIZARD 101*, for example, or adding new raid mechanics to *WORLD OF WARCRAFT*.

By contrast, broad games have a broad player base as well, and all of them want more and more interesting things to do when they consume their favorite kind of gameplay. In *ULTIMA ONLINE*, crafters and tamers demanded

as much design attention as combatants, and I'm sure the same can be said in *FREE REALMS* for fans of the soccer and mail delivery games. It is harder to improve the game on multiple tracks, and keep all fan bases happy while still maintaining the game's core balance integrity. Adding new features and game systems to increase the breadth of the game is always an option, and is typically popular, but it also risks increasing the complexity of the game—increasing the number of unexpected interactions that need to be considered, balanced for, and tested.

THE BROAD-GAME BULLET LIST

» Making a broad game isn't impossible, and some of the finest video games ever made are broad games. But here are some things to consider.


1// Don't think of your features in a vacuum. Think about how they interact with each other, how they balance against each other. Be sure they support each other (at best) and don't compete or invalidate each other. In *OBLIVION*, the desire to learn by doing to complete a sandbox experience resulted in players hopping through fields picking flowers in order to become master assassins (by increasing their jumping and poisoning skills).

2// Think of the audience. Any feature adds complexity to your game and takes development time. Yes, you could add washing laundry to your *GTA* clone, but would it hold anyone's interest for any length of time?

Does it support the core fantasy of the game?

3// Accept that your features will be simpler. Complexity in multiple systems is hard to support, hard to test, hard to balance, hard to expand, but perhaps most of all, hard for most players to actually follow and understand. In a broad game, the game systems need to be simple—the complexity will come from how those systems interact.

4// Be willing to lose control. Embracing emergent behavior means you are accepting on some level that players will surprise you. Your design team and management needs to decide up front if this is a good thing or not.

Broad games are hard to do, and really hard to do well. There is a reason that most broad games that get started never ship, and that many established developers flee them, philosophically. But still, I would sorely like to see more broad games on the market. Games like *ULTIMA ONLINE* and *GRAND THEFT AUTO IV* were exciting because they challenged their players to imagine the possibilities. If done well, this promise speaks right to the core of what makes video games amazing. 

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.



difference clearly. It has auto racing that fans of *NEED FOR SPEED* would sneer at, car crashes that are snoozers compared to *BURNOUT*, a story without the choice or depth of a BioWare game,

misunderstood. *WIZARD 101* and *FREE REALMS* are two lighter MMOs designed for a younger audience. Sony Online's *FREE REALMS* includes a series of minigames; today, the



You can bring an Orth to water...

ADAM ORTH JOINS MICROSOFT BY WAY OF POPCAP

IN THE PAST FEW YEARS, MANY RESPECTED GAME DEVELOPERS HAVE MIGRATED TO POSITIONS IN THE CASUAL/SOCIAL SPACE—ADAM ORTH WAS ONE OF THEM. IN MARCH 2011, HE MOVED TO CASUAL GIANT POPCAP, BUT BEFORE THAT HE WORKED AT SONY, EA, AND LUCASARTS. BUT HE'S SINCE LEFT POPCAP, AND IS NOW WORKING ON AN UNANNOUNCED PROJECT FOR HIS NEW GIG AT MICROSOFT. *GAME DEVELOPER* CHECKED IN WITH ADAM TO ASK ABOUT THE EXPERIENCE OF MOVING FROM CORE DEVELOPMENT TO SOCIAL/CASUAL, AND BACK AGAIN.



Alexandra Hall: What did you learn at PopCap that applies to your return to core-oriented games?

Adam Orth: I learned so much about social and casual games during the year I spent at PopCap. I've always been interested in that space, and take great care to try and be as current as possible, although I wasn't nearly as prepared as I thought I was. PopCap is full of experts and academic-level developers. I often refer to it as gaming grad school. I think I actually learned more about core game design than social/casual. It's the basis for all the PopCap magic. The knowledge I gained from being able to touch things like BEJEWELLED, PEGGLE, ZUMA, and PLANTS VS. ZOMBIES on various platforms was incredible, and I refer to those methodologies and principles every day. I'm very lucky and grateful for the experience.

AH: Do you think core game developers should put in time on the social/casual beat?

AO: Absolutely. It's a very different style and approach to making games. Being fluent in that language is only going to make you, your team, and your game stronger. Even if you aren't going to make a game in that space, it's crucial at this point in the evolution of the industry to be an expert, otherwise you are going to be passed by.

AH: Do you see any significant differences in workplace culture between core game development and social/mobile games development?

AO: Obviously, when there is a corporation involved, that is going to dictate the culture. The DNA of the corporation permeates everything and drives the creation of multiple products. Start-ups and independent studios are completely different because of the risk—it's very much like being in a band, with everyone focusing all of their energy on the one thing. All or nothing.

I've done both and I can't say which I prefer. It depends on the end goal. I've enjoyed the safety of a steady paycheck and the electricity of not knowing what's next. The only cultural point that matters to me is dedicated developers working hard every day to make awesome interactive magic. You can have that anywhere, you just have to be realistic about where you are.

AH: You're working on an unannounced Microsoft project. Can you talk at all about the experience of working in a newly formed internal studio?

AO: I can't really say much other than I love what I'm doing. It's very different from anything I've ever done, and I was desperate for something like this at this point in my career. Microsoft is very big, and being a small entity within a giant machine can be terrifying, but I kind of love it. It's unpredictable and exciting. It takes a lot of work to get everyone talking together, but the talent level and drive of the people I'm working with make the results amazing. There's always the risk that whatever you are working on will never see the light of day, but if you start thinking like that, it's already over. You've got to believe passionately in what you are doing and who you are doing it with, otherwise it's pointless to get out of bed in the morning. That's where I am right now. **GD**

who went where

/// Rovio has hired former Easy Studios general manager Oskar Burman to head a new studio in Stockholm. Also joining Burman is Patrick Liu, previously BATTLEFIELD 3 producer at DICE, who will be signing on as creative director.

/// Ito Masahito recently resigned as president of shoot-'em-up luminary CAVE Interactive. His is the second high-profile departure in recent months, following former VP and COO Mikio Watanabe.

/// Epic Games's former director of production, Rod Fergusson, has transitioned over to Irrational Games to help the studio ship the anticipated first-person shooter BIOSHOCK INFINITE. The project has seen a number of key departures over the past year.

new studios

/// Many of the Maryland-area developers left unemployed in the wake of the 38



Studios debacle have landed on their feet, thanks to a little help from Epic. The inaugural project

of newly formed Impossible Studios is the upcoming iOS title INFINITY BLADE: DUNGEONS, which Impossible is codeveloping with Epic and Chair Entertainment.

/// Online shopping giant Amazon.com is stepping into casual game development with the founding of Amazon Game Studios. Its first game, LIVING CLASSICS, is already available on Facebook. The studio joins Amazon's other game-related initiatives, such as GameCircle, a service for Kindle Fire gamers, and Game Connect, which lets customers download free-to-play games.

/// Activision recently founded The Blast Furnace, a new mobile development studio in Leeds. The studio will be headed by Mark Washbrook and Gordon Hall, both longtime veterans of Rockstar's UK studios.

/// In the wake of Zipper Interactive's closing in April, former employees David Kern and Russ Phillips have started Nobodinos [say it: "nobody knows"], a new studio that will focus on mobile titles.



CROWDFUNDING AND EMOTIONAL EQUITY

YOU LISTEN TO YOUR BACKERS, BUT DO YOU REALLY HEAR THEM?

In my column in the April issue of *Game Developer*, I discussed crowdfunding. One of the things I covered last time around was the fact that Kickstarter projects have no obligations to backers (beyond rewards promised), but that other types of crowdfunding mechanisms that would allow for crowdfunded equity investing were on the way. Whether backing comes via crowd-sourced equity or via Kickstarter-style donations, all crowdfunding will share a second type of investment—that of emotional equity.

PUTTING THEIR MONEY WHERE THEIR HEART IS

» As others have pointed out, there is more going on in an end-user backing of a project than just a quid pro quo exchange of money for a reward package. Certainly, in some cases that's all it is. In most cases, though, the backing of a project represents an emotional investment by the backer. The project is something they believe in and want to see happen, and they are ponying up money to state that this is the case.

This emotional investment is a powerful force. It's what can make backers not just a source of funding, but passionate evangelists, dedicated contributors, and loyal return customers. They believe they have a personal stake in your game and will do what they can to ensure its success.

However, like any other type of equity, this emotional equity is given with expectations—not always clearly stated—and it's here that things can go awry if not given proper consideration.

When you look at examples of this disconnect, you see they often mirror the very same things that can go wrong with a standard investor relationship, or for that matter, any relationship. A few examples:

- “You’re late!” It’s so commonplace for Kickstarter projects to take longer than their optimistically set schedules that I’m surprised people don’t just take it for granted. That said, it happens often, and when it does, backers are amazingly forgiving—provided they are given some background as to why, and are kept up to speed.
- “You said you’d call!” Another source of backer frustration is dead air. Backers feel they bought “insider status,” and get frustrated and nervous when updates grow further apart. This is often compounded by the schedule issues mentioned above.
- “You’ve changed!” The reality of developing any product is that things change along the way from conception to delivery. One of the risks of taking money based on the early-stage idea is that it may change along the way, and that may change something critical to the person who decided to back it. For example, when the wildly successful Kickstarter for the Ouya console announced that they’d be making the OnLive cloud-gaming solution available for playing mainstream titles, some backers were elated. Others viewed it as an about-face on the original indie-focused-console message that they’d laid down money to support. (For example, UBM’s Simon Carless tweeted, “I know OnLive is ‘just an option,’ but Ouya’s Kickstarter traded heavily on indie cred.”) That OnLive’s long-term viability is now in question only compounds the point.
- “You scoundrel!” Of course, it’s never good to pursue any unethical or illegal business tactics, period. However, in the case of crowdfunded projects, doing so can cause backers to feel you dragged them down with you. The folks at Penny Arcade recently blogged about receiving a “rewards for reviews” proposal from the developer of the EPIC SKATER Kickstarter campaign, where the developer offered them two \$125-tier rewards in exchange for a favorable review, which Penny Arcade characterized as “bribes for coverage.” Even if we give the developer the benefit of the doubt


and call it “naive and questionable marketing tactics,” the damage was done. At least one backer pulled their funding (based on comments on the Kickstarter page), and I expect more will do so. (You can find the Penny Arcade article here: <http://penny-arcade.com/report/editorial-article/bribery-spam-and-harassment-the-dark-side-of-kickstarter-promotion>).

If things go wrong, they can go really wrong. It’s no secret that the internet can amplify the scale and speed at which things get out of control when they go negative. For example, the Music ensemble Classic Crime funded a Kickstarter (www.kickstarter.com/projects/mattmacdonald/the-classic-crimes-new-album/posts/188274) for an album and tour, but what started as a miscommunication about the cost of rewards and costs of touring escalated into flame wars and damage control, as the band tried to deal with critics accusing them of taking in unreasonable amounts to fund their tour compared to what some believed it should cost.

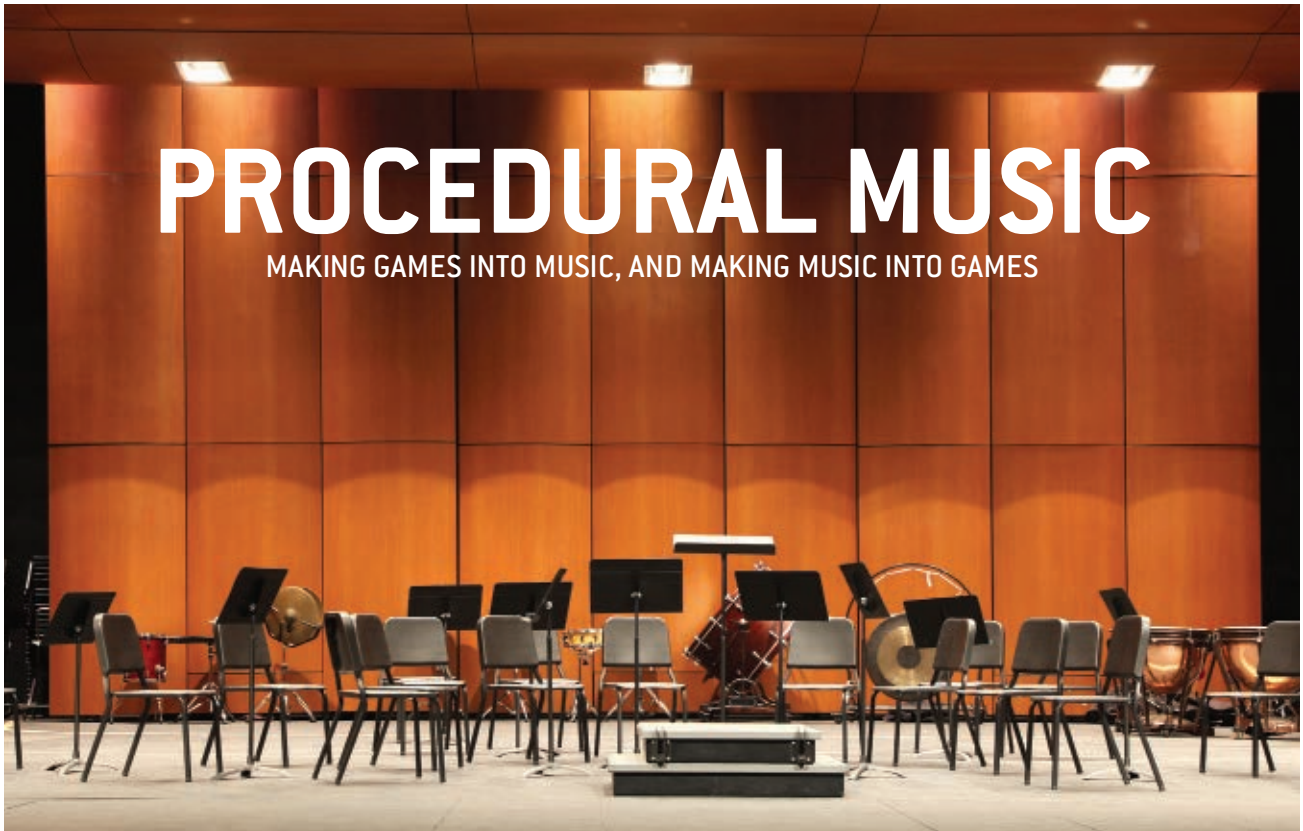
NEW FUNDING, SAME OLD RULES

» Even though the dollar amounts are smaller, and there isn’t a publishing or investing contract attached to the dollars, the same rules apply as if they were.

- **Communication is key.** Communicate early and often, letting backers know about changes before they happen.
- **Come clean.** Give background on why something is late, changing, up in the air, or whatever it might be. If there are still unanswered questions, acknowledge them. People prefer the unvarnished truth to polished spin.
- **Solicit input.** People are already vested in your product. Giving them the chance to further contribute is often a great way to increase that level of passion, even when it’s wrapped around a negative circumstance.
- **Give people options.** If the product is changing substantially, or the schedule is pushing out a significant amount of time, offer to refund their backing. They’ll likely not take you up on it, but will feel better for having been given the option.

Just because the money came through a click instead of a handshake doesn’t mean it doesn’t deserve the same level of respect and stewardship as any other deal. Consider your backers’ emotional equity and respect it as such and you’ll be rewarded for it in the long run. 

KIM PALLISTER works at Intel doing game industry forecasting and requirements planning. When not prepping the world for super-cool hardware, he blogs at www.kimpallister.com. His views in this column are his and do not reflect those of his employer.



PROCEDURAL MUSIC

MAKING GAMES INTO MUSIC, AND MAKING MUSIC INTO GAMES

If I were to describe a composition system that generates music by using a set of algorithmic parameters which are then reinterpreted by performers, many people would lose interest pretty quickly—in fact, I may have lost many of you by the time I got to “system.” However, if you replace “music” with “audio” in that sentence, I am effectively describing every single game audio system ever made. In some ways, talking about procedural music with game developers is preaching to the choir; game developers already work with this kind of thing regularly while working on game audio, but it’s worth looking into how we can make procedural music composition systems that can inform your game design (and vice versa).

NEW WORDS, OLD IDEAS

» Procedural music, also called generative or algorithmic music, is all about creating a set of rules that will dictate the music that is created. This is, of course, not dissimilar to game design, which is about creating a structure and letting players explore and make meaningful decisions. You have to account for loads of different scenarios, and the game is ultimately controlled not by the designer at all, but by the player. In the world of music composition, this isn’t anything new. Composers are often challenged or excited by the prospect of performers taking their music and turning it into something new.

This problem/opportunity (proletunity?) has existed as long as people have been writing down music. An early approach to

procedural music was the so-called Musical Dice Game occasionally attributed to Mozart, but generally popular among composers and players in the 18th century. In this game, the sheet music would be split up into a number of different musical phrases. Participants would roll the dice a set number of times, and use those numbers to cobble together a brand-new piece of music. While calling this a “game” is a stretch, it was an interesting way to get a nearly infinite number of variations out of a single piece of music.

Jumping forward a few hundred years, many forms of jazz are arguably procedural music, particularly the more improvisatory genres such as bebop or Dixieland. In these formats, often a simple melody serves as the basis for enormously complex variations, which are

entirely left up to the players. They are playing within the framework of the song’s chord changes, tempo, and rhythm, but otherwise they are exploring on their own.

Creators of 20th-century classical music became quite obsessed with the idea of procedural music, to the point where the elements of the composition that were left undefined could become the most important. John Cage demonstrated this with a number of different pieces, such as “As Slow as Possible,” the length of which can vary wildly (one ongoing performance will last for 639 years). His piece “Fontana Mix” took this approach even further, with the instrumentation defined as “any number of tracks of magnetic tape, or for any number of players, any kind and number of instruments.”

DESIGNING MUSIC, COMPOSING GAMES

» We can begin to think about what music can bring to game design by taking traditional composition methods and applying them to games. Both game design and music composition are essentially based around a series of choices, and both in many cases involve some sort of start, end, and cycle. As a counterpoint teacher once explained to me, once you have written a note of music you only ever have four choices. You can play that note again, you can play that note again differently, you can change the note, or you can not play any note. Similar sets of choices inhabit the world of game design, and there are similar techniques for simplifying the process as well. One particularly good example is

the technique used in the writing of a fugue, which involves taking a single melody and overlapping it on itself in varying ways. Most fugues involve three or four voices, each one playing the same core melody and cycling through variations. I always like thinking of the four fugal melodies as being like four different characters exploring the same space, perhaps finding different power-ups that change their speed, double their power, and so on.

In terms of a more modern composition technique, the rise of looping and sequencing as a musical force lends itself quite nicely to game design. One prototype we've developed here at Lucky Frame is a drum-machine space shooter. As the enemies come onscreen, they generate a drum sound, and as they are destroyed, they play a musical tone. If the levels are editable, this instantly creates a sequencer and drum machine that is playable and remixable by controlling a flying spaceship. It's really the best way to DJ [at least until it is possible to do that with actual spaceships, anyway].

So, now that you're as excited as I am by the intersection between music and game design, I hear you asking: "What are the three main things to be aware of?" Good question! My answer is: complexity, generation, and reproducibility.

COMPLEXITY

» The most basic form of generating music in a game is to attach a musical event to a game action, such as a character's movement. This can be very powerful and direct, and provides the player with crucial instant feedback, making the direct connection between action and music. However, music generation needs to have depth. This is no different from an instrument; without complexity, it is repetitive noise that will soon become stale (then boring, then irritating, then abandoned), but if the music that is being created changes too much, the connection between the action and the music will be lost.

For example, if a character will trigger the same musical

note whenever they cross a tile, that could get old very quickly (particularly if you have to cross that tile several times). However, if the character triggers a different note every time they cross that tile, the player will probably not notice the connection between the tile and the playing of the note. In short, you need to strike a balance that creates a strong connection between an action and a musical event, without making it repetitive and robotic. That balance is what can bring rewarding complexity to a musical game.

GENERATION

» Many (if not most) games now rely on some generative element, ranging from fully generated levels to character-name generators. This approach should absolutely be mirrored in the world of procedural music. If you set up a system that generates the musical framework within which a game is played, the music will almost certainly have more depth and potential. In our iOS game PUGS LUV BEATS, we approached this by procedurally generating the planet terrains, which controlled all the sound libraries. This ensured a potentially infinite amount of musical combinations.

REPRODUCIBILITY

» A simple (but extremely important) principle of any musical device is the idea that doing the same thing twice will result in the same output. Pressing the same piano key in exactly the same way twice will sound nearly the same. Turning the filter knob on a synth will have the same effect every time. Without the ability to rely on that process, the instrument would be impossible to master. This is, of course, very similar to game design, and as such any musical games need to have as much reproducibility as possible. The challenge with this concept is being able to retain the elements of mystery and challenge that any game needs.

THE GAME OF MUSIC

» I have thus far framed many of these concepts in the context of creating a musical game; music has much to offer game design as a whole. Even the most basic concepts

of rhythm and timing can be seen as a truly integral part of a gameplay experience—one that the best game designers may already be incorporating even subconsciously.

But it is equally important to note what the game world can offer music. This can be looked at both from a creation and an enjoyment level. Games are inherently social and fun experiences, with the vast majority of people playing games without any pretense of becoming the best, or even worrying about how good they are. It's very rare nowadays for anyone to say that they are "not good at games"—virtually everyone plays and enjoys

to game design, and it shows no sign of slowing down. Musically, it can be argued that people who would have become composers in the past are now designing software and games; these days, the most exciting developments in music are forms of interface, such as the Monome input device, or software like SOUND SHAPES for PS Vita.

It should be clear by now that I see music and games as two extremely similar things. They both have much to offer each other, both conceptually and practically. Too often they are seen as two separate things that need to be forced to work together, but if they



some form of game. There is no reason why music should not be the same way. Playing music is something that everyone can and should do, it should be a fun, social activity not reserved for experts.

In terms of the creative side of things, game design (and to a certain extent software design in general) is in many ways the most vibrant art form in the world right now. An enormous range of styles and approaches are being applied

are approached together they can definitely become greater than the sum of their parts. 🎮

YANN SEZNEC is an artist, musician, performer, and founder of creative studio Lucky Frame. Recent releases for iOS include the IGF-nominated PUGS LUV BEATS and the critically acclaimed BAD HOTEL, both of which use innovative procedural music techniques. Lucky Frame is based in Edinburgh, Scotland, and is currently working on several new interactive music projects.



REFRACTION

[HTTP://GAMES.CS.WASHINGTON.EDU/REFRACTION/REFRACTION.HTML](http://games.cs.washington.edu/refraction/refraction.html)

THE HISTORY OF "EDUTAINMENT" IS LITTERED WITH FAILED GAMES THAT WEREN'T FUN ENOUGH TO DELIVER THEIR EDUCATIONAL PAYLOADS. THE POST-GRAD TEAM BEHIND FRACTION-TEACHING PUZZLER REFRACTION IS DETERMINED TO NOT BE ANOTHER HISTORICAL FOOTNOTE—WHICH IS WHY THEY'VE ALSO CREATED "PLAYTRACER," A PIECE OF ANALYSIS SOFTWARE THAT TRACKS PLAYERS' ACTIONS AND TRANSLATES THEM INTO VISUALIZATIONS THAT THE DEVELOPERS CAN USE TO BALANCE REFRACTION'S DIFFICULTY, EDUCATIONAL POTENTIAL, AND FUN.

Alexandra Hall: *What inspired you to make an educational game about fractions?*

Yun-En Lu: I've always liked teaching. Helping out in an elementary math class is a lot of fun. Every student has a slightly different way of looking at things, and trying to come up with examples that students can reason through to find conceptual truths is a really interesting challenge. Games are a natural way for this to happen, since they're all about exploration of strategies and interactions between complex rule sets, so it seems only natural that we could let students experiment with math

and try to make as large an impact as possible on this specific topic before tackling other areas. Our goal is to create games and tools to supplement existing classroom curricula and make teachers more effective.

AH: *You've worked on REFRACTION for a couple years now—unusual for a student game. Have your goals shifted?*

YL: As Ph.D. students, creating REFRACTION and its sister games is more like a job than school—we're paid, we have funding, we take very few classes, and so on. Since we can devote a lot more time to these games than

to iterate and improve the game, and study its effectiveness. We're actually working on an entire set of fraction games, two of which are nearly ready for initial release. Each of these games can tailor the play experience to each player individually through procedurally generated levels and progressions along with sophisticated analysis tools to measure individual student learning.

AH: *Is it tricky to find a balance between fun and educational value?*

Erik Anderson: Difficulty balancing for educational games is not really that different from other games. All games involve learning, even those without an explicit educational purpose. However, one key difference of designing educational games is that our learning goals restrict our choices of game mechanics. Although game designers sometimes bend reality in order to improve usability or playability, we have no choice but to implement target educational concepts exactly as they appear in the real world.

AH: *Did Playtracer spring from a desire for finer-grained difficulty and experience-tuning?*

EA: We developed Playtracer because we needed a tool that could effectively visualize player behavior in games without a virtual environment. Heat maps are highly effective for games such as first-person shooters because a game designer can mark the map each time a player dies, for example, and determine the most dangerous sections of that map. However, for puzzle games like REFRACTION, there is no equivalent "map," so we need some other way to visualize player actions. Therefore, we

Publisher/Developer

Center for Game Science

Release Date

September 2010

Development time: ~1 year

Development budget: \$150,000

of lines of code in the game:

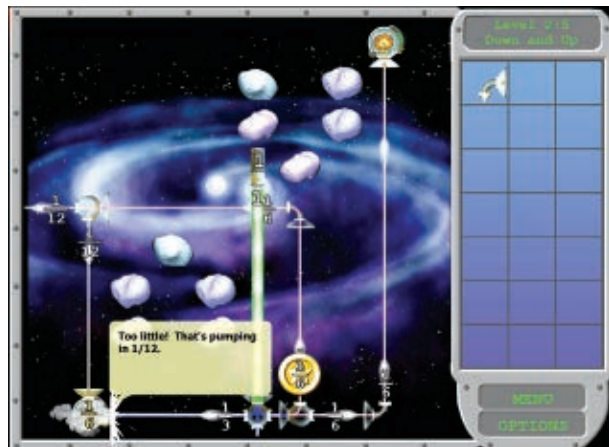
~100,000

A fun fact: Players have commented on the difficulty of getting the coin on level 2.5 more than any other. In fact, Michael John (senior creative director at EA) thought it was impossible to obtain when he first played through the game.

created a tool that shows how hundreds or thousands of players move through a game's state space by creating graphs in which circles represent game states and arrows represent player actions. PLAYTRACER uses multidimensional scaling to ensure that similar game states are shown close to each other, and vice versa. This allows game designers to rapidly find where players are getting stuck and what strategies they are using.

AH: *Are you planning to offer Playtracer as a tool for use by other developers?*

EB: Yes, we plan to make our Playtracer tool publicly available in the future. In the meantime, we have published the technical details of Playtracer in a few different research papers, allowing developers to integrate these ideas and methodologies into their own tools. None of the ideas are particularly complicated to implement; we know a few developers who wrote their own version of Playtracer to analyze their games. We are also currently working with some external developers to plug their games into our in-house infrastructure. (You can read more about Playtracer in this paper: bit.ly/playtrcr)



in the same way using games. It's my hope that through our games we can convince a generation of kids that everyone can learn math if they put their minds to it—and even find it fun.

Eric Butler: Why fractions specifically? The original plan for our group was to generically make educational games about science, but we quickly discovered that basic algebra and mathematics was a huge stumbling block for a lot of students. So we decided to focus very narrowly on the early education problem of fractions

undergraduates taking a game course could, it also means we can go much deeper into the really interesting problems, such as running experiments to see how players react to different tutorials or optional rewards, trying out algorithms to predict what people are likely to do so we can adjust the game experience for them, creating new data visualization and analysis tools, and so on.

EB: The project cannot really be called a success until we make a measurable impact to early math education. So we continue



|CDM

BECOME A LEADER IN DIGITAL MEDIA

With digital media in mind from conception to completion, the new CENTRE FOR DIGITAL MEDIA features student apartments, project rooms and classrooms all designed to inspire creativity and collaboration. Located in Vancouver, Canada the new CENTRE FOR DIGITAL MEDIA offers a full and part-time Master's program that focus on real-time, industry-facing collaborative projects.

Learn more about our MASTERS OF DIGITAL MEDIA PROGRAM and EXECUTIVE MASTERS OF DIGITAL MEDIA PROGRAM at www.thecdm.ca/programs

The future of work is at the new CENTRE FOR DIGITAL MEDIA.

CENTRE FOR DIGITAL MEDIA | www.thecdm.ca

gd

TELEVISION

GAME DEVELOPER MAGAZINE

the best of postmortems,
product reviews, and
standout columns

GET THE
PRINT+DIGITAL
ACCESS BUNDLE FOR ONLY

\$49.95 /YEAR

- + DIGITAL ACCESS TO BACK ISSUES
- + EXCLUSIVE INTERACTIVE EXTRAS

INCLUDES:



PRINT
SUBSCRIPTION



DIGITAL + GAME
DEVELOPER APP

+

BONUS!



BEST OF
POSTMORTEMS
PRINT ISSUE

SUBSCRIBE TODAY!

GDMAG.COM/SUBSCRIBE





OUR STUDENTS MAKE VIDEO GAMES IN THE HEART OF HOLLYWOOD!

DesignLAFilm.com
800-406-7485



WHERE OUR GRADS WORK

ACTIVISION BLIZZARD
SQUARE-ENIX
DISNEY INTERACTIVE
BANDAI NAMCO
TAKE-TWO INTERACTIVE
G4

TESTING
GAME DESIGN
SOUND DESIGN
PROJECT MANAGEMENT

PROGAMMING
MARKETING
LEVEL DESIGN
ART ANIMATION

WINDOVS 7 ULTIMATE
MICROSOFT OFFICE
ADOBE CREATIVE SUITE
AUTODESK MAYA

WORK HARD!

LEVEL DESIGN
game play!
SPITCH!
PROJECT MANAGEMENT
SOUND DESIGN

INTEL CORE 2DUO
ATI PREPRO
OLD

SWIPE TABLUD
NEW GRADS!

GO
HOLLYWOOD
LOS ANGELES FILM SCHOOL
SUCCESS
DEGREE

GAME DEVELOPMENT

For more information on our programs and their outcomes, visit www.lafilm.edu/outcomes. ©2012 The Los Angeles Film School. All rights reserved. The term "The Los Angeles Film School" and The Los Angeles Film School logo are all trademarks or registered service marks of The Los Angeles Film School. Accredited by ACCSC.

UNITED STATES POSTAL SERVICE Statement of Ownership, Management, and Circulation

1. Publication Title: gd Game Developer. 2. Publication No.: 13782. 3. Filing Date: August 30, 2012. 4. Issue Frequency: Monthly with a combined June/July issue. 5. Number of Issues Published Annually: 11. 6. Annual Subscription Price: \$49.95. 7. Complete Mailing Address of Known Office of Publication (Not Printer): United Business Media LLC, 303 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107. Contact Person: Roy Beagley. Telephone: 203-775-9465. 8. Complete Mailing Address of Headquarters or General Business Office of Publisher (Not Printer): United Business Media LLC, 303 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107. 9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor: Publisher: Simon Carless, United Business Media LLC, 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107; Editor: Brandon Sheffield, United Business Media LLC, 2nd Street – Suite 900 South, South Tower, San Francisco, CA 94107; Managing Editor: None. 10. Owner: United Business Media LLC, 600 Community Drive, Manhasset, NY 11030-3875, an indirect, wholly owned subsidiary of United Business Media LLC, Ludgate House, 245 Blackfriars Rd., London, SE1 9UY, U.K. 11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities: None. 12. Tax Status: Has Not Changed During Preceding 12 Months. 13. Publication Title: Game Developer. 14. Issue Date for Circulation Data Below: September 2012.

15. Extent and Nature of Circulation:	Average No. Copies Each Issue During Preceding 12 Months	No. Copies of Single Issue Published Nearest to Filing Date
a. Total No. Copies (Net Press Run)	28,463	25,415
b. Paid and/or Requested Circulation		
(1) Outside County Paid/Requested Mail Subscriptions Stated on Form 3541.	18,122	16,767
(2) In-County Paid/Requested Mail Subscriptions Stated on PS Form 3541	0	0
(3) Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid or Requested Distribution Outside USPS	1,457	1,265
(4) Requested Copies Distributed by Other Mail Classes Through the USPS	0	0
c. Total Paid and/or Requested Circulation [Sum of 15b. (1), (2), (3), and (4)]:	19,579	18,032
d. Nonrequested Distribution (By Mail and Outside the Mail)		
(1) Outside County Nonrequested Copies Stated on PS Form 3541	5,423	5,248
(2) In-County Nonrequested Copies Stated on PS Form 3541	0	0
(3) Nonrequested Copies Distributed Through the USPS by Other Classes of Mail	0	0
(4) Nonrequested Copies Distributed Outside the Mail (Pickup Stands, Trade Shows, Showrooms, and Other Sources)	3,145	1,800
e. Total Nonrequested Distribution	8,568	7,048
f. Total Distribution (Sum of 15c and 15e)	28,147	25,080
g. Copies Not Distributed	316	335
h. Total (Sum of 15g and 15h)	28,463	25,415
i. Percent Paid and/or Requested Circulation [15c Divided by 15f Times 100]	69.56%	71.90%

16. Publication of Statement of Ownership: This Statement of Ownership will be printed in the October 2012 issue of this publication. 17. Signature and Title of Editor, Publisher, Business Manager, or Owner (signed): Simon Carless, Date: August 30, 2012.

WELCOME TO THE
WORLD OF AIE

Last chance to apply for 2012-2013 Scholarships

AIE ACADEMY OF INTERACTIVE ENTERTAINMENT
THE EDUCATION EXPERTS IN GAMES, 3D ANIMATION & VISUAL EFFECTS
SEATTLE - LAFAYETTE - SYDNEY - MELBOURNE - CANBERRA

www.theaie.us
206.428.6350
Our non-discrimination policy can be found online

ADVERTISER INDEX

COMPANY NAME	PAGE	COMPANY NAME	PAGE
ACADEMY OF INTERACTIVE ENTERTAINMENT	55	LUCAS FILMS	C2
BLIZZARD ENTERTAINMENT	6 & 16	MASTERS OF DIGITAL MEDIA PROGRAM	53
EPIC GAMES	15	RAD GAME TOOLS	C4
HAVOK	C3	TWOFOUR 54	3
LOS ANGELES FILM SCHOOL	54	VANCOUVER FILM SCHOOL	25

gd Game Developer (ISSN 1073-922X) is published monthly by UBM 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 UBM 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: \$59.95; all other countries: \$69.95 (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/UBM LLC, unless otherwise indicated. Don't steal any of it.



THE PR COACH

INSIDE THE PUBLIC RELATIONS DEPARTMENT'S SECRET TRAINING SESSIONS

Preparing for an upcoming media event is hard work—you never know what devious curveballs a hard-hitting game journalist might lob your way. Thankfully, the big publishers often have professional, highly skilled PR staff who can advise and train game developers in the art of sticking to talking points and saying things correctly. Let's take a peek behind the curtain to see how this mysterious process works!

TRANSCRIPT #1

I've coached politicians, lawyers, CEOs—people in highly visible, highly scrutinized positions. So I'm pretty sure this will be fairly simple.

Hey there, yeah...uh, I'm not really sure why I'm here, to be honest. I already know how to conduct myself in front of an

audience. I just gave a big talk to my screaming fans last E3.

Yes, I watched that video. To be fair, all you actually said out loud at that E3 presentation was "DEMONS WITH GUNS 2 is coming this winter."

And they loved it. Right? The crowd went wild.

In that case, this should go fast. Why don't we begin by pretending I'm a game journalist.

I don't know. I think you're too hot to be a game journalist...a real one, anyway. I'd be, like, "Are you really into games? Probably not. You're just doing this because of all the free attention you get." That's a thing that happens in the game business, see—there are these girls who aren't really into games, they just—

Okay, slow down there. What did you just say?

Huh? That was just a little joke. Anyway, I said you were hot. You should feel complimented!

TRANSCRIPT #8

You still don't hear it? Let me play it back again. Okay? I'm going to rewind it and play it back again.

[faint audio]: "Easy mode is very forgiving. It's so simple your girlfriend could play it."

Okay, one more time. Tell me what's wrong with that.

I...I don't know.

You don't know? Think about it for just a little longer...

Uh, uh...is it because I'm assuming everyone has a girlfriend? I mean, I guess if someone was gay, they might get mad at me?

Keep thinking.

TRANSCRIPT #20

...and I don't know about those people who are into those anime-style games. I think they're biased, and it's not fair. Because people are, like, "Oh, it's awesome." But it's not. That anime stuff is all just gibberish anyway.

Okay, let's take a step back here...

It is, though. All I'm saying is, that's a double standard. If you like anime, but you don't like the kinds of games I make, then you're being biased toward anime games, and that's favoritism. That's, like, racist.

You don't have to try to talk me through your reasoning. I'm sure it makes perfect sense to you. But let's focus on how you can express your opinion in a way that—

How's that incendiary? I'm just stating a fact.

A fact? That sounded a lot like an opinion to me. Are you sure you know the difference between—

Sorry, I have to run. I need to catch the new episode of *My Little Pony: Friendship is Magic*. Hey, do you watch that show?

TRANSCRIPT #103B

But...But...They've done studies about this; I'll totally Google for them when I get home. They have, I remember it really clearly. It's this completely proven scientific fact that women are more—

Why don't you stop right there. I'm sorry, but arguing about this isn't going to get us where we need to be. I'm going to suggest instead that you simply avoid



talking about anything unrelated to the talking points in the packet. Okay? No

opinions, no "facts," nothing other than *DEMONS WITH GUNS 2* is going to be a great game, we're working hard on it, and it features a new multiplayer mode."

Well, To be honest, I think you're the one who doesn't understand. I mean, I know that, sure, politicians or whoever have to say things in a certain way. But this is the video game industry. It's part of the culture—

Shush. Did you hear me?

No talking.

I was just—

Shh.

Okay. Jeez. I—

Shh.

...

That's a good boy. Now go out there and talk about your game. 🐾

MATTHEW WASTELAND writes about games and game development on his blog, *Magical Wasteland* (www.magicalwasteland.com). email him at mwasteland@gdmag.com.

Get the innovative technology needed
to **unlock** the games of the future.



Award-winning technology.
Unparalleled support. Flexible licensing options.

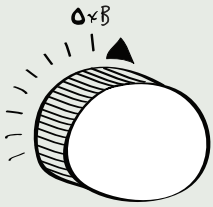
Learn More: www.havok.com

Havok is hiring! Visit www.havok.com/careers for more information.

Havok™ Technologies Include:

Havok Physics • Havok AI • Havok Animation • Havok Behavior • Havok Cloth • Havok Destruction • Havok Script • Havok Vision Engine

TURN UP THE



WICKED

AUDIO IN YOUR GAME WITH



Miles 9

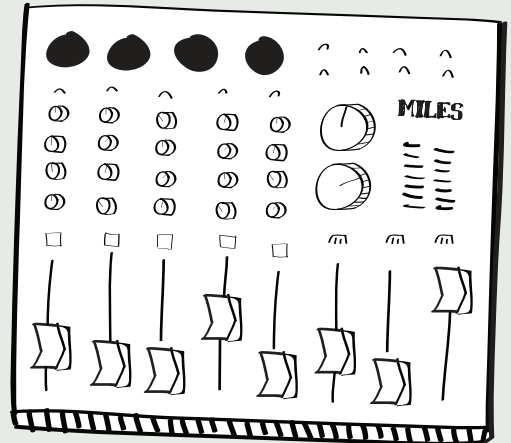
BRAND NEW!



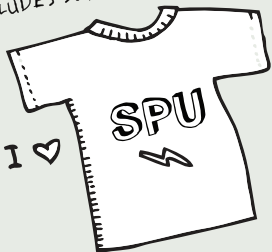
Our multichannel sound system features cool, new

high-level audio tools

+ the world's FASTEST



INCLUDES SUPPORT FOR THE PS3



MP3 and Ogg DECODERS.

USING MILES 9 NOT ONLY MAKES YOU

WANT TO TURN IT UP, IT MAKES YOU rad!



www.radgametools.com
(425) 893-4300